aws

Guide de l'utilisateur pour Aurora

Amazon Aurora

Amazon Aurora: Guide de l'utilisateur pour Aurora

Table of Contents

Qu’est-ce qu’Amazon Aurora ?

Amazon Aurora (Aurora) est un moteur de base de données relationnelle entièrement géré compatible avec MySQL et PostgreSQL. Vous savez déjà de quelle manière MySQL et PostgreSQL associent la vitesse et la fiabilité des bases de données commerciales haut de gamme à la simplicité et à la rentabilité des bases de données open source. Le code, les outils et les applications que vous utilisez aujourd’hui avec vos bases de données MySQL et PostgreSQL existantes peuvent être utilisés avec Aurora. Avec certaines charges de travail, Aurora peut offrir un débit jusqu’à cinq fois supérieur à celui de MySQL et jusqu’à trois fois supérieur à celui de PostgreSQL sans qu’il soit nécessaire de modifier la plupart de vos applications existantes.

Aurora comprend un sous-système de stockage très performant. Ses moteurs de bases de données compatibles avec MySQL et PostgreSQL sont personnalisés afin de tirer parti de ce stockage distribué et rapide. Le stockage sous-jacent évolue automatiquement en fonction des besoins. Un volume de cluster Aurora peut croître jusqu’à la taille maximale de 128 tébioctets (Tio). Aurora automatise et standardise également le clustering et la réplication des bases de données. Ces aspects figurent généralement parmi ceux qui représentent un défi dans le cadre de la configuration et de l’administration des bases de données.

Aurora fait partie du service de base de données géré Amazon Relational Database Service (Amazon RDS). Amazon RDS facilite la configuration, l’exploitation et la mise à l’échelle d’une base de données relationnelle dans le cloud. Si vous connaissez déjà Amazon RDS, consultez le Amazon Relational Database Service User Guide (Guide de l’utilisateur Amazon Relational Database Service). Pour en savoir plus sur les différentes options de bases de données disponibles sur Amazon Web Services, consultez Choisir la base de données adaptée à votre organisation sur AWS.

Rubriques

- Classes d’instance de base de données Amazon Aurora
- Stockage Amazon Aurora
- Fiabilité Amazon Aurora
- Sécurité Amazon Aurora
- Haute disponibilité pour Amazon Aurora
- Réplication avec Amazon Aurora
- Facturation d’une instance de base de données pour Aurora

Modèle de responsabilité partagée Amazon RDS

Amazon RDS est responsable de l’hébergement des composants logiciels et de l’infrastructure des instances de base de données et des clusters de bases de données. Vous êtes responsable du réglage des requêtes, qui consiste à ajuster les requêtes SQL afin d’améliorer les performances. Les performances des requêtes dépendent fortement de la conception de la base de données, de la taille des données, de la distribution des données, de la charge de travail des applications et des modèles de requêtes, qui peuvent varier considérablement. La surveillance et le réglage sont des processus hautement individualisés que vous possédez pour vos bases de données RDS. Vous pouvez utiliser l’analyse des performances d’Amazon RDS et d’autres outils pour identifier des requêtes problématiques.

Comment Amazon Aurora fonctionne avec Amazon RDS

Les éléments suivants illustrent de quelle manière Amazon Aurora est lié aux moteurs MySQL et PostgreSQL standard disponibles dans Amazon RDS :

- Vous choisissez Aurora MySQL ou Aurora PostgreSQL comme option de moteur de base de données lorsque vous configurez de nouveaux serveurs de base de données via Amazon RDS.
- Aurora tire parti des fonctions bien connues d’Amazon Relational Database Service (Amazon RDS) pour la gestion et l’administration. Aurora utilise l’interface AWS Management Console d’Amazon RDS, des commandes AWS CLI et des opérations d’API pour gérer des tâches de base de données de routine, telles que l’approvisionnement, l’application de correctifs, la sauvegarde, la récupération, la détection d’échecs et la réparation.
- Les opérations de gestion Aurora concernent généralement des clusters entiers de serveurs de bases de données qui sont synchronisés via des opérations de réplication, plutôt que des instances de base de données individuelles. Les fonctions automatiques de clustering, de

réplication et d'allocation du stockage facilitent et rentabilisent l'installation, l'exploitation et la mise en service de vos déploiements MySQL et PostgreSQL les plus importants.

- Vous pouvez transférer des données depuis Amazon RDS for MySQL et depuis Amazon RDS pour PostgreSQL dans Aurora en créant et en restaurant des instantanés, ou en configurant une réplication unilatérale. Vous pouvez utiliser des outils de migration à l'aide de boutons de commande pour convertir vos applications RDS for MySQL et RDS pour PostgreSQL existantes en applications Aurora.

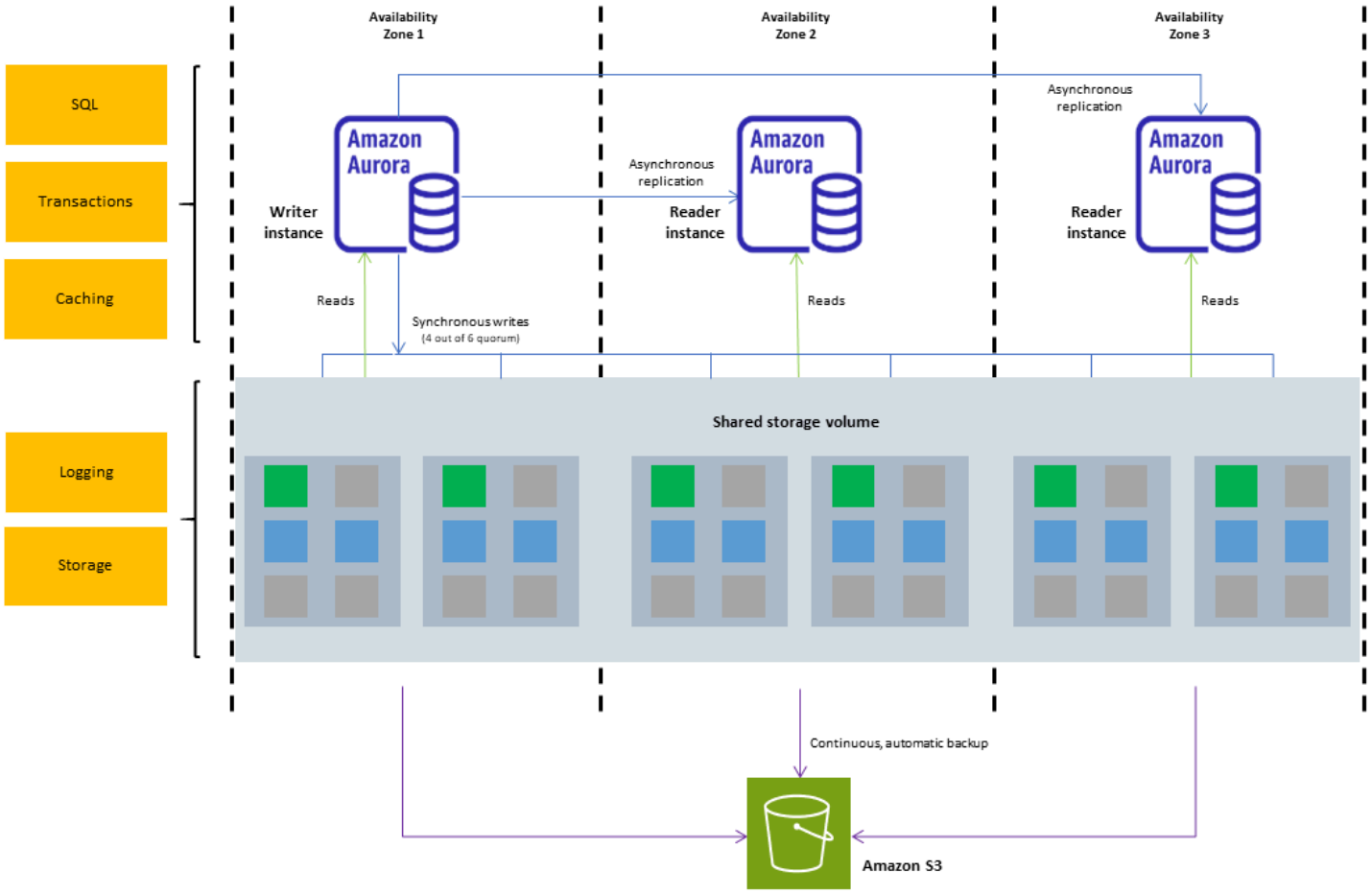
Avant d'utiliser Amazon Aurora, suivez la procédure dans Configuration de votre environnement pour Amazon Aurora, puis étudiez les concepts et les fonctions d'Aurora dans Clusters de bases de données Amazon Aurora.

Clusters de bases de données Amazon Aurora

Un cluster de bases de données Amazon Aurora se compose d'une ou plusieurs instances de base de données et d'un volume de cluster qui gère les données de ces instances. Un volume de cluster Aurora est un volume de stockage de base de données virtuel qui couvre plusieurs zones de disponibilité, chacune d'entre elles ayant une copie des données du cluster de bases de données. Deux types d'instances de base de données composent un cluster de bases de données Aurora :

- Instance de base de données principale (enregistreur) : prend en charge les opérations de lecture et d'écriture, et effectue toutes les modifications de données du volume de cluster. Chaque cluster de bases de données Aurora possède une seule instance de base de données principale.
- Réplica Aurora (instance de base de données de lecteur) : se connecte au même volume de stockage que l'instance de base de données principale et prend uniquement en charge les opérations de lecture. Chaque cluster de bases de données Aurora peut avoir jusqu'à 15 réplicas Aurora en plus de l'instance de base de données principale. Maintenez une haute disponibilité en plaçant les réplicas Aurora dans des zones de disponibilité distinctes. Aurora bascule automatiquement vers un réplica Aurora si l'instance de base de données principale devient indisponible. Vous pouvez spécifier la priorité de basculement pour les réplicas Aurora. Les réplicas Aurora peuvent également décharger l'instance de base de données principale des charges de travail en lecture.

Le schéma suivant illustre les relations entre le volume de cluster, l'instance de base de données d'enregistreur et les instances de base de données de lecteur dans un cluster Aurora.



> **Note**
>
> Les informations précédentes s'appliquent à tous les clusters de bases de données Aurora : provisionnés, requête parallèle, Aurora Global Database, Aurora Serverless, compatibles avec Aurora MySQL et compatibles avec Aurora PostgreSQL.

Le cluster de bases de données Aurora illustre la séparation de la capacité de calcul et du stockage. Par exemple, une configuration Aurora avec seulement une instance de base de données unique est quand même un cluster, car le volume de stockage sous-jacent implique plusieurs nœuds de stockage répartis entre plusieurs zones de disponibilité.

Les opérations d'entrée/sortie (E/S) dans les clusters de bases de données Aurora sont comptabilisées de la même manière, qu'elles se situent sur une instance de base de données d'écriture ou de lecture. Pour plus d'informations, consultez Configurations de stockage pour les clusters de bases de données Amazon Aurora.

Versions d'Amazon Aurora

Avec Amazon Aurora, vous pouvez choisir le moteur de base de données relationnelle pris en charge qui répond le mieux aux exigences de votre application tout en préservant la compatibilité avec les moteurs sous-jacents. Aurora réutilise le code des moteurs de base de données pris en charge, afin que vous puissiez tirer parti de leurs compétences, de leurs outils et de leurs bibliothèques. Lorsque vous créez un cluster, vous spécifiez la version du moteur de base de données Amazon Aurora que vous souhaitez utiliser. La version que vous choisissez détermine la compatibilité et les fonctionnalités disponibles.

Cette documentation explique les points communs et les différences entre Amazon Aurora et les moteurs de base de données correspondants. Grâce à ces informations, vous pourrez déterminer la version de logiciel à sélectionner et comment vérifier les fonctionnalités et les corrections de bogues disponibles dans chaque version. Vous pouvez également utiliser cette référence pour déterminer la cadence appropriée d'une mise à niveau et pour planifier cette dernière.

Moteurs de base de données pris en charge pour les clusters de bases de données Amazon Aurora

Les bases de données relationnelles disponibles sur Amazon Aurora sont les suivantes : Aurora réutilise le code et assure la compatibilité avec les moteurs de base de données sous-jacents. Cependant, Aurora a ses propres numéro de version, cycle de distribution et chronologie d'obsolescence. Chaque nouvelle version d'Aurora est fournie avec des notes de mise à jour qui énumèrent les nouvelles fonctionnalités, corrections, modifications et améliorations qui s'appliquent à chaque version.

Base de données Aurora	Guide de l'utilisateur	Versions disponibles	Notes de mise à jour
Amazon Aurora MySQL-Compatible Edition	Utilisation de Amazon Aurora MySQL	Mises à jour du moteur de base de données pour Amazon Aurora MySQL	Notes de mise à jour d'Aurora MySQL
Amazon Aurora PostgreSQL-Compatible Edition	Utilisation de Amazon Aurora PostgreSQL	Mises à jour du moteur de base de données pour	Notes de mise à jour d'Aurora PostgreSQL

Base de données Aurora	Guide de l’utilisateur	Versions disponibles	Notes de mise à jour
		Amazon Aurora PostgreSQL	

Spécification de la version de base de données Amazon Aurora pour votre cluster de bases de données

Lorsque vous créez un cluster de bases de données à l’aide de l’opération Créer une base de données dans la AWS Management Console, dans l’AWS CLI ou avec l’opération d’API `CreateDBCluster`, vous pouvez spécifier toute version de base de données Aurora actuellement disponible (majeure ou mineure).

Pour savoir comment créer des clusters Aurora, consultez Création d’un cluster de bases de données Amazon Aurora. Pour savoir comment modifier la version d’un cluster Aurora existant, consultez Modification d’un cluster de bases de données Amazon Aurora.

> **Note**
>
> Les versions de base de données Aurora ne sont pas toutes disponibles dans toutes les Région AWS. Pour en savoir plus sur les régions et les versions disponibles dans chaque Région AWS, consultez Régions et zones de disponibilité et Régions et moteurs de base de données pris en charge pour les bases de données globales Aurora.

Gestion des versions d’Amazon Aurora

Les versions d’Amazon Aurora sont différentes des bases de données communautaires en amont avec lesquelles elles sont compatibles. Pour vous aider à assurer la compatibilité des applications et à tirer parti des dernières fonctionnalités du moteur de base de données, les sections suivantes expliquent les conventions de gestion des versions d’Aurora et la manière dont les versions Aurora sont mappées à leurs bases de données communautaires respectives.

Pour obtenir la liste des bases de données relationnelles disponibles sur Amazon Aurora, consultez Moteurs de base de données pris en charge pour les clusters de bases de données Amazon Aurora.

Différences de numéros de version entre les bases de données communautaires et Aurora

Chaque version d'Amazon Aurora est compatible avec une version spécifique de sa base de données communautaire correspondante. Vous pouvez trouver la version communautaire de votre base de données à l'aide de la fonction `version`, et la version Aurora à l'aide de la fonction `aurora_version`.

Les exemples suivants montrent comment trouver la version communautaire de votre base de données pour Aurora MySQL et Aurora PostgreSQL.

Aurora MySQL

La fonction `version` renvoie la version communautaire de votre base de données pour Aurora MySQL.

```
mysql> select version();
```

Exemple de sortie :

```
+------------------+
|   version()      |
+------------------+
|  8.0.32          |
+------------------+
```

Et la fonction `aurora_version` renvoie la version Aurora :

```
mysql> select aurora_version(), @@aurora_version;
```

Exemple de sortie :

```
+------------------+------------------+
| aurora_version() | @@aurora_version |
+------------------+------------------+
| 3.05.2           | 3.05.2           |
+------------------+------------------+
```

Aurora PostgreSQL

La fonction `version` renvoie la version communautaire de votre base de données pour Aurora PostgreSQL.

```
postgres=> select version();
```

Exemple de sortie :

```
-----------------------------------------------------------------------------
PostgreSQL 11.7 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.9.3, 64-bit
(1 row)
```

Et la fonction `aurora_version` renvoie la version Aurora :

```
postgres=> select aurora_version();
```

Exemple de sortie :

```
aurora_version
----------------
3.2.2
```

Pour plus d'informations, consultez Vérification des versions Aurora MySQL avec SQL et Identification des versions Amazon Aurora PostgreSQL.

Versions par défaut d'Amazon Aurora

La version par défaut est la version qu'Aurora choisit automatiquement pour la création ou la mise à niveau de la base de données lorsque vous ne spécifiez pas manuellement la version du moteur cible. Par exemple, la commande suivante indique la version du moteur par défaut pour Aurora PostgreSQL (exemple de sortie inclus).

```
aws rds describe-db-engine-versions \
    --engine aurora-postgresql \
    --default-only \
    --query 'DBEngineVersions[0].EngineVersion' \
    --output text
```

```
16.4
```

À chaque version majeure correspond une version mineure par défaut. Par conséquent, la version mineure par défaut est 16.n pour Aurora PostgreSQL 16. Le numéro de version n change lors de la sortie de nouvelles versions mineures d'Aurora par défaut. En règle générale, Aurora publie deux versions par défaut pour chaque version majeure par an. Le script shell bash suivant montre les versions mineures par défaut pour un ensemble de versions majeures d'Aurora PostgreSQL (exemple de sortie inclus).

```
for major in 16 15 14 13 12 11; do
  echo -n "Default for Aurora PostgreSQL major version $major: "
  aws rds describe-db-engine-versions \
    --engine aurora-postgresql \
    --engine-version "$major" \
    --default-only \
    --query 'DBEngineVersions[0].EngineVersion' \
    --output text
done

Default for Aurora PostgreSQL major version 16: 16.4
Default for Aurora PostgreSQL major version 15: 15.8
Default for Aurora PostgreSQL major version 14: 14.13
Default for Aurora PostgreSQL major version 13: 13.16
Default for Aurora PostgreSQL major version 12: 12.20
Default for Aurora PostgreSQL major version 11: 11.21
```

Si vous activez les mises à niveau automatiques des versions mineures pour votre cluster de bases de données Aurora, Aurora utilise soit la version mineure par défaut, soit une version mineure plus récente correspondant à une version majeure donnée. Par exemple, si la version mineure par défaut d'Aurora PostgreSQL 15 est 15.8 et que la version 15.10 plus récente est également disponible, Aurora pourra effectuer une mise à niveau automatique vers la version 15.8 ou 15.10.

Versions majeures d'Amazon Aurora

Les versions d'Aurora utilisent le schéma *major*.*minor*.*patch*. Une version majeure d'Aurora fait référence à la version majeure communautaire MySQL ou PostgreSQL avec laquelle Aurora est compatible. Les versions majeures d'Aurora MySQL et d'Aurora PostgreSQL sont disponibles dans le cadre du support standard au moins jusqu'à la fin de vie de la version correspondante de la communauté. Vous pouvez continuer à exécuter une version majeure après la date de fin du support

standard Aurora moyennant des frais. Pour plus d'informations, consultez Support étendu Amazon RDS avec Amazon Aurora et Tarification d'Amazon Aurora.

Pour plus d'informations sur les versions majeures et le calendrier des versions d'Aurora MySQL et d'Aurora PostgreSQL, consultez les pages suivantes dans les notes de mise à jour respectives :

- Calendrier des versions majeures d'Aurora MySQL
- Calendrier des versions majeures d'Aurora PostgreSQL

Vous pouvez également consulter les dates de prise en charge des versions majeures du moteur en exécutant la commande AWS CLI describe-db-major-engine-versions ou en utilisant l'opération d'API RDS DescribeDBMajorEngineVersions.

> **Note**
>
> Le support étendu Amazon RDS pour Aurora MySQL version 2 commence le 1er novembre 2024, mais ne sera pas facturé avant le 1er décembre 2024. Entre le 1er et le 30 novembre 2024, tous les clusters de bases de données Aurora MySQL version 2 sont couverts par le support étendu Amazon RDS. Pour plus d'informations, consultez Support étendu Amazon RDS pour certaines versions d'Aurora.

Durée de disponibilité des versions majeures d'Amazon Aurora

Les versions majeures d'Amazon Aurora restent disponibles au moins jusqu'à la fin de vie de la version correspondante de la communauté. Vous pouvez utiliser les dates de fin du support standard Aurora pour planifier vos cycles de test et de mise à niveau. Ces dates représentent la première date à laquelle une mise à niveau vers une version plus récente pourrait être nécessaire. Pour plus d'informations sur les dates, consultez Versions majeures d'Amazon Aurora.

Avant qu'Aurora vous demande de passer à une version majeure plus récente, et pour vous aider à planifier, vous recevrez un rappel au moins 12 mois à l'avance. Ces rappels communiquent les informations suivantes concernant le processus de mise à niveau.

- Les dates et heures de certaines étapes clés
- L'impact sur vos clusters de bases de données
- Actions recommandées

Nous vous recommandons de tester soigneusement vos applications avec les nouvelles versions de base de données avant d'effectuer une mise à niveau de votre cluster vers une nouvelle version majeure.

Une fois que la version majeure a atteint la fin du support standard d'Aurora, tout cluster de bases de données exécutant encore la version précédente sera automatiquement mis à niveau vers une version de support étendu au cours d'une fenêtre de maintenance planifiée. Des frais de support étendu peuvent être facturés. Pour plus d'informations sur le support étendu Amazon RDS, consultez Utilisation du support étendu Amazon RDS.

Versions mineures d'Amazon Aurora

Les versions d'Aurora utilisent le schéma *major*.*minor*.*patch*. Une version mineure d'Aurora fournit des améliorations incrémentielles communautaires et spécifiques d'Aurora, telles que de nouvelles fonctions et des correctifs.

Pour plus d'informations sur les versions mineures et le calendrier des versions d'Aurora MySQL et d'Aurora PostgreSQL, consultez les pages suivantes dans les notes de mise à jour respectives :

- Calendrier des versions mineures d'Aurora MySQL
- Calendrier des versions mineures d'Aurora PostgreSQL

Les sections suivantes décrivent en détail la cadence et la durée de vie prévues des versions mineures d'Aurora.

Rubriques

- Fréquence de publication des versions mineures d'Amazon Aurora
- Durée de disponibilité des versions mineures d'Amazon Aurora

Fréquence de publication des versions mineures d'Amazon Aurora

En général, les versions mineures d'Amazon Aurora sont publiées chaque trimestre. Le calendrier de publication peut varier selon la nécessité d'intégrer des fonctions ou correctifs supplémentaires.

Durée de disponibilité des versions mineures d'Amazon Aurora

En règle générale, Amazon Aurora rend chaque version mineure d'une version majeure spécifique disponible pendant au moins 12 mois. À l'issue de cette période, il se peut qu'Aurora mette à

niveau automatiquement votre base de données vers la version mineure par défaut ou une version ultérieure. Aurora lancera la mise à niveau pendant la fenêtre de maintenance planifiée pour tout cluster de bases de données exécutant encore l'ancienne version mineure.

Dans certains cas, Aurora peut remplacer une version mineure d'une version majeure particulière avant la fin de la période habituelle de 12 mois. Les raisons peuvent inclure des problèmes de sécurité critiques ou la date de fin de support d'une version majeure.

Aurora envoie généralement un rappel trois mois avant de lancer les mises à niveau automatiques des versions mineures dont la fin de vie approche. Aurora détaille les éléments suivants concernant le processus de mise à niveau.

- Les dates et heures de certaines étapes clés
- L'impact sur vos clusters de bases de données
- Actions recommandées

Les notifications avec un préavis de moins de trois mois décrivent des problèmes critiques, tels que des problèmes de sécurité, qui nécessitent une action plus rapide.

Si le paramètre Mise à niveau automatique des versions mineures est activé, vous recevrez un rappel mais aucune notification d'événement RDS. Aurora met à niveau votre base de données pendant la fenêtre de maintenance qui suit la date d'échéance de la mise à niveau obligatoire.

Si le paramètre Mise à niveau automatique des versions mineures est désactivé, vous recevrez un rappel et un événement de cluster de bases de données Amazon RDS avec la catégorie `maintenance` et l'identifiant `RDS-EVENT-0156`. Aurora mettra à niveau votre base de données pendant la fenêtre de maintenance suivante.

Notez qu'une fois qu'une version mineure aura atteint la fin du support standard d'Aurora, aucune autre version de correctif ne sera publiée pour cette version. Pour pouvoir recevoir des corrections de bogues critiques ou des CVE, vous devrez passer à une version mineure avec support standard.

Pour plus d'informations sur les mises à niveau automatiques des versions mineures, consultez Mises à niveau automatiques des versions mineures pour les clusters de bases de données Aurora.

Versions correctives d'Amazon Aurora

Les versions d'Aurora utilisent le schéma *`major`*`.`*`minor`*`.`*`patch`*. Une version de correctifs d'Aurora inclut des correctifs importants ajoutés à une version mineure après sa sortie initiale (par

exemple, Aurora MySQL 3.04.0, 3.04.1, ..., 3.04.3). Si une nouvelle version mineure apporte de nouvelles fonctions Aurora, les nouvelles versions correctives d'une version mineure spécifique sont principalement utilisées pour résoudre des problèmes importants.

Pour plus d'informations sur l'application des correctifs, consultez Entretien d'un cluster de bases de données Amazon Aurora.

Mise à niveau des clusters de bases de données Amazon Aurora

Avec Amazon Aurora, vous pouvez contrôler et tester les mises à niveau de vos clusters de bases de données. Amazon Aurora fournit des options pour les mises à niveau automatiques des versions mineures, le contrôle manuel des mises à niveau, les mises à niveau obligatoires et les tests préalables à la mise à niveau. Vous pouvez maintenir vos clusters à jour avec la dernière version mineure, en différant les mises à niveau non critiques, en imposant les mises à niveau en cas de problème grave et en validant le comportement des mises à niveau dans les environnements hors production. Les sections suivantes expliquent comment gérer et tester les mises à niveau du cluster de bases de données Aurora à l'aide de ces fonctionnalités.

Mises à niveau automatiques des versions mineures d'Aurora

Les mises à niveau automatiques des versions mineures mettent régulièrement à jour votre base de données avec les versions récentes du moteur de base de données. Toutefois, la mise à niveau n'inclut pas toujours la dernière version du moteur de base de données. Si vous devez conserver des versions spécifiques de vos bases de données à un moment donné, nous vous recommandons de procéder à une mise à niveau manuelle vers les versions de base de données dont vous avez besoin conformément au calendrier requis. En cas de problèmes de sécurité critiques ou lorsqu'une version atteint sa date de fin de support, Amazon Aurora peut appliquer une mise à niveau de version mineure même si vous n'avez pas activé l'option Mise à niveau automatique des versions mineures. Pour plus d'informations, consultez la documentation de la mise à niveau de votre moteur de base de données spécifique.

Consultez Mise à niveau de la version mineure ou du niveau de correctif d'un cluster de bases de données Aurora MySQL et Réalisation d'une mise à niveau de version mineure.

Vous pouvez rester à jour avec les versions mineures d'Aurora en activant l'option Mise à niveau automatique des versions mineures pour chaque instance de base de données dans le cluster Aurora. Aurora n'effectue la mise à niveau automatique que si ce paramètre est activé pour toutes les instances de base de données de votre cluster.

Si l'option Mise à niveau automatique des versions mineures est définie sur Oui pour votre cluster de bases de données, Aurora passe automatiquement à la version mineure par défaut ou à une version mineure plus récente. Par exemple, si la version mineure par défaut est 15.8 pour Aurora PostgreSQL 15 et que la version 15.10 existe, la cible de la mise à niveau automatique pourra être 15.8 ou 15.10.

En général, Aurora planifie des mises à niveau automatiques deux fois par an pour les clusters de bases de données pour lesquels l'option de mise à niveau automatique des versions mineures est activée. Ces mises à niveau ont lieu pendant la fenêtre de maintenance que vous spécifiez pour votre cluster. Pour plus d'informations, consultez Mises à niveau automatiques des versions mineures pour les clusters de bases de données Aurora.

Les mises à niveau automatiques des versions mineures sont communiquées à l'avance via un événement de cluster de bases de données Amazon RDS avec une catégorie `maintenance` et un ID `RDS-EVENT-0156`. Pour plus d'informations, consultez Catégories d'événements et messages d'événements pour Aurora.

Contrôle manuel des mises à niveau des clusters de bases de données

Si le paramètre Mise à niveau automatique des versions mineures est activé, Aurora met automatiquement à niveau votre cluster de bases de données vers la version mineure par défaut ou vers une version mineure plus récente. En général, Aurora planifie des mises à niveau automatiques deux fois par an pour les clusters de bases de données pour lesquels le paramètre Mise à niveau automatique des versions mineures est activé. Ces mises à niveau sont lancées pendant les fenêtres de maintenance spécifiées par le client.

Si vous souhaitez désactiver la mise à niveau automatique des versions mineures, désactivez l'option Mise à niveau automatique des versions mineures sur les instances de base de données d'un cluster Aurora. Aurora n'effectue de mise à niveau automatique de version mineure que si ce paramètre est activé pour toutes les instances de base de données dans votre cluster.

> **Note**
>
> Pour les mises à niveau obligatoires telles que la fin de vie des versions mineures, Aurora met à niveau le cluster de bases de données même si le paramètre Mise à niveau automatique des versions mineures est désactivé. Vous recevrez un rappel mais aucune notification d'événement RDS. Aurora met à niveau votre cluster pendant la fenêtre de maintenance qui suit la date d'échéance de la mise à niveau obligatoire.

Les mises à niveau de version majeure présentant un risque en termes de compatibilité, elles ne se produisent pas automatiquement. Vous devez les lancer vous-même, sauf en cas d'obsolescence d'une mise à niveau majeure. Nous vous recommandons de tester soigneusement vos applications avec les nouvelles versions de base de données avant d'effectuer une mise à niveau de votre cluster vers une version majeure.

Pour plus d'informations sur la mise à niveau d'un cluster de bases de données vers une nouvelle version majeure d'Aurora, consultez Mise à niveau des clusters de bases de données Amazon Aurora MySQL et Mise à niveau des clusters de base de données Amazon Aurora PostgreSQL.

Mises à niveau d'Amazon Aurora obligatoires

Pour certains correctifs critiques, il peut arriver qu'Aurora effectue une mise à niveau gérée vers un correctif plus récent de la même version mineure. Dans ce cas, Aurora met à niveau votre cluster même si le paramètre Mise à niveau automatique des versions mineures est désactivé. Aurora vous communiquera au préalable les détails du processus de mise à niveau. Ces détails incluent les dates et heures de certaines étapes clés, l'impact sur vos clusters de bases de données, ainsi que les actions recommandées. Ces mises à niveau gérées ont lieu automatiquement pendant la fenêtre de maintenance du cluster.

Test de votre cluster de bases de données avec une nouvelle version d'Aurora avant la mise à niveau

Vous pouvez tester le processus de mise à niveau et le fonctionnement de la nouvelle version avec votre application et votre charge de travail. Utilisez l'une des méthodes suivantes :

- Clonez votre cluster à l'aide de la fonctionnalité de clonage rapide de base de données d'Amazon Aurora. Effectuez la mise à niveau et tous les tests subséquents sur le nouveau cluster.
- Opérez une restauration à partir d'un instantané de cluster pour créer un nouveau cluster Aurora. Vous pouvez créer vous-même un instantané de cluster à partir d'un cluster Aurora existant. Aurora crée aussi automatiquement pour vous des instantanés périodiques de chacun de vos clusters. Vous pouvez ensuite lancer une mise à niveau de version pour le nouveau cluster. Vous pouvez expérimenter la copie mise à niveau de votre cluster avant de décider de mettre à niveau le cluster d'origine.

Une version majeure d'Aurora Pour plus d'informations sur ces méthodes de création de clusters à des fins de test, consultez Clonage d'un volume pour un cluster de bases de données Amazon Aurora and Création d'un instantané de cluster de bases de données.

Versions Amazon Aurora prises en charge

Si votre base de données Amazon Aurora est étroitement liée au comportement d'un moteur de base de données spécifique, nous vous recommandons de procéder à des tests approfondis avant de passer aux nouvelles versions du moteur de base de données. Il existe des options de support à long terme qui vous permettent de conserver vos clusters de bases de données sur certaines versions du moteur Aurora, même lorsque des versions plus récentes les ont supplantées. Les sections suivantes décrivent les options de support à long terme pour vos clusters de bases de données Aurora.

Support à long terme pour certaines versions mineures d'Amazon Aurora

Pour chaque version majeure d'Aurora, certaines versions mineures désignées en tant que versions de support à long terme, ou versions LTS, sont mises à disposition pendant au moins trois ans. Autrement dit, au moins une version mineure par version majeure est rendue disponible plus longtemps que les 12 mois habituels. Aurora vous notifie généralement six mois avant la fin de cette période. Aurora communique les points suivants concernant le processus de mise à niveau.

- Les dates et heures de certaines étapes clés
- L'impact sur vos clusters de bases de données
- Actions recommandées

Les notifications avec un préavis de moins de six mois communiquent des problèmes critiques, tels que des problèmes de sécurité, qui nécessitent une action plus rapide.

Les versions mineures LTS incluent uniquement des correctifs critiques (via des versions correctives). Une version LTS n'inclut pas les nouvelles fonctions publiées après son introduction. Une fois par an, les clusters de bases de données s'exécutant sur une version mineure LTS sont mis à jour avec la dernière version de correctif de la version LTS. Aurora applique des correctifs à vos clusters afin que vous bénéficiiez des correctifs cumulatifs de sécurité et de stabilité. En cas de correctifs critiques affectant, par exemple, la sécurité, il peut arriver que nous corrigions plus fréquemment une version mineure LTS d'Aurora.

> **Note**
>
> Si vous souhaitez rester sur une version mineure LTS pendant toute la durée de son cycle de vie, veillez à désactiver la mise à niveau automatique des versions mineures pour vos instances de base de données. Pour éviter la mise à niveau automatique de votre cluster de bases de données à partir de la version mineure LTS, décochez la case Activer la mise à

niveau automatique des versions mineures pour toutes les instances de base de données de votre cluster Aurora.

Pour les numéros de version de toutes les versions de Aurora LTS, consultez Versions à long terme (LTS) d'Aurora MySQL et Utilisation d'une version LTS (Long-Term Support) d'Aurora PostgreSQL.

Support étendu Amazon RDS pour certaines versions d'Aurora

Avec le support étendu Amazon RDS, vous pouvez continuer à exécuter votre base de données sur une version majeure du moteur au-delà de la date de fin de support standard d'Aurora moyennant un coût supplémentaire. Dans le cadre du support étendu RDS, Amazon RDS fournit des correctifs pour les CVE critiques et élevées, conformément aux niveaux de gravité CVSS de la National Vulnerability Database (NVD). Pour plus d'informations, consultez Support étendu Amazon RDS avec Amazon Aurora.

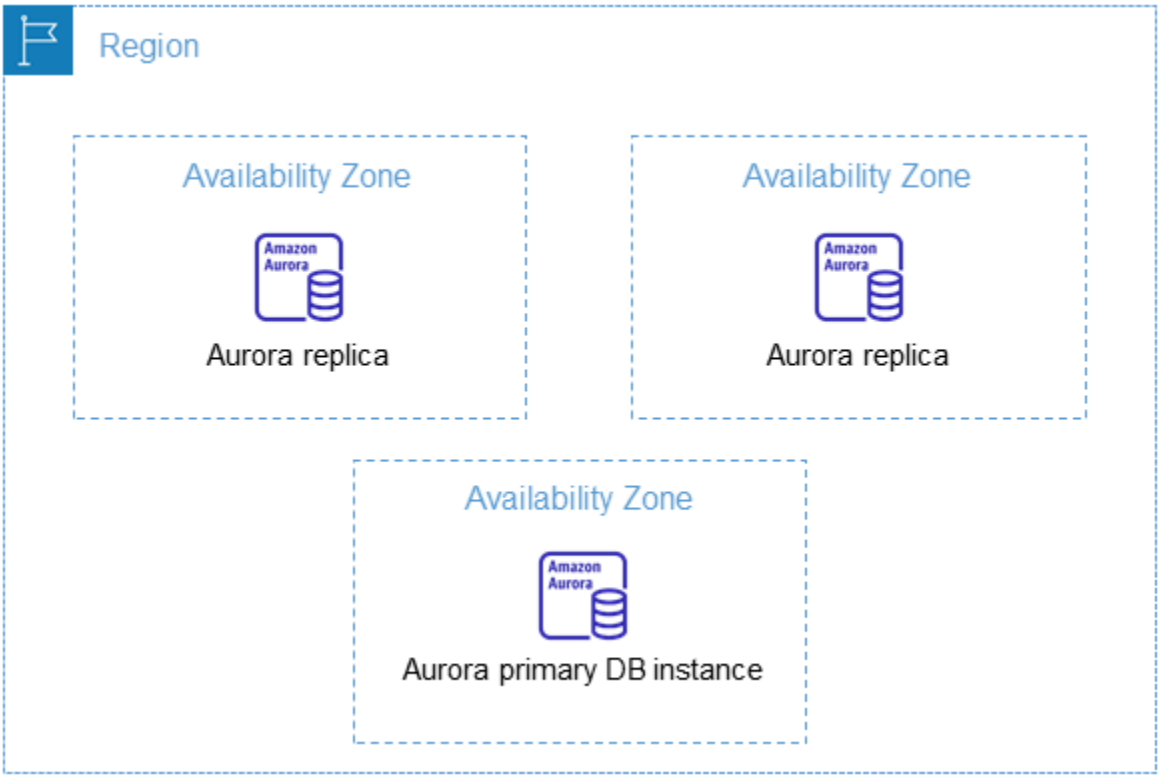
Le support étendu RDS n'est disponible que sur certaines versions mineures d'Aurora. Pour plus d'informations, consultez Versions majeures d'Amazon Aurora.

Régions et zones de disponibilité

Les ressources de cloud computing Amazon sont hébergées dans plusieurs emplacements à travers le monde. Ces emplacements sont composés de AWS régions et de zones de disponibilité. Chaque AWSrégion constitue une zone géographique séparée. Chaque AWS région possède plusieurs emplacements isolés appelés zones de disponibilité.

> **Note**
>
> Pour plus d'informations sur la recherche des zones de disponibilité d'une AWS région, consultez la section Décrire vos zones de disponibilité dans la EC2 documentation Amazon.

Amazon exploite state-of-the-art des centres de données hautement disponibles. Bien qu'elles soient rares, des pannes touchant la disponibilité des instances de base de données se trouvant au même emplacement peuvent se produire. Si vous hébergez toutes vos instances de base de données dans un seul emplacement touché par une panne de ce type, aucune de vos instances de base de données ne sera disponible.



Il est important de se rappeler que chaque AWS région est totalement indépendante. Toute activité Amazon RDS que vous lancez (par exemple, la création d'instances de base de données ou la liste des instances de base de données disponibles) s'exécute uniquement dans votre AWS région par défaut actuelle. La AWS région par défaut peut être modifiée dans la console ou en définissant la variable d'`AWS_DEFAULT_REGION`environnement. Il peut également être remplacé en utilisant le `--region` paramètre avec le AWS Command Line Interface ()AWS CLI. Pour plus d'informations, consultez Configuration de l'AWS Command Line Interface, plus précisément les sections sur les variables d'environnement et les options de ligne de commande.

Amazon RDS prend en charge les AWS régions spéciales appeléesAWS GovCloud (US). Elles sont conçues pour permettre aux agences gouvernementales et aux clients américains de déplacer des charges de travail plus sensibles vers le cloud. Les régions AWS GovCloud (US) aux exigences spécifiques du gouvernement américain en matière de réglementation et de conformité. Pour plus d'informations, voir Qu'est-ce que c'est AWS GovCloud (US) ?

Pour créer ou utiliser une instance de base de données Amazon RDS dans une AWS région spécifique, utilisez le point de terminaison de service régional correspondant.

> **Note**
>
> Aurora ne prend pas en charge les zones locales.

AWSRégions

Chaque AWS région est conçue pour être isolée des autres AWS régions. Cette conception permet d'atteindre la plus grande tolérance aux pannes possible et une stabilité optimale.

Lorsque vous consultez vos ressources, seules les ressources liées à la AWS région que vous avez spécifiée s'affichent. Cela est dû au fait que les AWS régions sont isolées les unes des autres et que nous ne répliquons pas automatiquement les ressources entre AWS les régions.

Disponibilité dans les Régions

Lorsque vous utilisez un cluster de bases de données Aurora à l'aide de l'interface de ligne de commande ou des opérations d'API, veillez à spécifier son point de terminaison régional.

Rubriques

- Aurora MySQL : disponibilité dans les régions

- Aurora PostgreSQL : disponibilité dans les régions

Aurora MySQL : disponibilité dans les régions

Le tableau suivant indique les AWS régions dans lesquelles Aurora MySQL est actuellement disponible et le point de terminaison de chaque région.

Nom de la région	Région	Point de terminaison	Protocole	
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
USA Est (Virginie du Nord)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
USA Ouest (Californie du Nord)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
Afrique (Le Cap)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
Asie-Pacifique (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	
Asie-Pacifique (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole
Asie-Pacifique (Jakarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS
Asie-Pacifique (Malaisie)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS
Asie-Pacifique (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS
Asia Pacific (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS
Asie-Pacifique (Nouvelle Zélande)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS
Asie-Pacifique (Osaka)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS
Asia Pacific (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS
Asie-Pacifique (Singapour)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS

Nom de la région	Région	Point de terminaison	Protocole	
Asia Pacific (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS	
Asie-Pacifique (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	
Asie-Pacifique (Thaïlande)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	
Asie-Pacifique (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS	
Canada (Centre)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS	
Canada-Ouest (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS	
Europe (Francfort)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS	
Europe (Irlande)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS	
Europe (Londres)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS	
Europe (Milan)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole
Europe (Paris)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS
Europe (Espagne)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS
Europe (Stockholm)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS
Europe (Zurich)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS
Israël (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS
Mexique (Centre)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
Moyen-Orient (Bahreïn)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS
Moyen-Orient (EAU)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS
Amérique du Sud (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS
AWSGovCloud (USA Est)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS

Nom de la région	Région	Point de terminaison	Protocole	
AWSGovCloud (US-Ouest)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS	

Aurora PostgreSQL : disponibilité dans les régions

Le tableau suivant indique les AWS régions dans lesquelles Aurora PostgreSQL est actuellement disponible et le point de terminaison de chaque région.

Nom de la région	Région	Point de terminaison	Protocole	
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
USA Est (Virginie du Nord)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS	
USA Ouest (Californie du Nord)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS	
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS	
Afrique (Le Cap)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS	
Asie-Pacifique (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
Asie-Pacifique (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS	
Asie-Pacifique (Jakarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com	HTTPS	
Asie-Pacifique (Malaisie)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS	
Asie-Pacifique (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com	HTTPS	
Asia Pacific (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com	HTTPS	
Asie-Pacifique (Nouvelle Zélande)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS	
Asie-Pacifique (Osaka)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com	HTTPS	
Asia Pacific (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
Asie-Pacifique (Singapour)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com	HTTPS	
Asia Pacific (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com	HTTPS	
Asie-Pacifique (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS	
Asie-Pacifique (Thaïlande)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS	
Asie-Pacifique (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com	HTTPS	
Canada (Centre)	ca-central-1	rds.ca-central-1.amazonaws.com	HTTPS	
Canada-Ouest (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com	HTTPS	
Europe (Francfort)	eu-central-1	rds.eu-central-1.amazonaws.com	HTTPS	
Europe (Irlande)	eu-west-1	rds.eu-west-1.amazonaws.com	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole
Europe (Londres)	eu-west-2	rds.eu-west-2.amazonaws.com	HTTPS
Europe (Milan)	eu-south-1	rds.eu-south-1.amazonaws.com	HTTPS
Europe (Paris)	eu-west-3	rds.eu-west-3.amazonaws.com	HTTPS
Europe (Espagne)	eu-south-2	rds.eu-south-2.amazonaws.com	HTTPS
Europe (Stockholm)	eu-north-1	rds.eu-north-1.amazonaws.com	HTTPS
Europe (Zurich)	eu-central-2	rds.eu-central-2.amazonaws.com	HTTPS
Israël (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com	HTTPS
Mexique (Centre)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
Moyen-Orient (Bahreïn)	me-south-1	rds.me-south-1.amazonaws.com	HTTPS
Moyen-Orient (EAU)	me-central-1	rds.me-central-1.amazonaws.com	HTTPS

Nom de la région	Région	Point de terminaison	Protocole	
Amérique du Sud (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com	HTTPS	
AWSGovCloud (USA Est)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com	HTTPS	
AWSGovCloud (US-Ouest)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com	HTTPS	

Zones de disponibilité

Une zone de disponibilité est un emplacement isolé dans une Région AWS donnée. Chaque région possède plusieurs zones de disponibilité (AZ, Availability Zone) conçues pour fournir une haute disponibilité pour la Région. Un AZ est identifié par le code de AWS région suivi d'une lettre d'identification (par exemple,`us-east-1a`). Si vous créez votre VPC et vos sous-réseaux plutôt que d'utiliser le VPC par défaut, vous définissez chaque sous-réseau dans une zone de disponibilité spécifique. Lorsque vous créez un cluster de bases de données Aurora, Aurora crée l'instance principale dans l'un des sous-réseaux du groupe de sous-réseaux de base de données du VPC. Il associe ainsi cette instance à une AZ spécifique choisi par Aurora.

Chaque cluster de base de données Aurora héberge des copies de son stockage en trois unités distinctes AZs sélectionnées automatiquement par Aurora AZs dans le groupe de sous-réseaux de votre base de données. Chaque instance de base de données du cluster doit se trouver dans l'une de ces trois instances AZs.

Lorsque vous créez une instance de base de données dans votre cluster, Aurora choisit automatiquement une zone de disponibilité appropriée pour cette instance si vous n'en spécifiez pas.

Utilisez la EC2 commande describe-availability-zonesAmazon comme suit pour décrire les zones de disponibilité activées pour votre compte dans la région spécifiée.

```
aws ec2 describe-availability-zones --region region-name
```

Par exemple, pour décrire les zones de disponibilité de la région USA Est (Virginie du Nord) (us-east-1) qui sont activées pour votre compte, exécutez la commande suivante :

```
aws ec2 describe-availability-zones --region us-east-1
```

Pour savoir comment spécifier la zone de disponibilité lorsque vous créez un cluster ou que vous y ajoutez des instances, consultez Configurer le réseau pour la base de données.

Fuseau horaire local pour les clusters de bases de données Amazon Aurora

Par défaut, le fuseau horaire d'un cluster de bases de données Amazon Aurora est le fuseau UTC (temps universel). Vous pouvez à la place définir le fuseau horaire des instances de votre cluster de bases de données sur le fuseau horaire local de votre application.

Pour définir le fuseau horaire local d'un cluster de bases de données, définissez le paramètre de fuseau horaire sur l'une des valeurs prises en charge. Vous définissez ce paramètre dans le groupe de paramètres du cluster pour votre cluster de bases de données.

- Pour Aurora MySQL, le nom de ce paramètre est `time_zone`. Pour plus d'informations sur les bonnes pratiques de définition du paramètre `time_zone`, consultez Optimisation des opérations d'horodatage.
- Pour Aurora PostgreSQL, le nom de ce paramètre est `timezone`.

Lorsque vous définissez le paramètre de fuseau horaire d'un cluster de bases de données, toutes les instances du cluster de bases de données changent pour utiliser le nouveau fuseau horaire local. Dans certains cas, d'autres clusters de bases de données Aurora peuvent utiliser le même groupe de paramètres de cluster. Si tel est le cas, toutes les instances de ces clusters de bases de données changent pour utiliser également le nouveau fuseau horaire local. Pour plus d'informations sur les paramètres de niveau cluster, consultez Paramètres de cluster de bases de données et d'instance de base de données Amazon Aurora.

Une fois que vous avez défini le fuseau horaire local, toutes les nouvelles connexions à la base de données reflètent la modification. Dans certains cas, des connexions à votre base de données sont ouvertes lorsque vous modifiez le fuseau horaire local. Si c'est le cas, la mise à jour du fuseau horaire local n'apparaît pas tant que vous n'avez pas fermé la connexion et ouvert une nouvelle.

Si vous effectuez une réplication entre AWS régions, le cluster de base de données source de réplication et la réplique utilisent des groupes de paramètres différents. Les groupes de paramètres sont uniques à une AWS région. Pour que chaque instance utilise le même fuseau horaire local, veillez à définir le paramètre de fuseau horaire dans les groupes de paramètres de la source de réplication et du réplica.

Lorsque vous restaurez un cluster de bases de données à partir d'un instantané de cluster de bases de données, le fuseau horaire local a la valeur UTC. Vous pouvez mettre à jour le fuseau horaire sur votre fuseau horaire local une fois la restauration terminée. Dans certains cas, vous pouvez restaurer un cluster de bases de données à un instant dans le passé. Dans ce cas, le fuseau horaire local du cluster de bases de données restauré est le paramètre de fuseau horaire du groupe de paramètres du cluster de bases de données restauré.

Le tableau suivant répertorie certaines valeurs sur lesquelles vous pouvez définir votre fuseau horaire local. Pour répertorier tous les fuseaux horaires disponibles, vous pouvez utiliser les requêtes SQL suivantes :

- Aurora MySQL : `select * from mysql.time_zone_name;`
- Aurora PostgreSQL : `select * from pg_timezone_names;`

> **Note**
>
> Pour certains fuseaux horaires, les valeurs de certaines plages de dates peuvent être mentionnées de façon incorrecte, comme noté dans le tableau. Pour les fuseaux horaires australiens, l'abréviation de fuseau horaire retournée est une valeur obsolète, comme noté dans le tableau.

Fuseau horaire	Remarques
`Africa/Harare`	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 28 février 1903 21:49:40 GMT et le 28 février 1903 21:55:48 GMT.
`Africa/Monrovia`	

Fuseau horaire	Remarques
`Africa/Nairobi`	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 31 décembre 1939 21:30:00 GMT et le 31 décembre 1959 21:15:15 GMT.
`Africa/Windhoek`	
`America/Bogota`	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 23 novembre 1914 04:56:16 GMT et le 23 novembre 1914 04:56:20 GMT.
`America/Caracas`	
`America/Chihuahua`	
`America/Cuiaba`	
`America/Denver`	
`America/Fortaleza`	Dans certains cas, pour un cluster de bases de données dans la région Amérique du Sud (São Paulo), l'heure ne s'affiche pas correctement pour un fuseau horaire récemment modifié du Brésil. Si tel est le cas, redéfinissez le paramètre de fuseau horaire du cluster de bases de données sur `America/Fortaleza` .
`America/Guatemala`	
`America/Halifax`	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 27 octobre 1918 05:00:00 GMT et le 31 octobre 1918 05:00:00 GMT.
`America/Manaus`	Si votre cluster de bases de données se trouve dans le fuseau horaire Amérique du Sud (Cuiaba) et que l'heure prévue ne s'affiche pas correctement pour le fuseau horaire récemment modifié du Brésil, réinitialisez le paramètre de fuseau horaire du cluster de bases de données sur `America/Manaus` .

Fuseau horaire	Remarques
America/Matamoros	
America/Monterrey	
America/Montevideo	
America/Noronha	
America/Phoenix	
America/Tijuana	
Asia/Ashgabat	
Asia/Baghdad	
Asia/Baku	
Asia/Bangkok	
Asia/Beirut	
Asia/Calcutta	
Asia/Kabul	
Asia/Karachi	
Asia/Kathmandu	
Asia/Muscat	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 31 décembre 1919 20:05:36 GMT et le 31 décembre 1919 20:05:40 GMT.
Asia/Riyadh	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 13 mars 1947 20:53:08 GMT et le 31 décembre 1949 20:53:08 GMT.

Fuseau horaire	Remarques
Asia/Seoul	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 30 novembre 1904 15:30:00 GMT et le 07 septembre 1945 15:00:00 GMT.
Asia/Shanghai	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 31 décembre 1927 15:54:08 GMT et le 02 juin 1940 16:00:00 GMT.
Asia/Singapore	
Asia/Taipei	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 30 septembre 1937 16:00:00 GMT et le 29 septembre 1979 15:00:00 GMT.
Asia/Tehran	
Asia/Tokyo	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 30 septembre 1937 15:00:00 GMT et le 31 décembre 1937 15:00:00 GMT.
Asia/Ulaanbaatar	
Atlantic/Azores	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 24 mai 1911 01:54:32 GMT et le 01 janvier 1912 01:54:32 GMT.
Australia/Adelaide	L’abréviation de ce fuseau horaire est retournée sous la forme CST au lieu d’ACDT/ACST.
Australia/Brisbane	L’abréviation de ce fuseau horaire est retournée sous la forme EST au lieu d’AEDT/AEST.
Australia/Darwin	L’abréviation de ce fuseau horaire est retournée sous la forme CST au lieu d’ACDT/ACST.
Australia/Hobart	L’abréviation de ce fuseau horaire est retournée sous la forme EST au lieu d’AEDT/AEST.

Fuseau horaire	Remarques
`Australia/Perth`	L'abréviation de ce fuseau horaire est renvoyée sous la forme WST au lieu de AWDT/AWST.
`Australia/Sydney`	L’abréviation de ce fuseau horaire est retournée sous la forme EST au lieu d’AEDT/AEST.
`Brazil/East`	
`Canada/Saskatchewan`	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 27 octobre 1918 08:00:00 GMT et le 31 octobre 1918 08:00:00 GMT.
`Europe/Amsterdam`	
`Europe/Athens`	
`Europe/Dublin`	
`Europe/Helsinki`	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 30 avril 1921 22:20:08 GMT et le 30 avril 1921 22:20:11 GMT.
`Europe/Paris`	
`Europe/Prague`	
`Europe/Sarajevo`	
`Pacific/Auckland`	
`Pacific/Guam`	
`Pacific/Honolulu`	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 21 mai 1933 11:30:00 GMT et le 30 septembre 1945 11:30:00 GMT.
`Pacific/Samoa`	Ce fuseau horaire peut retourner des valeurs incorrectes entre le 01 janvier 1911 11:22:48 GMT et le 01 janvier 1950 11:30:00 GMT.
`US/Alaska`	

Fuseau horaire	Remarques
US/Central	
US/Eastern	
US/East-Indiana	
US/Pacific	
UTC	

Fonctionnalités prises en charge dans Amazon Aurora by Région AWS et dans le moteur de base de données Aurora

Aurora Les moteurs de base de données compatibles MySQL et PostgreSQL prennent en charge plusieurs fonctionnalités et options Amazon Aurora et Amazon RDS. La prise en charge varie selon les versions spécifiques de chaque moteur de base de données et entre les Régions AWS. Pour identifier la prise en charge et la disponibilité des versions du moteur de base de données Aurora pour une fonctionnalité donnée Région AWS, vous pouvez utiliser les sections suivantes.

Certaines de ces fonctionnalités sont des capacités Aurora uniquement. Par exemple, Aurora Serverless, les bases de données Aurora globales et la prise en charge de l'intégration avec les services de machine learning AWS ne sont pas pris en charge par Amazon RDS. D'autres fonctionnalités, telles que Amazon RDS Proxy, sont prises en charge par Amazon Aurora et Amazon RDS.

Régions et moteurs de base de données pris en charge

- Conventions de tableau
- Régions et moteurs de base de données Aurora pris en charge pour les déploiements bleu/vert
- Régions et moteurs de base de données Aurora pris en charge pour les configurations de stockage en cluster
- Régions et moteurs de base de données Aurora pris en charge pour les flux d'activité de base de données
- Régions et moteurs de bases de données Aurora pris en charge pour l'exportation des données du cluster vers Amazon S3
- Régions et moteurs de bases de données Aurora pris en charge pour l'exportation des données d'instantanés vers Amazon S3
- Régions et moteurs de base de données pris en charge pour les bases de données globales Aurora
- Régions et moteurs de base de données Aurora pris en charge pour l'authentification de base de données IAM
- Régions et moteurs de base de données Aurora pris en charge pour l'authentification Kerberos
- Régions et moteurs de bases de données pris en charge pour le machine learning Aurora
- Régions et moteurs de base de données Aurora pris en charge pour Performance Insights

- Régions et moteurs de base de données Aurora pris en charge pour les intégration zéro ETL d'Amazon RDS
- Régions et moteurs de base de données Aurora pris en charge pour Proxy Amazon RDS
- Régions et moteurs de base de données Aurora pris en charge pour l'intégration de Secrets Manager
- Régions et moteurs de base de données Aurora pris en charge pour Aurora Serverless v2
- Aurora Serverless v1
- Régions et moteurs de base de données Aurora pris en charge pour l'API de données RDS
- Régions et moteurs de base de données Aurora pris en charge pour l'application de correctifs sans durée d'indisponibilité
- Régions prises en charge pour la base de données Aurora PostgreSQL Limitless
- Régions et moteurs de base de données pris en charge pour les fonctionnalités natives du moteur Aurora

Conventions de tableau

Les tableaux dans les sections utilisent les modèles suivants pour spécifier les numéros de version et le niveau de prise en charge :

- Version x.y – Seule cette version spécifique est prise en charge.
- Version x.y et ultérieures : la version spécifiée et toutes les versions mineures ultérieures de cette version majeure sont prises en charge. Par exemple, « versions 10.11 et ultérieures » signifie que les versions 10.11, 10.11.1 et 10.12 sont prises en charge.

Régions et moteurs de base de données Aurora pris en charge pour les déploiements bleu/vert

Un blue/green déploiement copie un environnement de base de données de production dans un environnement intermédiaire distinct et synchronisé. En utilisant Amazon RDS Blue/Green Deployments, vous pouvez apporter des modifications à la base de données dans l'environnement intermédiaire sans affecter l'environnement de production. Par exemple, vous pouvez mettre à niveau la version majeure ou mineure du moteur de base de données, modifier les paramètres de la base de données ou apporter des changements au schéma dans l'environnement intermédiaire. Lorsque vous êtes prêt, vous pouvez promouvoir l'environnement intermédiaire en tant que nouvel environnement

de base de données de production. Pour plus d'informations, consultez Utilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données.

Déploiements bleu/vert avec Aurora MySQL

La fonctionnalité Blue/Green Déploiements est disponible pour toutes les versions d'Aurora MySQL Régions AWS, y compris les clusters Aurora MySQL configurés en tant que base de données Aurora Global.

Déploiements bleu/vert avec Aurora PostgreSQL

Les régions et versions de moteur suivantes sont disponibles pour les Blue/Green déploiements avec Aurora PostgreSQL, y compris les clusters Aurora PostgreSQL configurés en tant que base de données globale Aurora.

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Tout Régions AWS	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Versions 11.21 et ultérieures

Régions et moteurs de base de données Aurora pris en charge pour les configurations de stockage en cluster

Amazon Aurora dispose de deux configurations de stockage pour les clusters de bases de données : Aurora I/O-Optimized and Aurora Standard. Pour plus d'informations, consultez Configurations de stockage pour les clusters de bases de données Amazon Aurora.

Aurora I/O-Optimized

Aurora I/O-Optimized est disponible dans toutes les Régions AWS pour les versions suivantes d'Amazon Aurora :

- Aurora MySQL versions 3.03.1 et ultérieures

- Aurora PostgreSQL versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.7 et ultérieures, 13.10 et ultérieures

Aurora Standard

Aurora Standard est disponible dans toutes les Régions AWS pour toutes les versions d'Aurora MySQL et d'Aurora PostgreSQL.

Régions et moteurs de base de données Aurora pris en charge pour les flux d'activité de base de données

En utilisant les flux d'activité des bases de données dans Aurora, vous pouvez surveiller et définir des alarmes pour auditer l'activité de votre base de données Aurora. Pour plus d'informations, consultez Surveillance d'Amazon Aurora à l'aide des flux d'activité de base de données.

Les flux d'activité de base de données ne sont pas pris en charge pour les fonctionnalités suivantes :

- Aurora Serverless v1
- Aurora Serverless v2
- Babelfish for Aurora PostgreSQL

Rubriques

- Flux d'activité de base de données avec Aurora MySQL
- Flux d'activité de la base de données avec Aurora PostgreSQL

Flux d'activité de base de données avec Aurora MySQL

Les régions et les versions de moteur suivantes sont disponibles pour les flux d'activité de la base de données avec Aurora MySQL.

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Est (Virginie du Nord)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Est (Ohio)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
USA Ouest (Californie du Nord)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
USA Ouest (Oregon)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Afrique (Le Cap)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Asie-Pacifique (Hong Kong)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Asie-Pacifique (Hyderabad)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Asie-Pacifique (Jakarta)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Asie-Pacifique (Malaisie)	Non disponible	Non disponible
Asie-Pacifique (Melbourne)	Non disponible	Non disponible
Asie-Pacifique (Mumbai)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible	Non disponible
Asie-Pacifique (Osaka)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Asie-Pacifique (Séoul)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
Asie-Pacifique (Singapour)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Asie-Pacifique (Sydney)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Asie-Pacifique (Taipei)	Non disponible	Non disponible
Asie-Pacifique (Thaïlande)	Non disponible	Non disponible
Asie-Pacifique (Tokyo)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Canada (Centre)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Canada-Ouest (Calgary)	Non disponible	Non disponible
Chine (Pékin)	Non disponible	Non disponible
Chine (Ningxia)	Non disponible	Non disponible
Europe (Francfort)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Europe (Irlande)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Europe (Londres)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Europe (Milan)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Europe (Paris)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
Europe (Espagne)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Europe (Stockholm)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Europe (Zurich)	Non disponible	Non disponible
Israël (Tel Aviv)	Non disponible	Non disponible
Middle East (Bahrain)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Mexique (Centre)	Non disponible	Non disponible
Moyen-Orient (EAU)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures
Amérique du Sud (São Paulo)	Toutes versions disponibles	Aurora versions 2.11 et ultérieures

Flux d’activité de la base de données avec Aurora PostgreSQL

Les régions et les versions de moteur suivantes sont disponibles pour les flux d’activité de la base de données avec Aurora PostgreSQL.

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
USA Est (Virginie du Nord)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
USA Est (Ohio)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
USA Ouest (Californie du Nord)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
USA Ouest (Oregon)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Afrique (Le Cap)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Hong Kong)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Hyderabad)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Jakarta)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Malaisie)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	
Asie-Pacifique (Melbourne)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Mumbai)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Osaka)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Séoul)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Singapour)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Sydney)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Taipei)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Thaïlande)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Tokyo)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Canada (Centre)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Canada-Ouest (Calgary)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Chine (Pékin)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Chine (Ningxia)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Europe (Francfort)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Europe (Irlande)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Europe (Londres)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europe (Milan)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Europe (Paris)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Europe (Espagne)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Europe (Stockholm)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Europe (Zurich)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Israël (Tel Aviv)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Mexique (Centre)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Middle East (Bahrain)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Moyen-Orient (EAU)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures
Amérique du Sud (São Paulo)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Toutes versions disponibles	Toutes versions disponibles	Toutes versions disponibles	Version 11.9 et versions 11.13 et ultérieures

Régions et moteurs de bases de données Aurora pris en charge pour l'exportation des données du cluster vers Amazon S3

Vous pouvez exporter les données du cluster de base Aurora vers un compartiment Amazon S3. Une fois les données exportées, vous pouvez les analyser directement via des outils tels que Amazon Athena ou Amazon Redshift Spectrum. Pour de plus amples informations, veuillez consulter Exportation des données du cluster de bases de données vers Amazon S3.

L'exportation des données du cluster vers S3 est disponible dans les Régions AWS suivantes :

- USA Est (Virginie du Nord)
- USA Est (Ohio)
- USA Ouest (Californie du Nord)
- US West (Oregon)
- Asie-Pacifique (Hong Kong)
- Asie-Pacifique (Hyderabad)
- Asie-Pacifique (Jakarta)
- Asie-Pacifique (Melbourne)
- Asie-Pacifique (Mumbai)
- Asie-Pacifique (Nouvelle-Zélande)
- Asie-Pacifique (Osaka)
- Asia Pacific (Seoul)
- Asie-Pacifique (Singapour)
- Asie-Pacifique (Sydney)
- Asie-Pacifique (Tokyo)
- Canada (Centre)
- Canada-Ouest (Calgary)
- Chine (Ningxia)
- Europe (Francfort)
- Europe (Irlande)
- Europe (Londres)
- Europe (Paris)
- Europe (Espagne)

- Europe (Stockholm)
- Europe (Zurich)
- Israël (Tel Aviv)
- Moyen-Orient (EAU)
- Amérique du Sud (São Paulo)

Rubriques

Exportation des données du cluster vers S3 avec Aurora MySQL

Toutes les versions du moteur Aurora MySQL actuellement disponibles prennent en charge l'exportation des données du cluster de bases de données vers Amazon S3. Pour obtenir plus d'informations sur les versions, consultez Notes de mise à jour d'Aurora MySQL.

Exportation des données du cluster vers S3 avec Aurora PostgreSQL

Toutes les versions du moteur Aurora PostgreSQL actuellement disponibles prennent en charge l'exportation des données du cluster de bases de données vers Amazon S3. Pour plus d'informations sur les versions, consultez Notes de mise à jour d'Aurora PostgreSQL.

Régions et moteurs de bases de données Aurora pris en charge pour l'exportation des données d'instantanés vers Amazon S3

Vous pouvez exporter des données d'instantanés de cluster de bases de données Aurora vers un compartiment Amazon S3. Vous pouvez exporter des instantanés manuels et des instantanés système automatisés. Une fois les données exportées, vous pouvez les analyser directement via des outils tels que Amazon Athena ou Amazon Redshift Spectrum. Pour de plus amples informations, veuillez consulter Exportation de données d'instantanés de cluster de bases de données vers Amazon S3.

L'exportation des instantanés vers S3 est disponible dans tous les domaines, Régions AWS sauf dans les domaines suivants :

- Asie-Pacifique (Malaisie)

- Asie-Pacifique (Nouvelle-Zélande)
- Asie-Pacifique (Taipei)
- Asie-Pacifique (Thaïlande)
- Mexique (Centre)
- AWS GovCloud (USA Est)
- AWS GovCloud (US-Ouest)

Rubriques

- Exportation de données d’instantanés vers S3 avec Aurora MySQL
- Exportation de données d’instantanés vers S3 avec Aurora PostgreSQL

Exportation de données d’instantanés vers S3 avec Aurora MySQL

Toutes les versions du moteur Aurora MySQL actuellement disponibles prennent en charge l’exportation des données d’instantanés du cluster de bases de données vers Amazon S3. Pour obtenir plus d’informations sur les versions, consultez Notes de mise à jour d’Aurora MySQL.

Exportation de données d’instantanés vers S3 avec Aurora PostgreSQL

Toutes les versions du moteur Aurora PostgreSQL actuellement disponibles prennent en charge l’exportation des données d’instantanés du cluster de bases de données vers Amazon S3. Pour plus d’informations sur les versions, consultez Notes de mise à jour d’Aurora PostgreSQL.

Régions et moteurs de base de données pris en charge pour les bases de données globales Aurora

Une base de données globale Aurora est une base de données unique qui couvre plusieurs bases de donnéesRégions AWS, ce qui permet des lectures globales à faible latence et une reprise après sinistre en cas de panne à l'échelle de la région. Il fournit une tolérance aux pannes intégrée pour votre déploiement, car l'instance de base de données ne repose pas sur une seuleRégion AWS, mais sur plusieurs régions et différentes zones de disponibilité. Pour de plus amples informations, veuillez consulter Utilisation d’Amazon Aurora Global Database.

Rubriques

- Bases de données Aurora globales avec Aurora MySQL

- Bases de données globales Aurora avec Aurora PostgreSQL

Bases de données Aurora globales avec Aurora MySQL

Les régions et les versions de moteur suivantes sont disponibles pour les bases de données globales Aurora avec Aurora MySQL.

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Est (Virginie du Nord)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
USA Est (Ohio)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
USA Ouest (Californie du Nord)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
USA Ouest (Oregon)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Afrique (Le Cap)	Versions 3.01.0 et ultérieures	Versions 2.07.1 et ultérieures
Asie-Pacifique (Hong Kong)	Versions 3.01.0 et ultérieures	Versions 2.07.1 et ultérieures
Asie-Pacifique (Hyderabad)	Versions 3.02.0 et ultérieures	Versions 2.11.2 et ultérieures
Asie-Pacifique (Jakarta)	Versions 3.01.0 et ultérieures	Versions 2.07.6 et ultérieures
Asie-Pacifique (Malaisie)	Versions 3.04.0 et ultérieures	Versions 2.07.6 et ultérieures
Asie-Pacifique (Melbourne)	Versions 3.03.0 et ultérieures	Versions 2.07.6 et ultérieures
Asie-Pacifique (Mumbai)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Asie-Pacifique (Nouvelle Zélande)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Asie-Pacifique (Osaka)	Versions 3.01.0 et ultérieures	Versions 2.07.3 et ultérieures
Asie-Pacifique (Séoul)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Asie-Pacifique (Singapour)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
Asie-Pacifique (Sydney)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Asie-Pacifique (Taipei)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Asie-Pacifique (Thaïlande)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Asie-Pacifique (Tokyo)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Canada (Centre)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Canada-Ouest (Calgary)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Chine (Pékin)	Versions 3.01.0 et ultérieures	Versions 2.07.2 et ultérieures
Chine (Ningxia)	Versions 3.01.0 et ultérieures	Versions 2.07.2 et ultérieures
Europe (Francfort)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Europe (Irlande)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Europe (Londres)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Europe (Milan)	Versions 3.01.0 et ultérieures	Versions 2.07.1 et ultérieures
Europe (Paris)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Europe (Espagne)	Versions 3.02.0 et ultérieures	Versions 2.07.6 et ultérieures
Europe (Stockholm)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
Europe (Zurich)	Versions 3.02.0 et ultérieures	Versions 2.07.6 et ultérieures
Israël (Tel Aviv)	Versions 3.02.0 et ultérieures	Versions 2.07.6 et ultérieures
Mexique (Centre)	Versions 3.02.0 et ultérieures	Versions 2.07.6 et ultérieures
Middle East (Bahrain)	Versions 3.01.0 et ultérieures	Versions 2.07.1 et ultérieures
Moyen-Orient (EAU)	Versions 3.02.0 et ultérieures	Versions 2.07.6 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
Amérique du Sud (São Paulo)	Versions 3.01.0 et ultérieures	Versions 2.07.1 et ultérieures
AWSGovCloud (USA Est)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures
AWSGovCloud (US-Ouest)	Versions 3.01.0 et ultérieures	Versions 2.07.0 et ultérieures

Bases de données globales Aurora avec Aurora PostgreSQL

Les régions et les versions de moteur suivantes sont disponibles pour les bases de données globales Aurora avec Aurora PostgreSQL.

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
USA Est (Virginie du Nord)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
USA Est (Ohio)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
USA Ouest (Californie du Nord)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
USA Ouest (Oregon)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Afrique (Le Cap)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Hong Kong)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Hyderabad)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Jakarta)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Malaisie)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Melbourne)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Mumbai)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Nouvelle Zélande)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Osaka)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Séoul)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Singapour)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Sydney)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Sydney)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Taipei)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Thaïlande)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Tokyo)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Canada (Centre)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Canada-Ouest (Calgary)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Chine (Pékin)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Chine (Ningxia)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europe (Francfort)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Irlande)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Londres)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Milan)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europe (Paris)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Espagne)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Stockholm)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Zurich)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Israël (Tel Aviv)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Mexique (Centre)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Middle East (Bahrain)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Moyen-Orient (EAU)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Amérique du Sud (São Paulo)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
AWSGovCloud (USA Est)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
AWSGovCloud (US-Ouest)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Régions et moteurs de base de données Aurora pris en charge pour l'authentification de base de données IAM

Avec l'authentification de base de données IAM dans Aurora, vous pouvez vous authentifier auprès de votre cluster de bases de données à l'aide de l'authentification de base de données AWS Identity and Access Management (IAM). Grâce à cette méthode d'authentification, vous n'avez plus besoin de mot de passe pour vous connecter à un cluster de bases de données. En revanche, un jeton d'authentification est nécessaire. Pour plus d'informations, consultez Authentification de base de données IAM.

Rubriques

- Authentification de bases de données IAM avec Aurora MySQL
- Authentification des bases de données IAM avec Aurora PostgreSQL

Authentification de bases de données IAM avec Aurora MySQL

L'authentification de bases de données IAM avec Aurora MySQL est disponible dans toutes les régions pour les versions suivantes :

- Aurora MySQL 3 : toutes versions disponibles
- Aurora MySQL 2 : toutes versions disponibles

Authentification des bases de données IAM avec Aurora PostgreSQL

L'authentification des bases de données IAM avec Aurora PostgreSQL est disponible dans toutes les régions pour les versions de moteur suivantes :

- Aurora PostgreSQL 17 : toutes versions disponibles
- Aurora PostgreSQL 16 : toutes versions disponibles
- Aurora PostgreSQL 15 : toutes versions disponibles
- Aurora PostgreSQL 14 : toutes versions disponibles
- Aurora PostgreSQL 13 : toutes versions disponibles
- Aurora PostgreSQL 12 : toutes versions disponibles
- Aurora PostgreSQL 11 : toutes versions disponibles

Régions et moteurs de base de données Aurora pris en charge pour l'authentification Kerberos

En utilisant l'authentification Kerberos avec Aurora, vous pouvez prendre en charge l'authentification externe des utilisateurs de la base de données en utilisant Kerberos et Microsoft Active Directory. L'utilisation de Kerberos et Active Directory procure les avantages d'une authentification unique et centralisée des utilisateurs de bases de données. Kerberos et Active Directory sont disponibles avec AWS Directory Service for Microsoft Active Directory, une fonctionnalité de. Directory Service Pour de plus amples informations, veuillez consulter Authentification Kerberos.

Rubriques

- Authentification Kerberos avec Aurora MySQL
- Authentification Kerberos avec Aurora PostgreSQL
- Groupes de sécurité Active Directory (AD) avec Aurora PostgreSQL

Authentification Kerberos avec Aurora MySQL

Les régions et les versions de moteur suivantes sont disponibles pour l'authentification Kerberos avec Aurora MySQL.

Région	Aurora MySQL version 3
USA Est (Virginie du Nord)	Versions 3.03.0 et ultérieures
USA Est (Ohio)	Versions 3.03.0 et ultérieures
USA Ouest (Californie du Nord)	Versions 3.03.0 et ultérieures
USA Ouest (Oregon)	Versions 3.03.0 et ultérieures
Afrique (Le Cap)	Non disponible
Asie-Pacifique (Hong Kong)	Non disponible
Asie-Pacifique (Jakarta)	Non disponible
Asie-Pacifique (Malaisie)	Non disponible
Asie-Pacifique (Melbourne)	Non disponible
Asie-Pacifique (Mumbai)	Versions 3.03.0 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible
Asie-Pacifique (Osaka)	Non disponible
Asie-Pacifique (Séoul)	Versions 3.03.0 et ultérieures
Asie-Pacifique (Singapour)	Versions 3.03.0 et ultérieures
Asie-Pacifique (Sydney)	Versions 3.03.0 et ultérieures

Région	Aurora MySQL version 3
Asie-Pacifique (Taipei)	Non disponible
Asie-Pacifique (Thaïlande)	Non disponible
Asie-Pacifique (Tokyo)	Versions 3.03.0 et ultérieures
Canada (Centre)	Versions 3.03.0 et ultérieures
Canada-Ouest (Calgary)	Non disponible
Chine (Pékin)	Versions 3.03.0 et ultérieures
Chine (Ningxia)	Versions 3.03.0 et ultérieures
Europe (Francfort)	Versions 3.03.0 et ultérieures
Europe (Irlande)	Versions 3.03.0 et ultérieures
Europe (Londres)	Versions 3.03.0 et ultérieures
Europe (Milan)	Non disponible
Europe (Paris)	Versions 3.03.0 et ultérieures
Europe (Espagne)	Non disponible
Europe (Stockholm)	Versions 3.03.0 et ultérieures
Europe (Zurich)	Non disponible
Israël (Tel Aviv)	Non disponible
Mexique (Centre)	Non disponible
Middle East (Bahrain)	Non disponible
Moyen-Orient (EAU)	Non disponible
Amérique du Sud (São Paulo)	Versions 3.03.0 et ultérieures

Région	Aurora MySQL version 3
AWS GovCloud (USA Est)	Versions 3.03.0 et ultérieures
AWS GovCloud (US-Ouest)	Versions 3.03.0 et ultérieures

Authentification Kerberos avec Aurora PostgreSQL

Les régions et les versions de moteur suivantes sont disponibles pour l'authentification Kerberos avec Aurora PostgreSQL.

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
USA Est (Virginie du Nord)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
USA Est (Ohio)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
USA Ouest (Californie du Nord)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
USA Ouest (Oregon)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Afrique (Le Cap)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Hong Kong)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Hyderabad)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Jakarta)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Malaisie)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Melbourne)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Mumbai)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Nouvelle-Zélande)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Osaka)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Séoul)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Singapour)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Sydney)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Taipei)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Thaïlande)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Tokyo)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Canada (Centre)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Canada-Ouest (Calgary)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Chine (Pékin)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Chine (Ningxia)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Francfort)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Irlande)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Londres)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Milan)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Paris)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Espagne)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europe (Stockholm)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Zurich)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Israël (Tel Aviv)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Mexique (Centre)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Middle East (Bahrain)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Moyen-Orient (EAU)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Amérique du Sud (São Paulo)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
AWS GovCloud (USA Est)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
AWS GovCloud (US-Ouest)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Groupes de sécurité Active Directory (AD) avec Aurora PostgreSQL

Les régions et versions de moteur suivantes sont disponibles pour ActiveDirectory Aurora PostgreSQL.

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
USA Est (Virginie du Nord)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
USA Est (Ohio)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
USA Ouest (Californie du Nord)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
USA Ouest (Oregon)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Afrique (Le Cap)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Hong Kong)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Hyderabad)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Jakarta)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Malaisie)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Melbourne)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Mumbai)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Nouvelle-Zélande)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Osaka)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Séoul)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Singapour)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Sydney)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Taipei)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Thaïlande)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Tokyo)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Canada (Centre)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Canada-Ouest (Calgary)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Chine (Pékin)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Chine (Ningxia)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Francfort)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Irlande)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Londres)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Milan)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Paris)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europe (Espagne)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Stockholm)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Zurich)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Israël (Tel Aviv)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Mexique (Centre)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Middle East (Bahrain)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Moyen-Orient (EAU)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Amérique du Sud (São Paulo)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
AWS GovCloud (USA Est)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
AWS GovCloud (US-Ouest)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Régions et moteurs de bases de données pris en charge pour le machine learning Aurora

En utilisant l'apprentissage automatique Amazon Aurora, vous pouvez intégrer votre cluster de base de données Aurora à l'un des services d'apprentissage AWS automatique suivants, en fonction de vos besoins. Ils prennent chacun en charge des cas d'utilisation propres au machine learning.

Amazon Bedrock est un service entièrement géré qui met à disposition les modèles de fondation de premier plan des entreprises d'IA à l'aide d'une API, ainsi que des outils de développement pour aider à créer et à mettre à l'échelle des applications d'IA générative.

Amazon Comprehend est un service géré de traitement du langage naturel (NLP) utilisé pour extraire des informations à partir de documents. En utilisant le machine learning Aurora avec Amazon Comprehend, vous pouvez déterminer le sentiment du texte dans les tables de votre base de données.

SageMaker L'IA est un service complet d'apprentissage automatique. Les data scientists utilisent Amazon SageMaker AI pour créer, former et tester des modèles d'apprentissage automatique pour diverses tâches d'inférence, telles que la détection des fraudes. En utilisant l'apprentissage automatique Aurora avec l' SageMaker IA, les développeurs de bases de données peuvent invoquer la fonctionnalité d' SageMaker IA dans le code SQL.

Tous ne sont pas Régions AWS compatibles avec tous les services d'apprentissage automatique. Seules certaines Régions AWS prennent en charge le machine learning Aurora et fournissent ainsi

un accès à ces services à partir d'un cluster de bases de données Aurora. Le processus d'intégration pour le machine learning Aurora diffère également selon le moteur de base de données. Pour plus d'informations, consultez Utilisation du machine learning Amazon Aurora.

Rubriques

- Machine Learning Aurora avec Aurora MySQL
- Machine Learning Aurora avec Aurora PostgreSQL

Machine Learning Aurora avec Aurora MySQL

Amazon Bedrock est uniquement pris en charge sur Aurora MySQL 3.06 ou version ultérieure. Pour plus d'informations sur les régions disponibles pour Amazon Bedrock, consultez Modèles pris en charge par Région AWS dans le Guide de l'utilisateur Amazon Bedrock.

L'apprentissage automatique Aurora avec Amazon Comprehend et Amazon SageMaker AI est pris en charge pour Aurora MySQL comme Régions AWS indiqué dans le tableau. En plus de disposer de votre version d'Aurora MySQL, elle Région AWS doit également prendre en charge le service que vous souhaitez utiliser. Pour obtenir une liste des Régions AWS endroits où Amazon SageMaker AI est disponible, consultez la section Points de terminaison et quotas Amazon SageMaker AI dans le Référence générale d'Amazon Web Services. Pour obtenir la liste des Régions AWS endroits où Amazon Comprehend est disponible, consultez la section Points de terminaison et quotas Amazon Comprehend dans le. Référence générale d'Amazon Web Services

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Est (Virginie du Nord)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
USA Est (Ohio)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
USA Ouest (Californie du Nord)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
USA Ouest (Oregon)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Afrique (Le Cap)	Non disponible	Non disponible
Asie-Pacifique (Hong Kong)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
Asie-Pacifique (Hyderabad)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Asie-Pacifique (Jakarta)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Asie-Pacifique (Malaisie)	Versions 3.04.0 et ultérieures	Non disponible
Asie-Pacifique (Melbourne)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Asie-Pacifique (Mumbai)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible	Non disponible
Asie-Pacifique (Osaka)	Versions 3.01.0 et ultérieures	Versions 2.07.3 et ultérieures
Asie-Pacifique (Séoul)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Asie-Pacifique (Singapour)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Asie-Pacifique (Sydney)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Asie-Pacifique (Taipei)	Non disponible	Non disponible
Asie-Pacifique (Thaïlande)	Non disponible	Non disponible
Asie-Pacifique (Tokyo)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Canada (Centre)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Canada-Ouest (Calgary)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Chine (Pékin)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Chine (Ningxia)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Europe (Francfort)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Europe (Irlande)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Europe (Londres)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
Europe (Milan)	Non disponible	Non disponible
Europe (Paris)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Europe (Espagne)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Europe (Stockholm)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Europe (Zurich)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Israël (Tel Aviv)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Mexique (Centre)	Non disponible	Non disponible
Middle East (Bahrain)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Moyen-Orient (EAU)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
Amérique du Sud (São Paulo)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
AWS GovCloud (USA Est)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures
AWS GovCloud (US-Ouest)	Versions 3.01.0 et ultérieures	Versions 2.07 et ultérieures

Machine Learning Aurora avec Aurora PostgreSQL

Pour plus d'informations sur les versions d'Amazon Bedrock prises en charge sur Aurora PostgreSQL, consultez Utilisation d'Aurora PostgreSQL comme base de connaissances pour Amazon Bedrock.

L'apprentissage automatique Aurora avec Amazon Comprehend et Amazon SageMaker AI est pris en charge pour Aurora PostgreSQL comme indiqué dans le tableau. Régions AWS Outre la disponibilité de votre version d'Aurora PostgreSQL, elle doit également prendre en charge Région AWS le service que vous souhaitez utiliser. Pour obtenir une liste des Régions AWS endroits où Amazon SageMaker AI est disponible, consultez la section Points de terminaison et quotas Amazon SageMaker AI dans le Référence générale d'Amazon Web Services. Pour obtenir la liste des Régions AWS endroits où Amazon Comprehend est disponible, consultez la section Points de terminaison et quotas Amazon Comprehend dans le. Référence générale d'Amazon Web Services

Les régions et les versions de moteur suivantes sont disponibles pour le machine learning Aurora avec Aurora PostgreSQL.

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
USA Est (Virginie du Nord)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
USA Est (Ohio)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
USA Ouest (Californie du Nord)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
USA Ouest (Oregon)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Afrique (Le Cap)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique	Versions 17 et	Versions 16 et	Versions 15 et	Versions 14 et	Versions 13.3 et	Versions 12.4 et	Versions 11.9, 11.12 et

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
(Hong Kong)	ultérieures	ultérieures	ultérieures	ultérieures	ultérieures	ultérieures	ultérieures
Asie-Pacifique (Hyderabad)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Asie-Pacifique (Jakarta)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Asie-Pacifique (Malaisie)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Asie-Pacifique (Melbourne)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Asie-Pacifique (Mumbai)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Nouvelle-Zélande)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Asie-Pacifique (Osaka)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Asie-Pacifique (Séoul)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Asie-Pacifique (Singapour)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Asie-Pacifique (Sydney)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Taipei)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Asie-Pacifique (Thaïlande)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Asie-Pacifique (Tokyo)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Canada (Centre)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Canada-Ouest (Calgary)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Chine (Pékin)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Chine (Ningxia)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Europe (Francfort)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Europe (Irlande)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Europe (Londres)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Europe (Milan)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europe (Paris)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Europe (Espagne)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Europe (Stockholm)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Europe (Zurich)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Israël (Tel Aviv)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Mexique (Centre)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Middle East (Bahrain)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Moyen-Orient (EAU)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
Amérique du Sud (São Paulo)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
AWS GovCloud (USA Est)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures
AWS GovCloud (US-Ouest)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13.3 et ultérieures	Versions 12.4 et ultérieures	Versions 11.9, 11.12 et ultérieures

Régions et moteurs de base de données Aurora pris en charge pour Performance Insights

Important

AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l'expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L'API Performance Insights continuera d'exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.

Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.

Si vous n'effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode Standard de Database Insights, vous risquez de perdre l'accès à l'historique des données de performance au-delà de 7 jours et de ne pas être en mesure d'utiliser les plans d'exécution et les fonctionnalités d'analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.

Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance à grande échelle. Pour plus d'informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

Performance Insights développe les fonctions de surveillance existantes d'Amazon RDS pour illustrer et vous aider à analyser les performances de votre base de données. Avec le tableau de bord Performance Insights, vous pouvez visualiser la charge de la base de données de votre charge d'instance de base de données Amazon RDS et filtrer la charge par attentes, instructions SQL, hôtes ou utilisateurs. Pour plus d'informations, consultez Présentation de Performance Insights sur Amazon Aurora.

Pour obtenir des informations de prise en charge de la région, du moteur de base de données et des classes d'instance pour les fonctionnalités d'analyse des performances, consultez Prise en charge de la classe d'instances, de la région et du moteur de base de données Amazon Aurora pour les fonctionnalités d'analyse des performances.

Rubriques

- Performance Insights avec Aurora MySQL
- Performance Insights avec Aurora PostgreSQL
- Performance Insights avec Aurora sans serveur

Performance Insights avec Aurora MySQL

> **Note**
>
> La prise en charge des versions du moteur est différente pour Performance Insights with Aurora MySQL si la requête parallèle est activée. Pour plus d'informations sur la requête parallèle, consultez Requêtes parallèles pour Amazon Aurora MySQL.

Rubriques

- Performance Insights avec Aurora MySQL et requête parallèle désactivée
- Performance Insights avec Aurora MySQL et requête parallèle activée

Performance Insights avec Aurora MySQL et requête parallèle désactivée

Les régions et les versions de moteur suivantes sont disponibles pour Performance Insights avec Aurora MySQL et la requête parallèle désactivée.

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Est (Virginie du Nord)	Toutes les versions	Toutes les versions
USA Est (Ohio)	Toutes les versions	Toutes les versions
USA Ouest (Californie du Nord)	Toutes les versions	Toutes les versions

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Ouest (Oregon)	Toutes les versions	Toutes les versions
Afrique (Le Cap)	Toutes les versions	Toutes les versions
Asie-Pacifique (Hong Kong)	Toutes les versions	Toutes les versions
Asie-Pacifique (Hyderabad)	Toutes les versions	Toutes les versions
Asie-Pacifique (Jakarta)	Toutes les versions	Toutes les versions
Asie-Pacifique (Malaisie)	Toutes les versions	Toutes les versions
Asie-Pacifique (Melbourne)	Toutes les versions	Toutes les versions
Asie-Pacifique (Mumbai)	Toutes les versions	Toutes les versions
Asie-Pacifique (Nouvelle-Zélande)	Toutes les versions	Toutes les versions
Asie-Pacifique (Osaka)	Toutes les versions	Toutes les versions
Asie-Pacifique (Séoul)	Toutes les versions	Toutes les versions
Asie-Pacifique (Singapour)	Toutes les versions	Toutes les versions
Asie-Pacifique (Sydney)	Toutes les versions	Toutes les versions
Asie-Pacifique (Taipei)	Toutes les versions	Toutes les versions
Asie-Pacifique (Thaïlande)	Toutes les versions	Toutes les versions
Asie-Pacifique (Tokyo)	Toutes les versions	Toutes les versions
Canada (Centre)	Toutes les versions	Toutes les versions
Canada-Ouest (Calgary)	Toutes les versions	Toutes les versions
Chine (Pékin)	Toutes les versions	Toutes les versions
Chine (Ningxia)	Toutes les versions	Toutes les versions

Région	Aurora MySQL version 3	Aurora MySQL version 2
Europe (Francfort)	Toutes les versions	Toutes les versions
Europe (Irlande)	Toutes les versions	Toutes les versions
Europe (Londres)	Toutes les versions	Toutes les versions
Europe (Milan)	Toutes les versions	Toutes les versions
Europe (Paris)	Toutes les versions	Toutes les versions
Europe (Espagne)	Toutes les versions	Toutes les versions
Europe (Stockholm)	Toutes les versions	Toutes les versions
Europe (Zurich)	Toutes les versions	Toutes les versions
Israël (Tel Aviv)	Toutes les versions	Toutes les versions
Mexique (Centre)	Toutes les versions	Toutes les versions
Middle East (Bahrain)	Toutes les versions	Toutes les versions
Moyen-Orient (EAU)	Toutes les versions	Toutes les versions
Amérique du Sud (São Paulo)	Toutes les versions	Toutes les versions
AWS GovCloud (USA Est)	Toutes les versions	Toutes les versions
AWS GovCloud (US-Ouest)	Toutes les versions	Toutes les versions

Performance Insights avec Aurora MySQL et requête parallèle activée

Les régions et les versions de moteur suivantes sont disponibles pour Performance Insights avec Aurora MySQL et la requête parallèle activée.

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Est (Virginie du Nord)	Non disponible	Versions 2.09.0 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Est (Ohio)	Non disponible	Versions 2.09.0 et ultérieures
USA Ouest (Californie du Nord)	Non disponible	Versions 2.09.0 et ultérieures
USA Ouest (Oregon)	Non disponible	Versions 2.09.0 et ultérieures
Afrique (Le Cap)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Hong Kong)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Hyderabad)	Non disponible	Toutes les versions
Asie-Pacifique (Jakarta)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Malaisie)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Melbourne)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Mumbai)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Osaka)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Séoul)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Singapour)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Sydney)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Taipei)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Thaïlande)	Non disponible	Versions 2.09.0 et ultérieures
Asie-Pacifique (Tokyo)	Non disponible	Versions 2.09.0 et ultérieures
Canada (Centre)	Non disponible	Versions 2.09.0 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
Canada-Ouest (Calgary)	Non disponible	Versions 2.09.0 et ultérieures
Chine (Pékin)	Non disponible	Versions 2.09.0 et ultérieures
Chine (Ningxia)	Non disponible	Versions 2.09.0 et ultérieures
Europe (Francfort)	Non disponible	Versions 2.09.0 et ultérieures
Europe (Irlande)	Non disponible	Versions 2.09.0 et ultérieures
Europe (Londres)	Non disponible	Versions 2.09.0 et ultérieures
Europe (Milan)	Non disponible	Versions 2.09.0 et ultérieures
Europe (Paris)	Non disponible	Versions 2.09.0 et ultérieures
Europe (Espagne)	Non disponible	Versions 2.09.0 et ultérieures
Europe (Stockholm)	Non disponible	Versions 2.09.0 et ultérieures
Europe (Zurich)	Non disponible	Versions 2.09.0 et ultérieures
Israël (Tel Aviv)	Non disponible	Versions 2.09.0 et ultérieures
Mexique (Centre)	Non disponible	Versions 2.09.0 et ultérieures
Middle East (Bahrain)	Non disponible	Versions 2.09.0 et ultérieures
Moyen-Orient (EAU)	Non disponible	Versions 2.09.0 et ultérieures
Amérique du Sud (São Paulo)	Non disponible	Versions 2.09.0 et ultérieures
AWS GovCloud (USA Est)	Non disponible	Versions 2.09.0 et ultérieures
AWS GovCloud (US-Ouest)	Non disponible	Versions 2.09.0 et ultérieures

Performance Insights avec Aurora PostgreSQL

Les régions et les versions de moteur suivantes sont disponibles pour Performance Insights avec Aurora PostgreSQL.

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
USA Est (Virginie du Nord)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
USA Est (Ohio)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
USA Ouest (Californie du Nord)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
USA Ouest (Oregon)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Afrique (Le Cap)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Hong Kong)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Asie-Pacifique (Hyderabad)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Jakarta)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Malaisie)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Melbourne)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Mumbai)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Nouvelle-Zélande)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Asie-Pacifique (Osaka)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Séoul)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Singapour)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Sydney)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Taipei)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Asie-Pacifique (Thaïlande)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Asie-Pacifique (Tokyo)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Canada (Centre)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Canada-Ouest (Calgary)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Chine (Pékin)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Chine (Ningxia)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Francfort)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Irlande)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Londres)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Europe (Milan)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Paris)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Espagne)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Stockholm)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Europe (Zurich)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Israël (Tel Aviv)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Mexique (Centre)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Middle East (Bahrain)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
Moyen-Orient (EAU)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11	Aurora PostgreSQL 10
Amérique du Sud (São Paulo)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
AWS GovCloud (USA Est)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions
AWS GovCloud (US-Ouest)	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions	Toutes les versions

Performance Insights avec Aurora sans serveur

Aurora Serverless v2 prend en charge Performance Insights pour toutes les versions compatibles avec MySQL et PostgreSQL. Nous vous recommandons de définir la capacité minimale à au moins 2 unités de capacité Aurora (ACUs).

Aurora Serverless v1 ne prend pas en charge Performance Insights.

Régions et moteurs de base de données Aurora pris en charge pour les intégration zéro ETL d'Amazon RDS

Les intégrations zéro ETL d'Amazon Aurora représentent une solution entièrement gérée qui permet de rendre les données transactionnelles disponibles dans Amazon Redshift ou Amazon SageMaker après leur écriture dans un cluster Aurora. Pour plus d'informations, consultez Intégrations sans ETL.

Rubriques

- Intégrations zéro ETL d'Aurora MySQL
- Intégrations zéro ETL d'Aurora PostgreSQL

Intégrations zéro ETL d’Aurora MySQL

Les régions et versions de moteur suivantes sont disponibles pour les intégrations zéro ETL d’Aurora MySQL à Amazon Redshift et Amazon SageMaker.

Région	Intégration zéro ETL à Amazon Redshift pour Aurora MySQL version 3	Intégration zéro ETL à Amazon SageMaker Aurora MySQL version 3
USA Est (Virginie du Nord)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
USA Est (Ohio)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
USA Ouest (Californie du Nord)	Versions 3.05.2 et ultérieures	Non disponible
USA Ouest (Oregon)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
Afrique (Le Cap)	Versions 3.05.2 et ultérieures	Non disponible
Asie-Pacifique (Hong Kong)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
Asie-Pacifique (Hyderabad)	Versions 3.05.2 et ultérieures	Non disponible
Asie-Pacifique (Jakarta)	Versions 3.05.2 et ultérieures	Non disponible
Asie-Pacifique (Malaisie)	Non disponible	Non disponible
Asie-Pacifique (Melbourne)	Versions 3.05.2 et ultérieures	Non disponible
Asie-Pacifique (Mumbai)	Versions 3.05.2 et ultérieures	Non disponible
Asie-Pacifique (Nouvelle Zélande)	Non disponible	Non disponible
Asie-Pacifique (Osaka)	Versions 3.05.2 et ultérieures	Non disponible
Asie-Pacifique (Séoul)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
Asie-Pacifique (Singapour)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures

Région	Intégration zéro ETL à Amazon Redshift pour Aurora MySQL version 3	Intégration zéro ETL à Amazon SageMaker Aurora MySQL version 3
Asie-Pacifique (Sydney)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
Asie-Pacifique (Taipei)	Non disponible	Non disponible
Asie-Pacifique (Thaïlande)	Non disponible	Non disponible
Asie-Pacifique (Tokyo)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
Canada (Centre)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
Canada-Ouest (Calgary)	Versions 3.05.2 et ultérieures	Non disponible
Chine (Pékin)	Versions 3.05.2 et ultérieures	Non disponible
Chine (Ningxia)	Versions 3.05.2 et ultérieures	Non disponible
Europe (Francfort)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
Europe (Irlande)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
Europe (Londres)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
Europe (Milan)	Versions 3.05.2 et ultérieures	Non disponible
Europe (Paris)	Versions 3.05.2 et ultérieures	Non disponible
Europe (Espagne)	Versions 3.05.2 et ultérieures	Non disponible
Europe (Stockholm)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
Europe (Zurich)	Versions 3.05.2 et ultérieures	Non disponible
Israël (Tel Aviv)	Versions 3.05.2 et ultérieures	Non disponible
Mexique (Centre)	Non disponible	Non disponible
Middle East (Bahrain)	Versions 3.05.2 et ultérieures	Non disponible

Région	Intégration zéro ETL à Amazon Redshift pour Aurora MySQL version 3	Intégration zéro ETL à Amazon SageMaker Aurora MySQL version 3
Moyen-Orient (EAU)	Versions 3.05.2 et ultérieures	Non disponible
Amérique du Sud (São Paulo)	Versions 3.05.2 et ultérieures	Versions 3.05.2 et ultérieures
AWS GovCloud (USA Est)	Non disponible	Non disponible
AWS GovCloud (US-Ouest)	Non disponible	Non disponible

Intégrations zéro ETL d’Aurora PostgreSQL

Les régions et versions du moteur suivantes sont disponibles pour les intégrations Aurora PostgreSQL Zero-ETL avec Amazon Redshift et. Amazon Sagemaker

Région	Intégration zéro ETL avec Amazon Redshift pour Aurora PostgreSQL version 17	Intégration zéro ETL avec Amazon SageMaker Aurora PostgreSQL version 17	Intégration zéro ETL avec Amazon Redshift pour Aurora PostgreSQL version 16	Intégration zéro ETL avec Amazon SageMaker Aurora PostgreSQL version 16
USA Est (Ohio)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
USA Est (Virginie du Nord)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
USA Ouest (Californie du Nord)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
USA Ouest (Oregon)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures

Région	Intégration zéro ETL avec Amazon Redshift pour Aurora PostgreSQL version 17	Intégration zéro ETL avec Amazon SageMaker Aurora PostgreSQL version 17	Intégration zéro ETL avec Amazon Redshift pour Aurora PostgreSQL version 16	Intégration zéro ETL avec Amazon SageMaker Aurora PostgreSQL version 16
Afrique (Le Cap)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Asie-Pacifique (Hong Kong)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
Asie-Pacifique (Hyderabad)	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Non disponible	Non disponible
Asie-Pacifique (Jakarta)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Asie-Pacifique (Melbourne)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Asie-Pacifique (Malaisie)	Non disponible	Non disponible		
Asie-Pacifique (Malaisie)	Non disponible	Non disponible		
Asie-Pacifique (Mumbai)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Asie-Pacifique (Nouvelle Zélande)	Non disponible	Non disponible		
Asie-Pacifique (Osaka)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible

Région	Intégration zéro ETL avec Amazon Redshift pour Aurora PostgreSQL version 17	Intégration zéro ETL avec Amazon SageMaker Aurora PostgreSQL version 17	Intégration zéro ETL avec Amazon Redshift pour Aurora PostgreSQL version 16	Intégration zéro ETL avec Amazon SageMaker Aurora PostgreSQL version 16
Asie-Pacifique (Séoul)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
Asie-Pacifique (Singapour)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
Asie-Pacifique (Sydney)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
Asie-Pacifique (Taipei)	Non disponible	Non disponible		
Asie-Pacifique (Thaïlande)	Non disponible	Non disponible		
Asie-Pacifique (Taipei)	Non disponible	Non disponible		
Asie-Pacifique (Thaïlande)	Non disponible	Non disponible		
Asie-Pacifique (Tokyo)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
Canada (Centre)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
Canada-Ouest (Calgary)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible

Région	Intégration zéro ETL avec Amazon Redshift pour Aurora PostgreSQL version 17	Intégration zéro ETL avec Amazon SageMaker Aurora PostgreSQL version 17	Intégration zéro ETL avec Amazon Redshift pour Aurora PostgreSQL version 16	Intégration zéro ETL avec Amazon SageMaker Aurora PostgreSQL version 16
Chine (Pékin)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Chine (Ningxia)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Europe (Francfort)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
Europe (Irlande)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
Europe (Londres)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
Europe (Milan)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Europe (Paris)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Europe (Espagne)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Europe (Stockholm)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
Europe (Zurich)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible

Région	Intégration zéro ETL avec Amazon Redshift pour Aurora PostgreSQL version 17	Intégration zéro ETL avec Amazon SageMaker Aurora PostgreSQL version 17	Intégration zéro ETL avec Amazon Redshift pour Aurora PostgreSQL version 16	Intégration zéro ETL avec Amazon SageMaker Aurora PostgreSQL version 16
Israël (Tel Aviv)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Mexique (Centre)	Non disponible	Non disponible		
Mexique (Centre)	Non disponible	Non disponible		
Middle East (Bahrain)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Moyen-Orient (EAU)	Versions 17.4 et ultérieures	Non disponible	Versions 16.4 et ultérieures	Non disponible
Amérique du Sud (São Paulo)	Versions 17.4 et ultérieures	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Versions 16.4 et ultérieures
AWS GovCloud (USA Est)	Non disponible	Non disponible	Non disponible	Non disponible
AWS GovCloud (US-Ouest)	Non disponible	Non disponible	Non disponible	Non disponible

Régions et moteurs de base de données Aurora pris en charge pour Proxy Amazon RDS

Le proxy Amazon RDS est un proxy de base de données entièrement géré et hautement disponible qui rend les applications plus évolutives en regroupant et en partageant les connexions de base

de données établies. Pour plus d'informations sur RDS Proxy, consultez Proxy Amazon RDS pour Aurora.

Rubriques

- Amazon RDS Proxy avec Aurora MySQL
- Amazon RDS Proxy avec Aurora PostgreSQL

Amazon RDS Proxy avec Aurora MySQL

Les régions et les versions de moteur suivantes sont disponibles pour RDS Proxy avec Aurora MySQL.

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Est (Virginie du Nord)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
USA Est (Ohio)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
USA Ouest (Californie du Nord)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
USA Ouest (Oregon)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Afrique (Le Cap)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Hong Kong)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Hyderabad)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Jakarta)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
Asie-Pacifique (Malaisie)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Melbourne)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Mumbai)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible	Non disponible
Asie-Pacifique (Osaka)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Séoul)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Singapour)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Sydney)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Taipei)	Non disponible	Non disponible
Asie-Pacifique (Thaïlande)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Asie-Pacifique (Tokyo)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Canada (Centre)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Canada-Ouest (Calgary)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
Chine (Pékin)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Chine (Ningxia)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Europe (Francfort)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Europe (Irlande)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Europe (Londres)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Europe (Milan)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Europe (Paris)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Europe (Espagne)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Europe (Stockholm)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Europe (Zurich)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Israël (Tel Aviv)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Mexique (Centre)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Middle East (Bahrain)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures

Région	Aurora MySQL version 3	Aurora MySQL version 2
Moyen-Orient (EAU)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
Amérique du Sud (São Paulo)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
AWS GovCloud (USA Est)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures
AWS GovCloud (US-Ouest)	Versions 3.01.0 et ultérieures	Version 2.07 et versions 2.11 et ultérieures

Amazon RDS Proxy avec Aurora PostgreSQL

Les régions et les versions de moteur suivantes sont disponibles pour RDS Proxy avec Aurora PostgreSQL.

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
USA Est (Virginie du Nord)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
USA Est (Ohio)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
USA Ouest (Californie du Nord)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
USA Ouest (Oregon)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Afrique (Le Cap)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Hong Kong)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Hyderabad)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Jakarta)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Malaisie)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Melbourne)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Mumbai)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Osaka)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Séoul)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Singapour)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Asie-Pacifique (Sydney)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Taipei)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Thaïlande)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Asie-Pacifique (Tokyo)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Canada (Centre)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Canada-Ouest (Calgary)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Chine (Pékin)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Chine (Ningxia)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Francfort)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europe (Irlande)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Londres)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Milan)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Paris)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Europe (Espagne)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Stockholm)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Europe (Zurich)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Israël (Tel Aviv)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
Mexique (Centre)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Middle East (Bahrain)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Moyen-Orient (EAU)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
Amérique du Sud (São Paulo)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13	Aurora PostgreSQL 12	Aurora PostgreSQL 11
AWS GovCloud (USA Est)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures
AWS GovCloud (US-Ouest)	Versions 17 et ultérieures	Versions 16 et ultérieures	Versions 15 et ultérieures	Versions 14 et ultérieures	Versions 13 et ultérieures	Versions 12 et ultérieures	Version 11.9 et versions 11.13 et ultérieures

Régions et moteurs de base de données Aurora pris en charge pour l'intégration de Secrets Manager

Avec AWS Secrets Manager, vous pouvez remplacer les informations d'identification codées en dur dans votre code, y compris les mots de passe des bases de données, par un appel d'API à Secrets Manager pour récupérer le secret par programme. Pour plus d'informations sur Secrets Manager, consultez le Guide de l'utilisateur AWS Secrets Manager.

Vous pouvez spécifier qu'Amazon Aurora gère le mot de passe de l'utilisateur principal dans Secrets Manager pour un cluster de bases de données Aurora. Aurora génère le mot de passe, le stocke dans Secrets Manager et effectue régulièrement sa rotation. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager.

L'intégration de Secrets Manager est disponible dans toutes les Régions AWS.

Régions et moteurs de base de données Aurora pris en charge pour Aurora Serverless v2

Aurora Serverless v2 est une fonction de scalabilité automatique à la demande conçue pour être une approche rentable de l'exécution de charges de travail intermittentes ou imprévisibles sur Amazon Aurora. Elle augmente et réduit automatiquement la capacité en fonction des besoins de vos applications. La mise à l'échelle est plus rapide et plus granulaire qu'avec Aurora Serverless v1. Avec Aurora Serverless v2, chaque cluster peut contenir une instance de base de données en écriture et plusieurs instances de base de données en lecture. Vous pouvez combiner Aurora Serverless v2 et des instances de base de données allouées de manière traditionnelle au sein d'un même cluster. Pour plus d'informations, consultez Utiliser Aurora Serverless v2.

Rubriques

- Aurora Serverless v2 avec Aurora MySQL
- Aurora Serverless v2 avec Aurora PostgreSQL

Aurora Serverless v2 avec Aurora MySQL

Les régions et les versions de moteur suivantes sont disponibles pour Aurora Serverless v2 avec Aurora MySQL.

Région	Aurora MySQL version 3
USA Est (Virginie du Nord)	Versions 3.02.0 et ultérieures
USA Est (Ohio)	Versions 3.02.0 et ultérieures
USA Ouest (Californie du Nord)	Versions 3.02.0 et ultérieures
USA Ouest (Oregon)	Versions 3.02.0 et ultérieures
Afrique (Le Cap)	Versions 3.02.0 et ultérieures
Asie-Pacifique (Hong Kong)	Versions 3.02.0 et ultérieures
Asie-Pacifique (Hyderabad)	Versions 3.02.3 et ultérieures
Asie-Pacifique (Jakarta)	Versions 3.02.0 et ultérieures

Région	Aurora MySQL version 3
Asie-Pacifique (Malaisie)	Versions 3.04.3, 3.05.2, 3.06.1, 3.07.1 et ultérieures
Asie-Pacifique (Melbourne)	Versions 3.02.3 et ultérieures
Asie-Pacifique (Mumbai)	Versions 3.02.0 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible
Asie-Pacifique (Osaka)	Versions 3.02.0 et ultérieures
Asie-Pacifique (Séoul)	Versions 3.02.0 et ultérieures
Asie-Pacifique (Singapour)	Versions 3.02.0 et ultérieures
Asie-Pacifique (Sydney)	Versions 3.02.0 et ultérieures
Asie-Pacifique (Taipei)	Non disponible
Asie-Pacifique (Thaïlande)	Versions 3.04.3 et supérieures, 3.08.0 et supérieures
Asie-Pacifique (Tokyo)	Versions 3.02.0 et ultérieures
Canada (Centre)	Versions 3.02.0 et ultérieures
Canada-Ouest (Calgary)	Versions 3.04.0 et ultérieures
Chine (Pékin)	Versions 3.02.2 et ultérieures
Chine (Ningxia)	Versions 3.02.2 et ultérieures
Europe (Francfort)	Versions 3.02.0 et ultérieures
Europe (Irlande)	Versions 3.02.0 et ultérieures
Europe (Londres)	Versions 3.02.0 et ultérieures
Europe (Milan)	Versions 3.02.0 et ultérieures

Région	Aurora MySQL version 3
Europe (Paris)	Versions 3.02.0 et ultérieures
Europe (Espagne)	Versions 3.02.3 et ultérieures
Europe (Stockholm)	Versions 3.02.0 et ultérieures
Europe (Zurich)	Versions 3.02.3 et ultérieures
Israël (Tel Aviv)	Versions 3.02.3 et ultérieures, 3.03.1 et ultérieures
Mexique (Centre)	Non disponible
Middle East (Bahrain)	Versions 3.02.0 et ultérieures
Moyen-Orient (EAU)	Versions 3.02.3 et ultérieures
Amérique du Sud (São Paulo)	Versions 3.02.0 et ultérieures
AWS GovCloud (USA Est)	Versions 3.02.2 et ultérieures
AWS GovCloud (US-Ouest)	Versions 3.02.2 et ultérieures

Les limites de capacité supérieure et inférieure de l’ACU pour Aurora Serverless v2 peuvent varier en fonction de la version de votre moteur. Pour en savoir plus, consultez Capacité Aurora Serverless v2.

Aurora Serverless v2 avec Aurora PostgreSQL

Les régions et les versions de moteur suivantes sont disponibles pour Aurora Serverless v2 avec Aurora PostgreSQL.

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
USA Est (Virginie du Nord)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
USA Est (Ohio)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
USA Ouest (Californie du Nord)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
USA Ouest (Oregon)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Afrique (Le Cap)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Asie-Pacifique (Hong Kong)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Asie-Pacifique (Hyderabad)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.6 et ultérieures	Versions 13.9 et ultérieures
Asie-Pacifique (Jakarta)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Asie-Pacifique (Malaisie)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.4 et ultérieures	Versions 14.6, 14.9 et ultérieures	Versions 13.9, 13.12 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asie-Pacifique (Melbourne)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.6 et ultérieures	Versions 13.9 et ultérieures
Asie-Pacifique (Mumbai)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Osaka)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Asie-Pacifique (Séoul)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Asie-Pacifique (Singapour)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Asie-Pacifique (Taipei)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Sydney)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Asie-Pacifique (Thaïlande)	Versions 17.4 et ultérieures	Versions 16.4 et ultérieures	Version 15.8 et supérieure	Version 14.13 et supérieure	Non disponible

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asie-Pacifique (Tokyo)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Canada (Centre)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Canada-Ouest (Calgary)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.6, 14.8 et ultérieures	Versions 13.9, 13.11 et ultérieures
Chine (Pékin)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Chine (Ningxia)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Europe (Francfort)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Europe (Irlande)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Europe (Londres)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Europe (Milan)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Europe (Paris)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Europe (Espagne)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.6 et ultérieures	Versions 13.9 et ultérieures

Région	Aurora PostgreSQL L 17	Aurora PostgreSQL L 16	Aurora PostgreSQL L 15	Aurora PostgreSQL L 14	Aurora PostgreSQL L 13
Europe (Stockholm)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Europe (Zurich)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.6 et ultérieures	Versions 13.9 et ultérieures
Israël (Tel Aviv)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.6 et ultérieures	Versions 13.9 et ultérieures
Mexique (Centre)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Middle East (Bahrain)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
Moyen-Orient (EAU)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.6 et ultérieures	Versions 13.9 et ultérieures
Amérique du Sud (São Paulo)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
AWS GovCloud (USA Est)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures
AWS GovCloud (US-Ouest)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.2 et ultérieures	Versions 14.3 et ultérieures	Versions 13.6 et ultérieures

Les limites de capacité supérieure et inférieure de l’ACU pour Aurora Serverless v2 peuvent varier en fonction de la version de votre moteur. Pour en savoir plus, consultez Capacité Aurora Serverless v2.

Aurora Serverless v1

> **Important**
>
> AWS a annoncé la end-of-life date du Aurora Serverless v1 : 31 mars 2025. Nous recommandons vivement de mettre à niveau tous les clusters de bases de données Aurora Serverless v1 vers Aurora Serverless v2 avant cette date. La mise à niveau peut impliquer une modification du numéro de version majeure du moteur de base de données. Il est donc important de planifier, de tester et de mettre en œuvre cette transition avant la end-of-life date prévue. À compter du 8 janvier 2025, les clients ne seront plus en mesure de créer de nouveaux Aurora Serverless v1 clusters ou instances avec la CLI AWS Management Console ou la CLI. Pour plus d'informations sur le processus de migration, consultez Mise à niveau d'un cluster Aurora Serverless v1 vers Aurora Serverless v2.
>
> Aurora Serverless v2 se met à l'échelle plus rapidement et de manière plus granulaire. Aurora Serverless v2 offre également une meilleure compatibilité avec d'autres fonctionnalités d'Aurora, telles que les instances de base de données de lecteur. Pour en savoir plus sur Aurora Serverless v2, consultez Utiliser Aurora Serverless v2.

Aurora Serverless v1 est une fonction de scalabilité automatique à la demande conçue pour être une approche rentable de l'exécution de charges de travail intermittentes ou imprévisibles sur Amazon Aurora. Elle démarre, s'arrête et augmente ou met à l'échelle la capacité en fonction des besoins de votre applications, en utilisant une seule instance de base de données dans chaque cluster. Pour plus d'informations, consultez Utilisation d'Amazon Aurora Serverless v1.

Rubriques

- Aurora Serverless v1 avec Aurora MySQL
- Aurora Serverless v1 avec Aurora PostgreSQL

Aurora Serverless v1 avec Aurora MySQL

Les régions et les versions de moteur suivantes sont disponibles pour Aurora Serverless v1 avec Aurora MySQL.

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Est (Virginie du Nord)	Non disponible	Version 2.11.4
USA Est (Ohio)	Non disponible	Version 2.11.4
USA Ouest (Californie du Nord)	Non disponible	Version 2.11.4
USA Ouest (Oregon)	Non disponible	Version 2.11.4
Afrique (Le Cap)	Non disponible	Non disponible
Asie-Pacifique (Hong Kong)	Non disponible	Non disponible
Asie-Pacifique (Hyderabad)	Non disponible	Non disponible
Asie-Pacifique (Jakarta)	Non disponible	Non disponible
Asie-Pacifique (Malaisie)	Non disponible	Non disponible
Asie-Pacifique (Melbourne)	Non disponible	Non disponible
Asie-Pacifique (Mumbai)	Non disponible	Version 2.11.4
Asie-Pacifique (Osaka)	Non disponible	Non disponible
Asie-Pacifique (Séoul)	Non disponible	Version 2.11.4
Asie-Pacifique (Singapour)	Non disponible	Version 2.11.4
Asie-Pacifique (Sydney)	Non disponible	Version 2.11.4
Asie-Pacifique (Thaïlande)	Non disponible	Non disponible
Asie-Pacifique (Tokyo)	Non disponible	Version 2.11.4
Canada (Centre)	Non disponible	Version 2.11.4
Canada-Ouest (Calgary)	Non disponible	Non disponible
Chine (Pékin)	Non disponible	Non disponible

Région	Aurora MySQL version 3	Aurora MySQL version 2
Chine (Ningxia)	Non disponible	Version 2.11.4
Europe (Francfort)	Non disponible	Version 2.11.4
Europe (Irlande)	Non disponible	Version 2.11.4
Europe (Londres)	Non disponible	Version 2.11.4
Europe (Milan)	Non disponible	Non disponible
Europe (Paris)	Non disponible	Version 2.11.4
Europe (Espagne)	Non disponible	Non disponible
Europe (Stockholm)	Non disponible	Non disponible
Europe (Zurich)	Non disponible	Non disponible
Israël (Tel Aviv)	Non disponible	Non disponible
Middle East (Bahrain)	Non disponible	Non disponible
Moyen-Orient (EAU)	Non disponible	Non disponible
Amérique du Sud (São Paulo)	Non disponible	Non disponible
AWS GovCloud (USA Est)	Non disponible	Non disponible
AWS GovCloud (US-Ouest)	Non disponible	Non disponible

Aurora Serverless v1 avec Aurora PostgreSQL

Les régions et les versions de moteur suivantes sont disponibles pour Aurora Serverless v1 avec Aurora PostgreSQL.

Région	Aurora PostgreSQL 13
USA Est (Virginie du Nord)	Version 13.12

Région	Aurora PostgreSQL 13
USA Est (Ohio)	Version 13.12
USA Ouest (Californie du Nord)	Version 13.12
USA Ouest (Oregon)	Version 13.12
Afrique (Le Cap)	Non disponible
Asie-Pacifique (Hong Kong)	Non disponible
Asie-Pacifique (Hyderabad)	Non disponible
Asie-Pacifique (Jakarta)	Non disponible
Asie-Pacifique (Malaisie)	Non disponible
Asie-Pacifique (Melbourne)	Non disponible
Asie-Pacifique (Mumbai)	Version 13.12
Asie-Pacifique (Osaka)	Non disponible
Asie-Pacifique (Séoul)	Version 13.12
Asie-Pacifique (Singapour)	Version 13.12
Asie-Pacifique (Sydney)	Version 13.12
Asie-Pacifique (Thaïlande)	Non disponible
Asie-Pacifique (Tokyo)	Version 13.12
Canada (Centre)	Version 13.12
Canada-Ouest (Calgary)	Non disponible
Chine (Pékin)	Non disponible
Chine (Ningxia)	Non disponible

Région	Aurora PostgreSQL 13
Europe (Francfort)	Version 13.12
Europe (Irlande)	Version 13.12
Europe (Londres)	Version 13.12
Europe (Milan)	Non disponible
Europe (Paris)	Version 13.12
Europe (Espagne)	Non disponible
Europe (Stockholm)	Non disponible
Europe (Zurich)	Non disponible
Israël (Tel Aviv)	Non disponible
Middle East (Bahrain)	Non disponible
Moyen-Orient (EAU)	Non disponible
Amérique du Sud (São Paulo)	Non disponible
AWS GovCloud (USA Est)	Non disponible
AWS GovCloud (US-Ouest)	Non disponible

Régions et moteurs de base de données Aurora pris en charge pour l'API de données RDS

L'API de données RDS (API de données) fournit une interface de services Web à un cluster de bases de données Amazon Aurora. Plutôt que de gérer les connexions de base de données à partir d'applications de client, vous pouvez exécuter des commandes SQL sur un point de terminaison HTTPS. Pour plus d'informations, consultez Utilisation de l'API de données Amazon RDS.

Rubriques

- API de données avec Aurora PostgreSQL sans serveur v2 et provisionné

- API de données avec Aurora MySQL sans serveur v2 et provisionné
- API de données avec Aurora PostgreSQL sans serveur v1
- API de données avec Aurora MySQL sans serveur v1

API de données avec Aurora PostgreSQL sans serveur v2 et provisionné

Les régions et les versions de moteur suivantes sont disponibles pour l'API de données avec Aurora PostgreSQL sans serveur v2 et les clusters de bases de données provisionnés.

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
USA Est (Virginie du Nord)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
USA Est (Ohio)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
USA Ouest (Californie du Nord)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
USA Ouest (Oregon)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Afrique (Le Cap)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Asie-Pacifique (Hong Kong)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Asie-Pacifique (Hyderabad)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asie-Pacifique (Jakarta)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Asie-Pacifique (Malaisie)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Asie-Pacifique (Melbourne)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Mumbai)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Osaka)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Asie-Pacifique (Séoul)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Asie-Pacifique (Singapour)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Asie-Pacifique (Sydney)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Asie-Pacifique (Taipei)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Asie-Pacifique (Thaïlande)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Asie-Pacifique (Tokyo)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Canada (Centre)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Canada-Ouest (Calgary)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Chine (Pékin)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Chine (Ningxia)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Europe (Francfort)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Europe (Irlande)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Europe (Londres)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Europe (Milan)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures

Région	Aurora PostgreSQL 17	Aurora PostgreSQL 16	Aurora PostgreSQL 15	Aurora PostgreSQL 14	Aurora PostgreSQL 13
Europe (Paris)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Europe (Espagne)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Europe (Stockholm)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Europe (Zurich)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Israël (Tel Aviv)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Middle East (Bahrain)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Mexique (Centre)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
Moyen-Orient (EAU)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
Amérique du Sud (São Paulo)	Versions 17.4 et ultérieures	Versions 16.1 et ultérieures	Versions 15.3 et ultérieures	Versions 14.8 et ultérieures	Versions 13.11 et ultérieures
AWS GovCloud (USA Est)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible
AWS GovCloud (US-Ouest)	Non disponible	Non disponible	Non disponible	Non disponible	Non disponible

API de données avec Aurora MySQL sans serveur v2 et provisionné

Les régions et les versions de moteur suivantes sont disponibles pour l'API de données avec Aurora MySQL sans serveur v2 et les clusters de bases de données provisionnés.

Région	Aurora MySQL version 3
USA Est (Ohio)	Versions 3.07 et ultérieures
USA Est (Virginie du Nord)	Versions 3.07 et ultérieures
USA Ouest (Californie du Nord)	Versions 3.07 et ultérieures
USA Ouest (Oregon)	Versions 3.07 et ultérieures
Afrique (Le Cap)	Versions 3.07 et ultérieures
Asie-Pacifique (Hong Kong)	Versions 3.07 et ultérieures
Asie-Pacifique (Hyderabad)	Non disponible
Asie-Pacifique (Jakarta)	Versions 3.07 et ultérieures
Asie-Pacifique (Malaisie)	Versions 3.07 et ultérieures
Asie-Pacifique (Melbourne)	Non disponible
Asie-Pacifique (Mumbai)	Versions 3.07 et ultérieures
Asie-Pacifique (Nouvelle-Zélande)	Non disponible
Asie-Pacifique (Osaka)	Versions 3.07 et ultérieures
Asie-Pacifique (Séoul)	Versions 3.07 et ultérieures
Asie-Pacifique (Singapour)	Versions 3.07 et ultérieures
Asie-Pacifique (Sydney)	Versions 3.07 et ultérieures
Asie-Pacifique (Taipei)	Non disponible
Asie-Pacifique (Thaïlande)	Versions 3.07 et ultérieures

Région	Aurora MySQL version 3
Asie-Pacifique (Tokyo)	Versions 3.07 et ultérieures
Canada (Centre)	Versions 3.07 et ultérieures
Canada-Ouest (Calgary)	Non disponible
Chine (Pékin)	Non disponible
Chine (Ningxia)	Non disponible
Europe (Francfort)	Versions 3.07 et ultérieures
Europe (Irlande)	Versions 3.07 et ultérieures
Europe (Londres)	Versions 3.07 et ultérieures
Europe (Milan)	Versions 3.07 et ultérieures
Europe (Paris)	Versions 3.07 et ultérieures
Europe (Espagne)	Versions 3.07 et ultérieures
Europe (Stockholm)	Versions 3.07 et ultérieures
Europe (Zurich)	Non disponible
Israël (Tel Aviv)	Non disponible
Mexique (Centre)	Non disponible
Middle East (Bahrain)	Versions 3.07 et ultérieures
Moyen-Orient (EAU)	Versions 3.07 et ultérieures
Amérique du Sud (São Paulo)	Versions 3.07 et ultérieures
AWS GovCloud (USA Est)	Non disponible
AWS GovCloud (US-Ouest)	Non disponible

API de données avec Aurora PostgreSQL sans serveur v1

Les régions et les versions de moteur suivantes sont disponibles pour l'API de données avec Aurora PostgreSQL sans serveur v1.

Région	Aurora PostgreSQL 13	Aurora PostgreSQL 11
USA Est (Virginie du Nord)	Version 13.9	Version 11.18
USA Est (Ohio)	Version 13.9	Version 11.18
USA Ouest (Californie du Nord)	Version 13.9	Version 11.18
USA Ouest (Oregon)	Version 13.9	Version 11.18
Afrique (Le Cap)	Non disponible	Non disponible
Asie-Pacifique (Hong Kong)	Non disponible	Non disponible
Asie-Pacifique (Hyderabad)	Non disponible	Non disponible
Asie-Pacifique (Jakarta)	Non disponible	Non disponible
Asie-Pacifique (Malaisie)	Non disponible	Non disponible
Asie-Pacifique (Melbourne)	Non disponible	Non disponible
Asie-Pacifique (Mumbai)	Version 13.9	Version 11.18
Asie-Pacifique (Osaka)	Non disponible	Non disponible
Asie-Pacifique (Séoul)	Version 13.9	Version 11.18
Asie-Pacifique (Singapour)	Version 13.9	Version 11.18
Asie-Pacifique (Sydney)	Version 13.9	Version 11.18
Asie-Pacifique (Thaïlande)	Non disponible	Non disponible
Asie-Pacifique (Tokyo)	Version 13.9	Version 11.18

Région	Aurora PostgreSQL 13	Aurora PostgreSQL 11
Canada (Centre)	Version 13.9	Version 11.18
Chine (Pékin)	Non disponible	Non disponible
Chine (Ningxia)	Non disponible	Non disponible
Europe (Francfort)	Version 13.9	Version 11.18
Europe (Irlande)	Version 13.9	Version 11.18
Europe (Londres)	Version 13.9	Version 11.18
Europe (Milan)	Non disponible	Non disponible
Europe (Paris)	Version 13.9	Version 11.18
Europe (Espagne)	Version 13.9	Version 11.18
Europe (Stockholm)	Non disponible	Non disponible
Europe (Zurich)	Non disponible	Non disponible
Israël (Tel Aviv)	Non disponible	Non disponible
Middle East (Bahrain)	Non disponible	Non disponible
Moyen-Orient (EAU)	Non disponible	Non disponible
Amérique du Sud (São Paulo)	Non disponible	Non disponible
AWS GovCloud (USA Est)	Non disponible	Non disponible
AWS GovCloud (US-Ouest)	Non disponible	Non disponible

API de données avec Aurora MySQL sans serveur v1

Les régions et les versions de moteur suivantes sont disponibles pour l’API de données avec Aurora MySQL sans serveur v1.

Région	Aurora MySQL version 2
USA Est (Virginie du Nord)	Version 2.11.3
USA Est (Ohio)	Version 2.11.3
USA Ouest (Californie du Nord)	Version 2.11.3
USA Ouest (Oregon)	Version 2.11.3
Afrique (Le Cap)	Non disponible
Asie-Pacifique (Hong Kong)	Non disponible
Asie-Pacifique (Hyderabad)	Non disponible
Asie-Pacifique (Jakarta)	Non disponible
Asie-Pacifique (Malaisie)	Non disponible
Asie-Pacifique (Melbourne)	Non disponible
Asie-Pacifique (Mumbai)	Version 2.11.3
Asie-Pacifique (Osaka)	Non disponible
Asie-Pacifique (Séoul)	Version 2.11.3
Asie-Pacifique (Singapour)	Version 2.11.3
Asie-Pacifique (Sydney)	Version 2.11.3
Asie-Pacifique (Thaïlande)	Non disponible
Asie-Pacifique (Tokyo)	Version 2.11.3
Canada (Centre)	Version 2.11.3
Canada-Ouest (Calgary)	Non disponible
Chine (Pékin)	Non disponible

Région	Aurora MySQL version 2
Chine (Ningxia)	Version 2.11.3
Europe (Francfort)	Version 2.11.3
Europe (Irlande)	Version 2.11.3
Europe (Londres)	Version 2.11.3
Europe (Milan)	Non disponible
Europe (Paris)	Version 2.11.3
Europe (Espagne)	Version 2.11.3
Europe (Stockholm)	Non disponible
Europe (Zurich)	Non disponible
Israël (Tel Aviv)	Non disponible
Middle East (Bahrain)	Non disponible
Moyen-Orient (EAU)	Non disponible
Amérique du Sud (São Paulo)	Non disponible
AWS GovCloud (USA Est)	Non disponible
AWS GovCloud (US-Ouest)	Non disponible

Régions et moteurs de base de données Aurora pris en charge pour l’application de correctifs sans durée d’indisponibilité

L’exécution de mises à niveau pour les clusters de bases de données Aurora implique la possibilité d’une panne lorsque la base de données est arrêtée et pendant sa mise à niveau. Par défaut, si vous démarrez la mise à niveau alors que la base de données est occupée, vous perdez toutes les connexions et transactions traitées par le cluster de bases de données. Si vous attendez que la base de données soit inactive pour effectuer la mise à niveau, vous devrez peut-être attendre longtemps.

La fonctionnalité d'application de correctifs sans interruption de service (ZDP) tente, dans un souci d'optimisation, de conserver les connexions client tout au long de la mise à niveau d'Aurora. Si l'application de correctifs sans durée d'indisponibilité s'exécute correctement, les sessions d'application sont conservées et le moteur de base de données redémarre pendant que la mise à niveau est en cours. Le redémarrage du moteur de base de données peut entraîner une chute du débit qui dure de quelques secondes à environ une minute.

Pour des informations détaillées sur les conditions et les versions du moteur dans lesquelles l'application de correctifs sans interruption de service est disponible pour les mises à niveau d'Aurora MySQL, consultez Utilisation des correctifs sans durée d'indisponibilité.

Pour des informations détaillées sur les conditions et les versions du moteur dans lesquelles l'application de correctifs sans interruption de service est disponible pour les mises à niveau d'Aurora PostgreSQL, consultez Mises à niveau de versions mineures et application de correctifs sans durée d'indisponibilité.

Régions prises en charge pour la base de données Aurora PostgreSQL Limitless

La base de données Amazon Aurora PostgreSQL Limitless assure une mise à l'échelle horizontale automatisée pour traiter des millions de transactions d'écriture par seconde et gère des pétaoctets de données tout en préservant la simplicité offerte par l'utilisation d'une seule et même base de données. Avec la base de données Aurora PostgreSQL Limitless, vous pouvez vous concentrer sur la création d'applications à grande échelle sans avoir à créer ni à gérer de solutions complexes pour mettre à l'échelle vos données entre plusieurs instances de base de données afin de répondre aux besoins de vos charges de travail.

Pour plus d'informations, consultez Utilisation d'Amazon Aurora PostgreSQL Limitless Database.

La base de données Aurora PostgreSQL Limitless est disponible dans toutes les Régions AWS, sauf dans la région Asie-Pacifique (Taipei).

Régions et moteurs de base de données pris en charge pour les fonctionnalités natives du moteur Aurora

Les moteurs de base de données Aurora prennent également en charge des fonctions et des fonctionnalités supplémentaires spécifiques à Aurora. Certaines fonctions natives du moteur peuvent avoir une prise en charge limitée ou des privilèges restreints pour un moteur de base de données Aurora, une version ou une région en particulier.

Rubriques

- Fonctions natives du moteur pour Aurora MySQL
- Fonctions natives du moteur pour Aurora PostgreSQL

Fonctions natives du moteur pour Aurora MySQL

Voici les fonctions natives du moteur pour Aurora MySQL.

- Audit avancé
- Retour sur trace
- Demandes d'injection d'erreurs
- Transfert d'écriture intracluster
- Requête parallèle

Fonctions natives du moteur pour Aurora PostgreSQL

Voici les fonctions natives du moteur pour Aurora PostgreSQL.

- Babelfish
- Demandes d'injection d'erreurs
- Gestion de plans de requêtes

Connexions de point de terminaison Amazon Aurora

Amazon Aurora implique généralement un cluster d'instances de base de données au lieu d'une seule instance. Chaque connexion est gérée par une instance de base de données spécifique. Lorsque vous vous connectez à un cluster Aurora, le nom d'hôte et le port que vous spécifiez pointent vers un gestionnaire intermédiaire appelé point de terminaison. Aurora utilise le mécanisme de point de terminaison pour abstraire ces connexions. De cette manière, vous n'avez pas à coder en dur tous les noms d'hôte ni à écrire votre propre logique pour l'équilibrage et le reroutage des connexions en cas d'indisponibilité de certaines instances de base de données.

Pour certaines tâches Aurora, différentes instances ou différents groupes d'instances exécutent des rôles distincts. Par exemple, l'instance principale gère toutes les instructions de langage de manipulation de données (DDL) et de langage de définition de données (DML). Jusqu'à 15 réplicas Aurora gèrent le trafic des requêtes en lecture seule.

Rubriques

Types de points de terminaison Aurora

Les points de terminaison permettent d'associer chaque connexion à l'instance ou au groupe d'instances approprié en fonction de votre cas d'utilisation. Par exemple, pour exécuter des instructions DDL, vous pouvez vous connecter à n'importe quelle instance principale. Pour exécuter des requêtes, vous pouvez vous connecter au point de terminaison dz lecteur, tandis qu'Aurora effectue automatiquement l'équilibrage des connexions entre tous les réplicas Aurora. Pour les clusters dotés d'instances de base de données avec des capacités ou des configurations distinctes, vous pouvez vous connecter à des points de terminaison personnalisés associés à différents sous-ensembles d'instances de base de données. Pour le diagnostic et le réglage, vous pouvez vous connecter au point de terminaison d'une instance spécifique pour examiner les détails relatifs à cette dernière.

Un point de terminaison est une URL spécifique à Aurora, qui contient une adresse d'hôte et un port. Les types de points de terminaison suivants sont disponibles à partir d'un cluster de bases de données Aurora.

Point de terminaison de cluster

Connectez-vous à l'instance principale de votre cluster pour développer et tester des applications, et pour effectuer des transformations telles que des instructions `INSERT` ainsi que des opérations DDL, DML et ETL. Trouvez l'emplacement du point de terminaison de cluster à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API Amazon RDS, comme décrit dans Affichage des points de terminaison d'un cluster Aurora.

Pour plus d'informations sur les points de terminaison de cluster, consultez Points de terminaison de cluster pour Amazon Aurora.

Point de terminaison du lecteur

Exécutez des requêtes. Aurora effectue automatiquement l'équilibrage des connexions entre tous les réplicas Aurora. Trouvez l'emplacement du point de terminaison de lecteur à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API Amazon RDS, comme décrit dans Affichage des points de terminaison d'un cluster Aurora.

Pour plus d'informations sur les points de terminaison de lecteur, consultez Points de terminaison de lecteur pour Amazon Aurora.

Point de terminaison d'instance

Examinez les détails d'une instance de base de données spécifique à des fins de diagnostic ou d'affinement. Vous ne trouverez l'emplacement du point de terminaison de chacune de vos instances que dans la AWS Management Console, sur la page détaillée de votre instance.

Pour plus d'informations sur les points de terminaison d'instance, consultez Points de terminaison d'instance pour Amazon Aurora.

Point de terminaison personnalisé

Connectez-vous à différents sous-ensembles d'instances de base de données sur le cluster de bases de données. Cela est utile lorsque vous disposez de différentes capacités et configurations d'instance au sein de votre cluster de bases de données. Trouvez les emplacements de point de terminaison personnalisé à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API Amazon RDS, comme décrit dans Affichage des points de terminaison d'un cluster Aurora.

Pour plus d'informations sur les points de terminaison personnalisés, consultez Points de terminaison personnalisés pour Amazon Aurora.

Point de terminaison d'enregistreur Aurora Global Database

Aurora Global Database inclut un type spécial de point de terminaison qui a le même objectif que le point de terminaison d'un cluster Aurora autonome. Il gère à la fois les demandes d'écriture et de lecture. Lorsqu'un cluster secondaire devient le nouveau cluster principal à la suite d'une opération de bascule ou de basculement, Aurora fait automatiquement pointer ce point de terminaison vers le point de terminaison du nouveau cluster principal, dans l'autre Région AWS. Ainsi, vous n'avez pas à encoder la région AWS dans la chaîne de connexion de votre application, ni à modifier la chaîne de connexion lorsque la structure de la base de données globale change. Aurora crée ce point de terminaison lorsque vous configurez une base de données Aurora Global Database, par exemple en choisissant Ajouter une région pour un cluster Aurora dans la AWS Management Console.

Pour en savoir plus sur la manière dont vous pouvez utiliser ce type de point de terminaison avec Aurora Global Database, consultez Connexion à Amazon Aurora Global Database.

Affichage des points de terminaison d'un cluster Aurora

Bien que vous ne puissiez trouver l'emplacement du point de terminaison d'instance que sur la page détaillée de l'instance dans la AWS Management Console, vous pouvez utiliser la console, l'AWS CLI ou l'API Amazon RDS pour trouver l'emplacement du point de terminaison du cluster, du points de terminaison du lecteur et des points de terminaison personnalisés.

Console

Dans la AWS Management Console, vous trouverez le point de terminaison du cluster, le point de terminaison du lecteur et les points de terminaison personnalisés sur la page des détails de l'instance pour votre cluster. La page de détails de chaque instance affiche le point de terminaison de cette dernière. Lorsque vous vous connectez, ajoutez le numéro de port associé, après un point-virgule, au nom du point de terminaison indiqué sur cette page de détails.

AWS CLI

Avec l'AWS CLI, vous trouverez les points de terminaison de l'enregistreur et du lecteur et les points de terminaison personnalisés dans la sortie de la commande describe-db-clusters. Par exemple, la commande suivante affiche les attributs de point de terminaison pour tous les clusters figurant dans votre région AWS actuelle.

```
aws rds describe-db-clusters --query '*[].
{Endpoint:Endpoint,ReaderEndpoint:ReaderEndpoint,CustomEndpoints:CustomEndpoints}'
```

Amazon RDS API

L'API Amazon RDS vous permet de récupérer les points de terminaison en appelant l'opération DescribeDBClusterEndpoints.

Les points de terminaison Aurora et la haute disponibilité

Pour les clusters où la haute disponibilité est importante, utilisez le point de terminaison du cluster pour les connexions en lecture/écriture ou les connexions à usage général et le point de terminaison du lecteur pour les connexions en lecture seule. Les points de terminaison de l'enregistreur et du

lecteur gèrent le basculement d'instance de base de données mieux que ne le font les points de terminaison d'instance. Contrairement aux points de terminaison d'instance, les points de terminaison de l'enregistreur et du lecteur modifient automatiquement l'instance de base de données à laquelle ils se connectent si une instance de base de données de votre cluster devient indisponible. Pour plus d'informations sur les points de terminaison de cluster et de lecteur, consultez Points de terminaison de cluster pour Amazon Aurora et Points de terminaison de lecteur pour Amazon Aurora.

En cas de défaillance de l'instance de base de données principale d'un cluster DB, Aurora bascule automatiquement vers une nouvelle instance de base de données principale. Pour ce faire, il promeut un réplica Aurora existant en tant que nouvelle instance de base de données principale ou il crée une instance de base de données principale. En cas de basculement, vous pouvez utiliser le point de terminaison de cluster pour vous connecter à l'instance principale qui vient d'être promue ou créée, ou le point de terminaison de lecteur pour vous reconnecter à l'un des réplicas Aurora du cluster de bases de données. Pendant un basculement, le point de terminaison de lecteur peut brièvement diriger les connexions vers la nouvelle instance de base de données principale d'un cluster de bases de données, après qu'un réplica Aurora est promu comme nouvelle instance de base de données principale.

Si vous concevez votre propre logique applicative pour gérer les connexions aux points de terminaison d'instance, vous pouvez détecter manuellement ou par programmation l'ensemble obtenu d'instances de base de données disponibles dans le cluster de bases de données. Utilisez la commande describe-db-clusters AWS CLI ou l'opération DescribeDBClusters de l'API RDS pour trouver les points de terminaison des clusters de bases de données et des lecteurs, ainsi que les instances de base de données et pour déterminer si les instances de base de données sont des lecteurs, ainsi que leurs niveaux de promotion. Vous pouvez ensuite confirmer leurs classes d'instance après le basculement et vous connecter à un point de terminaison d'instance approprié.

Pour plus d'informations sur les basculements, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.

Pour plus d'informations sur la haute disponibilité dans Amazon Aurora, consultez Haute disponibilité pour Amazon Aurora.

Points de terminaison de cluster pour Amazon Aurora

Un point de terminaison de cluster (ou point de terminaison d'enregistreur) pour un cluster de bases de données Aurora se connecte à l'instance de base de données principale de ce cluster de bases de données. Ce point de terminaison est le seul à pouvoir exécuter des opérations d'écriture, telles

que des instructions DDL. C'est pour cette raison que le point de terminaison du cluster est celui auquel vous vous connectez la première fois que vous configurez un cluster ou lorsque le cluster contient une seule instance de base de données.

Chaque cluster de bases de données Aurora possède un seul point de terminaison et une seule instance de base de données principale.

Le point de terminaison du cluster est destiné à toutes les opérations d'écriture sur le cluster de bases de données, y compris les insertions, les mises à jour, les suppression et les modifications de langage de définition de données (DDL). Vous pouvez aussi utiliser le point de terminaison de cluster pour les opérations de lecture, par exemple les requêtes.

Le point de terminaison de cluster assure la prise en charge du basculement pour les connexions en lecture/écriture au cluster de bases de données. En cas de défaillance de l'instance de base de données principale en cours d'un cluster de bases de données, Aurora bascule automatiquement vers une nouvelle instance de base de données principale. Pendant un basculement, le cluster DB continue à traiter les demandes de connexion adressées au point de terminaison de cluster par la nouvelle instance DB principale, avec une interruption de service minimale.

L'exemple suivant illustre un point de terminaison de cluster pour un cluster de bases de données Aurora MySQL.

```
mydbcluster.cluster-c7tj4example.us-east-1.rds.amazonaws.com:3306
```

Chaque cluster Aurora intègre un seul point de terminaison de cluster dont le nom et les autres attributs sont gérés par Aurora. Vous ne pouvez pas créer, supprimer ni modifier ce type de point de terminaison.

Le point de terminaison du cluster vous permet d'administrer ce dernier, d'exécuter des opération ETL ou de développer et de tester des applications. Le point de terminaison du cluster vous connecte à l'instance principale du cluster. L'instance principale est la seule instance de base de données où vous pouvez créer des tables et des index, exécuter des instructions `INSERT` et effectuer d'autres opérations DDL et DML.

L'adresse IP physique à laquelle renvoie le point de terminaison du cluster change lorsque le mécanisme de basculement choisit une nouvelle instance de base de données comme instance principale du cluster en lecture/écriture. Si vous utilisez toute forme de groupes de connexions ou de multiplexage quelconque, préparez-vous à réduire la durée de vie (TTL, time-to-live) des informations DNS mises en cache. Cette technique vous empêche d'essayer d'établir une connexion en lecture/

écriture à une instance de base de données qui est devenue indisponible ou qui n'est plus qu'en lecture seule après un basculement.

Points de terminaison de lecteur pour Amazon Aurora

Un point de terminaison de lecteur pour un cluster de bases de données Aurora prend en charge l'équilibrage des connexions en lecture seule au cluster de bases de données. Utilisez le point de terminaison de lecteur pour les opérations de lecture, par exemple les requêtes. En traitant ces instructions sur les réplicas Aurora en lecture seule, ce point de terminaison réduit la surcharge sur l'instance principale. Il aide également le cluster à mettre à l'échelle la capacité de traiter des requêtes `SELECT` simultanées, proportionnelle au nombre de réplicas Aurora dans le cluster. Chaque cluster de bases de données Aurora possède un seul point de terminaison de lecteur.

Si le cluster contient un ou plusieurs réplicas Aurora, le point de terminaison du lecteur équilibre chaque demande de connexion entre les réplicas Aurora. Dans ce cas, vous ne pouvez effectuer que des instructions en lecture seule, telles que `SELECT` dans cette session. Si le cluster contient uniquement une instance principale et aucun réplica Aurora, le point de terminaison du lecteur se connecte à l'instance principale. Dans ce cas, vous pouvez effectuer des opérations d'écriture via le point de terminaison.

L'exemple suivant illustre un point de terminaison de lecteur pour un cluster de bases de données Aurora MySQL.

```
mydbcluster.cluster-ro-c7tj4example.us-east-1.rds.amazonaws.com:3306
```

Le point de terminaison du lecteur est destiné aux connexions en lecture seule au cluster Aurora. Ce point de terminaison utilise un mécanisme d'équilibrage des connexions pour aider le cluster à gérer les charges de travail impliquant beaucoup de requêtes. Le point de terminaison du lecteur est le point de terminaison que vous fournissez aux applications qui créent les rapports ou qui effectuent d'autres opérations en lecture seule sur le cluster.

Le point de terminaison de lecteur équilibre les connexions aux réplicas Aurora disponibles dans un cluster de bases de données Aurora. Il n'équilibre pas des requêtes individuelles. Si vous souhaitez équilibrer chaque requête afin de répartir la charge de travail de lecture d'un cluster de bases de données, ouvrez une nouvelle connexion au point de terminaison de lecteur pour chaque requête.

Chaque cluster Aurora intègre un seul point de terminaison de lecteur dont le nom et les autres attributs sont gérés par Aurora. Vous ne pouvez pas créer, supprimer ni modifier ce type de point de terminaison.

Si votre cluster contient uniquement une cible principale (instance ou groupe de partitions de base de données) et aucun réplica Aurora, le point de terminaison de lecteur se connecte à l'instance principale. Dans ce cas, vous pouvez effectuer des opérations d'écriture via ce point de terminaison.

> **Tip**
>
> Grâce à RDS Proxy, vous pouvez créer des points de terminaison supplémentaires en lecture seule pour un cluster Aurora. Ces points de terminaison effectuent le même type d'équilibrage des connexions que le point de terminaison de lecteur Aurora. Les applications peuvent se reconnecter plus rapidement aux points de terminaison proxy que le point de terminaison du lecteur Aurora si les instances de lecteur deviennent indisponibles. Les points de terminaison proxy peuvent également tirer parti d'autres fonctions de proxy telles que le multiplexage. Pour plus d'informations, consultez Utilisation des points de terminaison de lecteur avec les clusters Aurora.

Points de terminaison d'instance pour Amazon Aurora

Un point de terminaison d'instance se connecte à une instance de base de données spécifique dans un cluster Aurora. Chaque instance de base de données d'un cluster de bases de données a son propre point de terminaison d'instance unique. Par conséquent, il existe un point de terminaison d'instance pour l'instance principale du cluster de bases de données, et un point de terminaison d'instance pour chacun des réplicas Aurora de ce cluster.

Le point de terminaison d'instance exerce un contrôle direct sur les connexions au cluster de bases de données, pour les scénarios où l'utilisation du point de terminaison de cluster ou du point de terminaison de lecteur peut ne pas être appropriée. Par exemple, votre application client peut exiger un équilibrage plus précis des connexions en fonction du type de charge de travail. Dans ce cas, vous pouvez configurer plusieurs clients afin d'obtenir une connexion à différents réplicas Aurora dans un cluster de bases de données pour répartir les charges de travail en lecture. Pour voir un exemple d'utilisation des points de terminaison d'instance pour améliorer la vitesse de connexion après un basculement pour Aurora PostgreSQL, consultez Basculement rapide avec Amazon Aurora PostgreSQL. Pour voir un exemple d'utilisation des points de terminaison d'instance permettant d'améliorer la vitesse de connexion après un basculement pour Aurora MySQL, consultez Prise en charge du basculement vers MariaDB Connector/J – étude de case Amazon Aurora.

L'exemple suivant illustre un point de terminaison d'instance pour une instance de base de données d'un cluster de bases de données Aurora MySQL.

```
mydbinstance.c7tj4example.us-east-1.rds.amazonaws.com:3306
```

Chaque instance de base de données d'un cluster Aurora a son propre point de terminaison d'instance intégré dont le nom et les autres attributs sont gérés par Aurora. Vous ne pouvez pas créer, supprimer ni modifier ce type de point de terminaison. Il est possible que vous soyez familier avec les points de terminaison d'instance si vous utilisez Amazon RDS. Cependant, avec Aurora vous utilisez généralement les points de terminaison de l'enregistreur et du lecteur plus souvent que les points de terminaison d'instance.

Dans les opérations Aurora au quotidien, vous utilisez principalement les points de terminaison d'instance pour diagnostiquer les problèmes de capacité ou de performances qui affectent une instance spécifique dans un cluster Aurora. Lorsque vous êtes connecté à une instance particulière, vous pouvez examiner ses variables d'état, ses métriques, etc. Cette approche vous permet de déterminer en quoi le comportement de cette instance diffère de celui des autres instances du cluster.

Dans certains cas d'utilisation avancés, vous pouvez configurer certaines instances de base de données différemment des autres. Dans cette situation, utilisez le point de terminaison d'instance pour vous connecter directement à une instance qui est plus petite, qui est plus grande ou qui présente des caractéristiques distinctes de celles des autres. Configurez également la priorité de basculement pour que cette instance de base de données spécifique soit la dernière choisie comme instance principale. Dans ces cas de figure, nous vous recommandons d'utiliser des points de terminaison personnalisés plutôt que des points de terminaison d'instance. Cette approche permet de simplifier la gestion des connexions et la haute disponibilité lorsque vous ajoutez d'autres instances de base de données au cluster.

Points de terminaison personnalisés pour Amazon Aurora

Un point de terminaison pour un cluster Aurora représente un ensemble d'instances de base de données que vous choisissez. Lorsque vous vous connectez au point de terminaison, Aurora effectue l'équilibrage des connexions et choisit l'une des instances dans le groupe pour qu'elle gère la connexion. C'est vous qui définissez les instances auxquelles ce point de terminaison renvoie, ainsi que la fonction même de chaque point de terminaison.

Un cluster de bases de données Aurora n'a pas de point de terminaison personnalisé tant que vous n'en créez pas un. Vous pouvez créer jusqu'à cinq points de terminaison personnalisés pour chaque cluster Aurora ou chaque cluster Aurora Serverless v2 provisionné. Vous ne pouvez pas utiliser de points de terminaison personnalisés pour les clusters Aurora Serverless v1.

Le point de terminaison personnalisé assure des connexions de base de données équilibrées selon des critères autres que la fonctionnalité de lecture seule ou de lecture/écriture des instances de base de données. Par exemple, vous pouvez définir un point de terminaison personnalisé pour vous connecter à des instances utilisant une classe d'instance AWS spécifique ou un groupe de paramètres de base de données particulier. Vous pouvez ensuite communiquer ce point de terminaison personnalisé à un groupe d'utilisateurs donné. Par exemple, vous pouvez renvoyer les utilisateurs internes vers des instances à faible capacité pour la génération de rapports ou la création de requêtes ad hoc (ponctuelles), et diriger le trafic de production vers des instances à haute capacité.

Comme la connexion peut exploiter n'importe quelle instance de base de données associée au point de terminaison personnalisé, il est conseillé de s'assurer que toutes les instances de base de données de ce groupe partagent des caractéristiques similaires. De cette manière, les performances, le capacité de la mémoire ou autres paramètres seront cohérents pour tous les utilisateurs qui se connectent à ce point de terminaison.

Cette fonction est destinée aux utilisateurs avancés dotés de types de charges de travail spécialisés où il n'est pas pratique que tous les réplicas Aurora du cluster soient identiques. Les points de terminaison personnalisés vous permettent de prédire la capacité de l'instance de base de données utilisée pour chaque connexion. Lorsque vous avez recours à des points de terminaison personnalisés, vous n'utilisez généralement pas le point de terminaison du lecteur pour ce cluster.

L'exemple suivant illustre le point de terminaison personnalisé d'une instance de base de données dans un cluster de bases de données Aurora MySQL.

```
myendpoint.cluster-custom-c7tj4example.us-east-1.rds.amazonaws.com:3306
```

Les points de terminaison personnalisés permettent de simplifier la gestion des connexions lorsque le cluster contient des instances de base de données avec des paramètres de capacité ou de configuration distincts.

Auparavant, pour obtenir des résultats similaires, vous utilisiez peut-être le mécanisme CNAME pour configurer des alias DNS à partir de votre propre domaine. L'utilisation de points de terminaison personnalisés vous évite d'avoir à mettre à jour les enregistrements CNAME lorsque la taille du cluster augmente ou diminue. Avec les points de terminaison personnalisés, vous pouvez également utiliser des connexions TLS/SSL (Transport Layer Security/Secure Sockets Layer).

Au lieu d'employer une seule instance de base de données pour chaque objectif spécifique et de vous connecter au point de terminaison de son instance, vous pouvez recourir à plusieurs groupes

d’instances de base de données spécialisées. Dans ce cas, chaque groupe dispose de son propre point de terminaison personnalisé. De cette façon, Aurora est en mesure d’effectuer l’équilibrage des connexions entre toutes les instances dédiées à des tâches telles que la création de rapports ou la gestion des requêtes internes ou de production. Les points de terminaison personnalisés distribuent les connexions entre les instances de manière passive, en utilisant le DNS pour renvoyer l’adresse IP de l’une des instances de manière aléatoire. Si l’une des instances de base de données au sein d’un groupe devient indisponible, Aurora renvoie les connexions suivantes au point de terminaison personnalisé vers l’une des autres instances de base de données associées à ce dernier.

Rubriques

Considérations relatives aux points de terminaison personnalisés dans Amazon Aurora

Utilisez les sections suivantes pour gérer les points de terminaison personnalisés, pour en spécifier les propriétés et pour utiliser les règles d’appartenance associées.

Rubriques

Gestion des points de terminaison personnalisés

Comme les nouveaux clusters Aurora ne contiennent pas de points de terminaison personnalisés, vous devez créer et gérer ces objets vous-même. Pour ce faire, vous utilisez l’AWS Management Console, l’AWS CLI ou l’API Amazon RDS.

> **Note**
>
> Vous devez également créer et gérer les points de terminaison personnalisés des clusters Aurora restaurés à partir d'instantanés. Les points de terminaison personnalisés ne sont pas inclus dans les instantanés. Ils sont recréés après leur restauration, et de nouveaux noms de points de terminaison sont choisis si le cluster restauré se trouve dans la même région que celui d'origine.

Pour manipuler les points de terminaison personnalisés à partir de l'AWS Management Console, accédez à la page de détails de votre cluster Aurora et utilisez les contrôles situés dans la section Points de terminaison personnalisés.

Pour manipuler les points de terminaison personnalisés depuis l'AWS CLI, utilisez ces opérations :

- create-db-cluster-endpoint
- describe-db-cluster-endpoints
- modify-db-cluster-endpoint
- delete-db-cluster-endpoint

Pour manipuler les points de terminaison personnalisés via l'API Amazon RDS, vous pouvez utiliser les fonctions suivantes :

- CreateDBClusterEndpoint
- DescribeDBClusterEndpoints
- ModifyDBClusterEndpoint
- DeleteDBClusterEndpoint

Spécification des propriétés des points de terminaison personnalisés

La longueur maximale du nom d'un point de terminaison personnalisé est de 63 caractères. Le format du nom est le suivant :

```
endpoint_name.cluster-custom-customer_DNS_identifier.AWS_Region.rds.amazonaws.com
```

Vous ne pouvez pas réutiliser le même nom de point de terminaison personnalisé pour d'autres clusters de la même Région AWS. L'identifiant DNS personnalisé est un identifiant unique associé à votre Compte AWS dans une Région AWS particulière.

Chaque point de terminaison est lié à un type qui détermine les instances de base de données qui peuvent être associées à ce point de terminaison. À l'heure actuelle, il peut s'agir de `READER` ou `ANY`. Les considérations suivantes s'appliquent aux types de points de terminaison personnalisés :

- Il n'est pas possible de sélectionner le type de point de terminaison personnalisé dans la AWS Management Console. Tous les points de terminaison personnalisés que vous créez via AWS Management Console sont de type `ANY`.

  Vous pouvez définir et modifier le type de point de terminaison personnalisé à l'aide de AWS CLI ou de l'API Amazon RDS.
- Seules les instances de base de données de lecteur peuvent faire partie d'un point de terminaison personnalisé `READER`.
- Les instances de base de données de lecteur et d'enregistreur peuvent faire partie d'un point de terminaison personnalisé `ANY`. Aurora dirige les connexions aux points de terminaison de cluster de type `ANY` vers n'importe quelle instance de base de données associée avec une probabilité égale. Le type `ANY` s'applique aux clusters utilisant une topologie de réplication.
- Si vous essayez de créer un point de terminaison personnalisé doté d'un type qui ne correspond pas à la configuration de la réplication d'un cluster, Aurora renvoie une erreur.

Règles d'appartenance des points de terminaison personnalisés

Lorsque vous ajoutez ou supprimez une instance de base de données dans un point de terminaison personnalisé, toutes les connexions existantes à cette instance restent actives.

Vous pouvez définir une liste des instances de base de données à inclure (ou à exclure) dans un point de terminaison personnalisé. C'est ce que l'on appelle des listes statiques et des listes d'exclusion, respectivement. Vous pouvez utiliser le mécanisme d'inclusion/exclusion pour subdiviser davantage les groupes d'instances de base de données et pour vous assurer que l'ensemble de points de terminaison personnalisés couvre toutes les instances de base de données du cluster. Chaque point de terminaison personnalisé ne peut contenir qu'un de ces types de liste.

Dans AWS Management Console :

- Le choix est représenté par la case à cocher Attach future instances added to this cluster (Attacher les instances futures ajoutées à ce cluster). Lorsque cette case n'est pas cochée, le point de terminaison personnalisé utilise une liste statique contenant uniquement les instances de base de données spécifiées sur la page. Lorsque vous cochez cette case, le point de terminaison personnalisé utilise une liste d'exclusion. Dans ce cas, le point de terminaison personnalisé représente toutes les instances de base de données du cluster (y compris celles que vous ajouterez par la suite), sauf celles qui ne sont pas sélectionnées sur la page.
- La console ne vous permet pas de spécifier le type de point de terminaison. Tout point de terminaison personnalisé créé à l'aide de la console est du type `ANY`.

  Par conséquent, Aurora ne modifie pas l'appartenance du point de terminaison personnalisé lorsque les instances de base de données changent les rôles entre l'enregistreur et le lecteur en raison d'un basculement ou d'une promotion.

Dans l'AWS CLI et l'API Amazon RDS :

- Vous pouvez spécifier le type de point de terminaison. Par conséquent, lorsque le type de point de terminaison est défini sur `READER`, l'appartenance du point de terminaison est automatiquement ajustée lors des basculements et des promotions.

  Par exemple, un point de terminaison personnalisé avec le type `READER` inclut un réplica Aurora qui est ensuite promu en tant qu'instance de base de données d'enregistreur. La nouvelle instance d'enregistreur ne fait plus partie du point de terminaison personnalisé.
- Vous pouvez ajouter des membres individuels aux listes et les supprimer de celles-ci une fois qu'ils ont changé de rôle. Utilisez la commande de l'AWS CLI modify-db-cluster-endpoint ou l'opération d'API ModifyDBClusterEndpoint.

Vous pouvez associer une instance de base de données à plusieurs points de terminaison personnalisés. Supposons, par exemple, que vous ajoutiez une nouvelle instance de base de données à un cluster ou qu'Aurora ajoute une instance de base de données automatiquement via le mécanisme Auto Scaling. Dans ce cas, l'instance de base de données est ajoutée à tous les points de terminaison personnalisés auxquels elle est éligible. Les points de terminaison auxquels l'instance de base de données est ajoutée dépendent de leur type (`READER` ou `ANY`) et des listes statiques ou d'exclusion définies pour chacun d'eux. Par exemple, si le point de terminaison comprend une liste statique d'instances de base de données, les réplicas Aurora qui viennent d'être ajoutés ne sont pas inclus dans ce point de terminaison. Inversement, si le point de terminaison dispose d'une liste

d’exclusion, les réplicas Aurora qui viennent d’être ajoutés sont inclus dans le point de terminaison s’ils ne font pas partie de la liste d’exclusion et si leur rôle correspond au type du point de terminaison personnalisé.

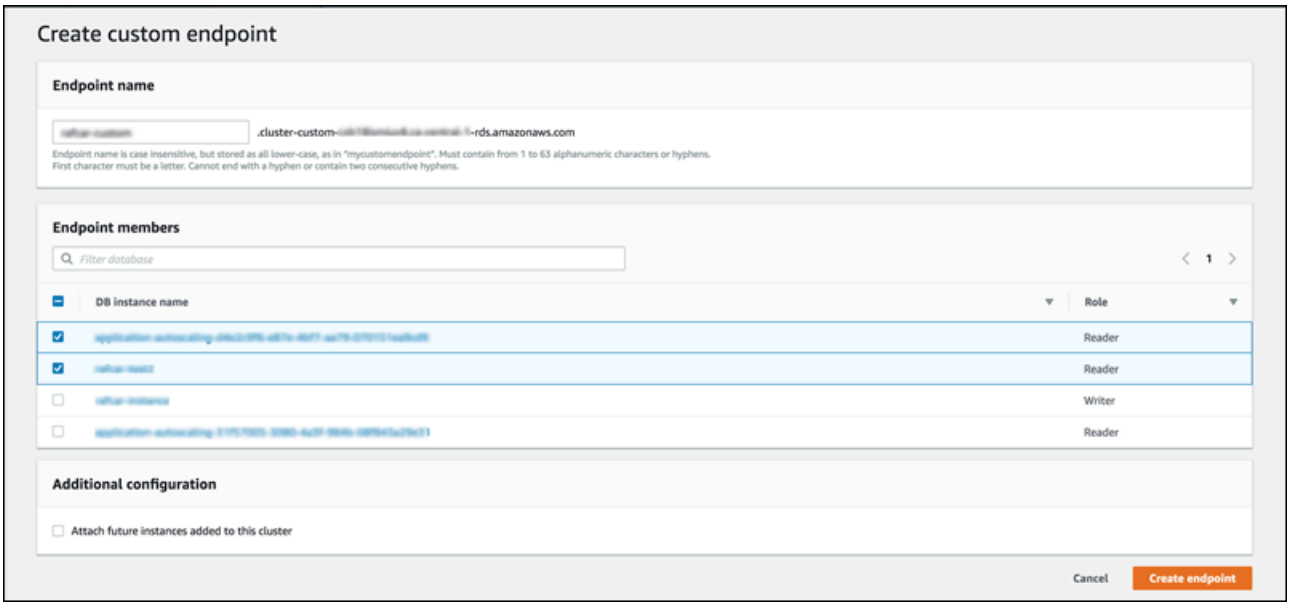
Si un réplica Aurora devient indisponible, il reste associé aux points de terminaison personnalisés. Par exemple, il continue à faire partie du point de terminaison personnalisé s’il n’est pas sain, s’il est arrêté, s’il redémarre, etc. Toutefois, il doit redevenir disponible pour vous puissiez vous y connecter via ces points de terminaison.

Création d’un point de terminaison personnalisé

Créez un point de terminaison personnalisé à l’aide de la AWS Management Console, de l’AWS CLI ou de l’API Amazon RDS.

Console

Pour créer un point de terminaison personnalisé avec AWS Management Console, accédez à la page de détails du cluster et choisissez l’action `Create custom endpoint` dans la section Endpoints (Points de terminaison). Choisissez un nom pour le point de terminaison personnalisé. Ce nom sera spécifique à votre ID utilisateur et à votre région. Pour choisir une liste d’instances de base de données qui restera la même à mesure que le cluster s’élargira, laissez la case Attach future instances added to this cluster (Attacher les instances futures ajoutées à ce cluster) décochée. Lorsque vous cochez cette case, le point de terminaison personnalisé ajoute les nouvelles instances de manière dynamique à mesure que vous les ajoutez au cluster.



Il n'est pas possible de sélectionner le type de point de terminaison personnalisé dans la AWS Management Console. Tous les points de terminaison personnalisés que vous créez via la AWS Management Console sont de type ANY.

AWS CLI

Pour créer un point de terminaison personnalisé avec l'AWS CLI, exécutez la commande create-db-cluster-endpoint.

La commande suivante crée un point de terminaison personnalisé attaché à un cluster spécifique. Initialement, le point de terminaison est associé à toutes les instances de réplica Aurora du cluster. Une commande ultérieure l'associe à l'ensemble spécifique d'instances de base de données du cluster.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster-endpoint --db-cluster-endpoint-identifier custom-endpoint-
doc-sample \
  --endpoint-type reader \
  --db-cluster-identifier cluster_id

aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier custom-endpoint-
doc-sample \
  --static-members instance_name_1 instance_name_2
```

Pour Windows :

```
aws rds create-db-cluster-endpoint --db-cluster-endpoint-identifier custom-endpoint-
doc-sample ^
  --endpoint-type reader ^
  --db-cluster-identifier cluster_id

aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier custom-endpoint-
doc-sample ^
  --static-members instance_name_1 instance_name_2
```

API RDS

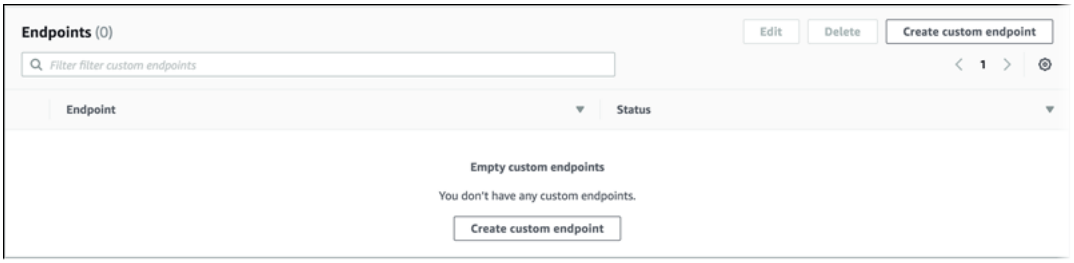
Pour créer un point de terminaison personnalisé avec l'API RDS, exécutez l'opération CreateDBClusterEndpoint.

Affichage des points de terminaison personnalisés

Console

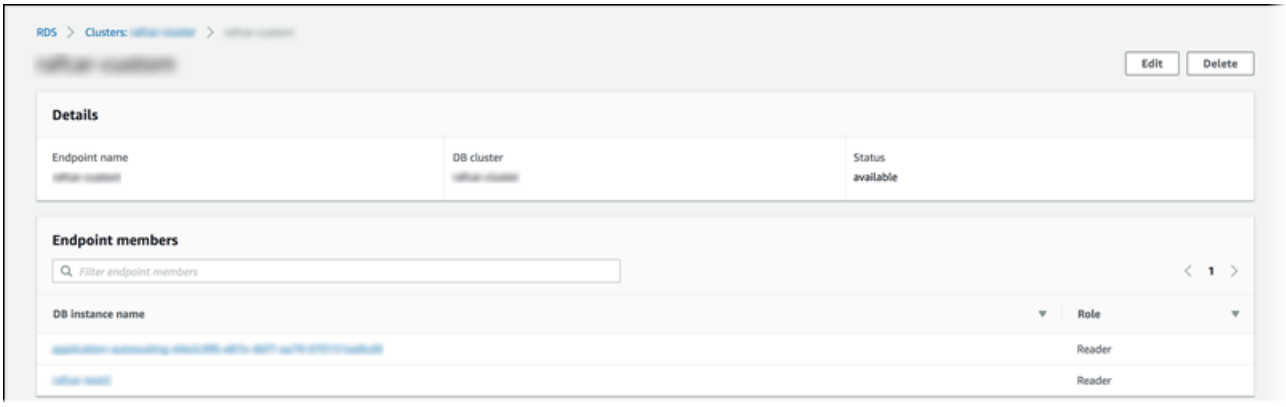
Pour afficher les points de terminaison personnalisés avec AWS Management Console, accédez à la page de détails du cluster et examinez la section Endpoints (Points de terminaison). Cette section contient uniquement des informations sur les points de terminaison personnalisés. Les détails relatifs aux points de terminaison intégrés sont indiqués dans la section Details (Détails) principale. Pour consulter les détails relatifs à un point de terminaison personnalisé, sélectionnez son nom afin d'afficher la page de détails correspondante.

La capture d'écran suivante présente la liste de points de terminaison personnalisés d'un cluster Aurora, laquelle est vide initialement.



Une fois que vous avez créé des points de terminaison personnalisés pour ce cluster, ils apparaissent dans la section Endpoints (Points de terminaison).

Accédez à la page de détails pour afficher les instances de base de données auxquelles le point de terminaison est associé.



Pour voir si les nouvelles instances de DB ajoutées au cluster sont automatiquement ajoutées au point de terminaison, ouvrez la page Edit (Modifier) du point de terminaison.

AWS CLI

Pour afficher les points de terminaison personnalisés avec l'AWS CLI, exécutez la commande describe-db-cluster-endpoints.

La commande suivante affiche les points de terminaison personnalisés associés à un cluster spécifique dans une région donnée. La sortie inclut à la fois les points de terminaison intégrés et les points de terminaison personnalisés.

Pour Linux, macOS ou Unix :

```
aws rds describe-db-cluster-endpoints --region region_name \
  --db-cluster-identifier cluster_id
```

Pour Windows :

```
aws rds describe-db-cluster-endpoints --region region_name ^
  --db-cluster-identifier cluster_id
```

Voici un exemple de sortie générée par la commande `describe-db-cluster-endpoints`. La valeur `EndpointType` ou `WRITER` pour`READER` indique les points de terminaison en lecture/écriture et en lecture seule du cluster. La valeur `EndpointType` pour `CUSTOM` indique les points de terminaison que vous créez et les instances de base de données associées que vous choisissez. L'un des points de terminaison ne contient aucune valeur dans le champ `StaticMembers`, ce qui indique qu'il est associé à un ensemble précis d'instances de base de données. L'autre point de terminaison ne contient aucune valeur dans le champ `ExcludedMembers`, ce qui indique qu'il est associé à toutes les instances de base de données autres que celles qui sont répertoriées dans la liste `ExcludedMembers`. Ce deuxième type de point de terminaison personnalisé s'élargit afin de pouvoir accueillir les nouvelles instances que vous ajoutez au cluster.

```
{
 "DBClusterEndpoints": [
  {
   "Endpoint": "custom-endpoint-demo.cluster-c7tj4example.ca-
central-1.rds.amazonaws.com",
   "Status": "available",
   "DBClusterIdentifier": "custom-endpoint-demo",
   "EndpointType": "WRITER"
```

```
  },
  {
   "Endpoint": "custom-endpoint-demo.cluster-ro-c7tj4example.ca-
central-1.rds.amazonaws.com",
   "Status": "available",
   "DBClusterIdentifier": "custom-endpoint-demo",
   "EndpointType": "READER"
  },
  {
   "CustomEndpointType": "ANY",
   "DBClusterEndpointIdentifier": "powers-of-2",
   "ExcludedMembers": [],
   "DBClusterIdentifier": "custom-endpoint-demo",
   "Status": "available",
   "EndpointType": "CUSTOM",
   "Endpoint": "powers-of-2.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
   "StaticMembers": [
     "custom-endpoint-demo-04",
     "custom-endpoint-demo-08",
     "custom-endpoint-demo-01",
     "custom-endpoint-demo-02"
   ],
   "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHXQKFU6J6NV5FHU",
   "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:powers-of-2"
  },
  {
   "CustomEndpointType": "ANY",
   "DBClusterEndpointIdentifier": "eight-and-higher",
   "ExcludedMembers": [
     "custom-endpoint-demo-04",
     "custom-endpoint-demo-02",
     "custom-endpoint-demo-07",
     "custom-endpoint-demo-05",
     "custom-endpoint-demo-03",
     "custom-endpoint-demo-06",
     "custom-endpoint-demo-01"
   ],
   "DBClusterIdentifier": "custom-endpoint-demo",
   "Status": "available",
   "EndpointType": "CUSTOM",
```

```
   "Endpoint": "eight-and-higher.cluster-custom-123456789012.ca-
central-1.rds.amazonaws.com",
   "StaticMembers": [],
   "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHYQKFU6J6NV5FHU",
   "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:eight-and-higher"
  }
 ]
}
```

API RDS

Pour afficher les points de terminaison personnalisés avec l'API RDS, exécutez l'opération DescribeDBClusterEndpoints.html.

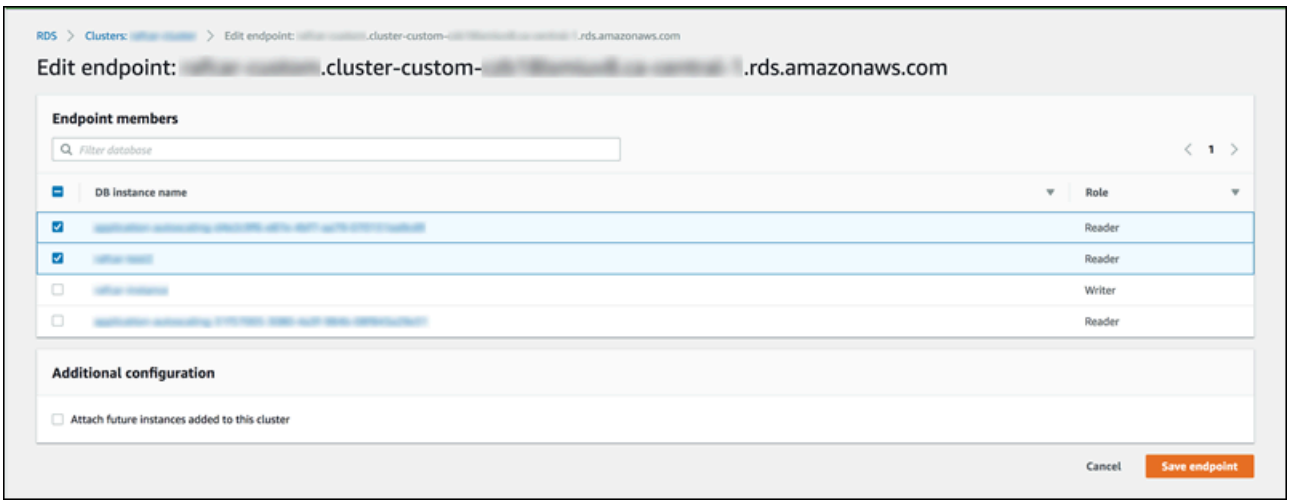
Modification d'un point de terminaison personnalisé

Vous pouvez modifier les propriétés d'un point de terminaison pour changer les instances de base de données qui lui sont associées. Vous pouvez également changer le type de liste dans laquelle un point de terminaison est inclus : liste statique ou liste d'exclusion. Pour plus d'informations sur les propriétés de ces points de terminaison, consultez Règles d'appartenance des points de terminaison personnalisés.

Vous pouvez continuer à vous connecter à un point de terminaison personnalisé et à l'utiliser pendant que les changements d'une action de modification sont en cours.

Console

Pour modifier un point de terminaison personnalisé avec AWS Management Console, vous pouvez sélectionner le point de terminaison sur la page de détails du cluster ou afficher la page de détails du point de terminaison, puis choisir l'action Edit (Modifier).



AWS CLI

Pour modifier un point de terminaison personnalisé avec l'AWS CLI, exécutez la commande modify-db-cluster-endpoint.

Les commandes suivantes modifient l'ensemble d'instances de base de données qui s'appliquent à un point de terminaison personnalisé et permettent éventuellement de passer d'une liste statique à une liste d'exclusion, ou inversement. Les paramètres `--static-members` et `--excluded-members` incluent une liste d'identifiants d'instance de base de données, séparés par un espace.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier my-custom-endpoint
 \
  --static-members db-instance-id-1 db-instance-id-2 db-instance-id-3 \
  --region region_name

aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier my-custom-endpoint
 \
  --excluded-members db-instance-id-4 db-instance-id-5 \
  --region region_name
```

Pour Windows :

```
aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier my-custom-endpoint
 ^
  --static-members db-instance-id-1 db-instance-id-2 db-instance-id-3 ^
  --region region_name
```

```
aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier my-custom-endpoint
 ^
  --excluded-members db-instance-id-4 db-instance-id-5 ^
  --region region_name
```

API RDS

Pour modifier un point de terminaison personnalisé avec l'API RDS, exécutez l'opération ModifyDBClusterEndpoint.html.

Suppression d'un point de terminaison personnalisé

Supprimez un point de terminaison personnalisé à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API Amazon RDS.

Console

Pour supprimer un point de terminaison personnalisé avec AWS Management Console, accédez à la page de détails du cluster, sélectionnez le point de terminaison personnalisé approprié, puis sélectionnez l'action Delete (Supprimer).

AWS CLI

Pour supprimer un point de terminaison personnalisé avec l'AWS CLI, exécutez la commande delete-db-cluster-endpoint.

La commande suivante supprime un point de terminaison personnalisé. Il n'est pas nécessaire de spécifier le cluster associé, mais vous devez indiquer la région.

Pour Linux, macOS ou Unix :

```
aws rds delete-db-cluster-endpoint --db-cluster-endpoint-identifier custom-end-point-id
 \
  --region region_name
```

Pour Windows :

```
aws rds delete-db-cluster-endpoint --db-cluster-endpoint-identifier custom-end-point-id
 ^
  --region region_name
```

API RDS

Pour supprimer un point de terminaison personnalisé avec l'API RDS, exécutez l'opération DeleteDBClusterEndpoint.

Examples d'AWS CLI de points de terminaison personnalisés pour Amazon Aurora

Le didacticiel suivant utilise des exemples de l'AWS CLI avec la syntaxe du shell Unix pour montrer comment définir un cluster avec plusieurs instances de base de données de petite taille et quelques instances de base de données de grande taille, et comment créer des points de terminaison personnalisés pour connecter chaque ensemble d'instances de base de données. Pour exécuter des commandes similaires sur votre propre système, vous devez connaître les notions de base liées à l'utilisation des clusters Aurora et de l'AWS CLI afin de fournir vos propres valeurs pour divers paramètres tels que la région, le groupe de sous-réseau et le groupe de sécurité du VPC.

Cet exemple présente les étapes de configuration initiales : création d'un cluster Aurora et ajout d'instances de base de données à ce cluster. Il s'agit d'un cluster hétérogène, ce qui signifie que toutes les instances de base de données n'ont pas la même capacité. La plupart des instances utilisent la classe d'instance AWS `db.r4.4xlarge`, mais les deux dernières instances de base de données utilisent `db.r4.16xlarge`. Chacun de ces exemples de commandes `create-db-instance` imprime sa sortie à l'écran et enregistre une copie du code JSON dans un fichier pour permettre tout examen ultérieur.

```
aws rds create-db-cluster --db-cluster-identifier custom-endpoint-demo --engine aurora-
mysql \
     --engine-version 8.0.mysql_aurora.3.04.0 --master-username $MASTER_USER --manage-
master-user-password \
     --db-subnet-group-name $SUBNET_GROUP  --vpc-security-group-ids $VPC_SECURITY_GROUP
 \
     --region $REGION

for i in 01 02 03 04 05 06 07 08
do
  aws rds create-db-instance --db-instance-identifier custom-endpoint-demo-${i} \
```

```
    --engine aurora --db-cluster-identifier custom-endpoint-demo --db-instance-class
 db.r4.4xlarge \
    --region $REGION \
    | tee custom-endpoint-demo-${i}.json
done

for i in 09 10
do
  aws rds create-db-instance --db-instance-identifier custom-endpoint-demo-${i} \
    --engine aurora --db-cluster-identifier custom-endpoint-demo --db-instance-class
 db.r4.16xlarge \
    --region $REGION \
    | tee custom-endpoint-demo-${i}.json
done
```

Les instances de grande taille sont réservées à des types spécifiques de requêtes de rapport. Pour éviter qu'elles ne soient promues en tant qu'instances principales, l'exemple suivant attribue la priorité la plus faible à leur niveau de promotion. Cet exemple spécifie l'option `--manage-master-user-password` permettant de générer le mot de passe de l'utilisateur principal et de le gérer dans Secrets Manager. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager. Vous pouvez également utiliser l'option `--master-password` pour spécifier et gérer vous-même le mot de passe.

```
for i in 09 10
do
  aws rds modify-db-instance --db-instance-identifier custom-endpoint-demo-${i} \
    --region $REGION --promotion-tier 15
done
```

Supposons que vous ne souhaitiez utiliser les deux instances les plus grandes que pour les requêtes les plus communément utilisées. Pour ce faire, vous pouvez d'abord créer un point de terminaison personnalisé en lecture seule. Ensuite, vous pouvez ajouter une liste statique de membres de sorte que le point de terminaison se connecte uniquement à ces instances de base de données. Ces instances ayant déjà le niveau de promotion le plus faible, il est peu probable qu'elles soient promues en tant qu'instances principales. Si l'une d'elles est promue en tant qu'instance principale, elle n'est plus accessible via ce point de terminaison, car le type `READER` a été spécifié au lieu du type `ANY`.

L'exemple suivant présente les commandes de création et de modification du point de terminaison, ainsi qu'un extrait de code JSON indiquant l'état initial et l'état modifié du point de terminaison personnalisé.

```
$ aws rds create-db-cluster-endpoint --region $REGION \
    --db-cluster-identifier custom-endpoint-demo \
    --db-cluster-endpoint-identifier big-instances --endpoint-type reader
{
    "EndpointType": "CUSTOM",
    "Endpoint": "big-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
    "DBClusterEndpointIdentifier": "big-instances",
    "DBClusterIdentifier": "custom-endpoint-demo",
    "StaticMembers": [],
    "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHXQKFU6J6NV5FHU",
    "ExcludedMembers": [],
    "CustomEndpointType": "READER",
    "Status": "creating",
    "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:big-instances"
}

$ aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier big-instances \
    --static-members custom-endpoint-demo-09 custom-endpoint-demo-10 --region $REGION
{
    "EndpointType": "CUSTOM",
    "ExcludedMembers": [],
    "DBClusterEndpointIdentifier": "big-instances",
    "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHXQKFU6J6NV5FHU",
    "CustomEndpointType": "READER",
    "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:big-instances",
    "StaticMembers": [
        "custom-endpoint-demo-10",
        "custom-endpoint-demo-09"
    ],
    "Status": "modifying",
    "Endpoint": "big-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
    "DBClusterIdentifier": "custom-endpoint-demo"
}
```

Le point de terminaison READER par défaut du cluster peut se connecter à la fois aux instances de base de données de petite taille et de grande taille, ce qui ne permet pas d'anticiper facilement les performances des requêtes et l'évolutivité lorsque le cluster est occupé. Pour répartir la

charge de travail de manière explicite entre les ensembles d'instances de base de données, vous pouvez ignorer le point de terminaison READER par défaut et créer un second point de terminaison personnalisé qui se connecte à toutes les autres instances de base de données. Pour atteindre cet objectif, l'exemple suivant crée un point de terminaison personnalisé et ajoute une liste d'exclusion. Toute autre instance de base de données que vous ajouterez ultérieurement au cluster sera automatiquement ajoutée à ce point de terminaison. Le type ANY signifie que ce point de terminaison est associé à huit instances au total : l'instance principale et sept autres réplicas Aurora. Si cet exemple utilisait le type READER, le point de terminaison personnalisé ne serait associé qu'aux sept réplicas Aurora.

```
$ aws rds create-db-cluster-endpoint --region $REGION --db-cluster-identifier custom-
endpoint-demo \
    --db-cluster-endpoint-identifier small-instances --endpoint-type any
{
    "Status": "creating",
    "DBClusterEndpointIdentifier": "small-instances",
    "CustomEndpointType": "ANY",
    "EndpointType": "CUSTOM",
    "Endpoint": "small-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
    "StaticMembers": [],
    "ExcludedMembers": [],
    "DBClusterIdentifier": "custom-endpoint-demo",
    "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:small-instances",
    "DBClusterEndpointResourceIdentifier": "cluster-
endpoint-6RDDXQOC3AKKZT2PRD7ST37BMY"
}

$ aws rds modify-db-cluster-endpoint --db-cluster-endpoint-identifier small-instances \
    --excluded-members custom-endpoint-demo-09 custom-endpoint-demo-10 --region $REGION
{
    "DBClusterEndpointIdentifier": "small-instances",
    "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:c7tj4example:cluster-
endpoint:small-instances",
    "DBClusterEndpointResourceIdentifier": "cluster-
endpoint-6RDDXQOC3AKKZT2PRD7ST37BMY",
    "CustomEndpointType": "ANY",
    "Endpoint": "small-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
    "EndpointType": "CUSTOM",
    "ExcludedMembers": [
```

```
        "custom-endpoint-demo-09",
        "custom-endpoint-demo-10"
    ],
    "StaticMembers": [],
    "DBClusterIdentifier": "custom-endpoint-demo",
    "Status": "modifying"
}
```

L'exemple suivant renvoie vérifie l'état du point de terminaison du cluster. Le cluster inclut toujours son point de terminaison d'origine, avec la valeur `EndPointType` pour `WRITER`, que vous pouvez continuer à utiliser pour l'administration, ainsi que les opérations ETL et d'autres opérations d'écriture. Il a toujours son point de terminaison `READER` d'origine, que vous n'utiliseriez pas, car chaque connexion établie avec celui-ci peut être renvoyée vers une instance de base de données de petite taille ou de grande taille. Les points de terminaison personnalisés permettent de rendre ce comportement prévisible, en assurant que les connexions utilisent l'une des instances de base de données de petite taille ou de grande taille en fonction du point de terminaison que vous spécifiez.

```
$ aws rds describe-db-cluster-endpoints --region $REGION
{
    "DBClusterEndpoints": [
        {
            "EndpointType": "WRITER",
            "Endpoint": "custom-endpoint-demo.cluster-c7tj4example.ca-
central-1.rds.amazonaws.com",
            "Status": "available",
            "DBClusterIdentifier": "custom-endpoint-demo"
        },
        {
            "EndpointType": "READER",
            "Endpoint": "custom-endpoint-demo.cluster-ro-c7tj4example.ca-
central-1.rds.amazonaws.com",
            "Status": "available",
            "DBClusterIdentifier": "custom-endpoint-demo"
        },
        {
            "Endpoint": "small-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
            "CustomEndpointType": "ANY",
            "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:small-instances",
            "ExcludedMembers": [
                "custom-endpoint-demo-09",
```

```
                "custom-endpoint-demo-10"
            ],
            "DBClusterEndpointResourceIdentifier": "cluster-
endpoint-6RDDXQOC3AKKZT2PRD7ST37BMY",
            "DBClusterIdentifier": "custom-endpoint-demo",
            "StaticMembers": [],
            "EndpointType": "CUSTOM",
            "DBClusterEndpointIdentifier": "small-instances",
            "Status": "modifying"
        },
        {
            "Endpoint": "big-instances.cluster-custom-c7tj4example.ca-
central-1.rds.amazonaws.com",
            "CustomEndpointType": "READER",
            "DBClusterEndpointArn": "arn:aws:rds:ca-central-1:111122223333:cluster-
endpoint:big-instances",
            "ExcludedMembers": [],
            "DBClusterEndpointResourceIdentifier": "cluster-endpoint-
W7PE3TLLFNSHXQKFU6J6NV5FHU",
            "DBClusterIdentifier": "custom-endpoint-demo",
            "StaticMembers": [
                "custom-endpoint-demo-10",
                "custom-endpoint-demo-09"
            ],
            "EndpointType": "CUSTOM",
            "DBClusterEndpointIdentifier": "big-instances",
            "Status": "available"
        }
    ]
}
```

Les exemples finaux illustrent la façon dont des connexions de base de données successives des points de terminaison personnalisés se connectent aux diverses instances de base de données du cluster Aurora. Le point de terminaison `small-instances` se connecte toujours aux instances de base de données `db.r4.4xlarge`, qui correspondent aux hôtes dont le nombre est le plus faible dans ce cluster.

```
$ mysql -h small-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -
u $MYUSER -p
mysql> select @@aurora_server_id;
+--------------------------+
| @@aurora_server_id       |
+--------------------------+
```

```
| custom-endpoint-demo-02 |
+-------------------------+

$ mysql -h small-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -
u $MYUSER -p
mysql> select @@aurora_server_id;
+-------------------------+
| @@aurora_server_id      |
+-------------------------+
| custom-endpoint-demo-07 |
+-------------------------+

$ mysql -h small-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -
u $MYUSER -p
mysql> select @@aurora_server_id;
+-------------------------+
| @@aurora_server_id      |
+-------------------------+
| custom-endpoint-demo-01 |
+-------------------------+
```

Le point de terminaison `big-instances` se connecte toujours aux instances de base de données `db.r4.16xlarge`, qui correspondent aux deux hôtes dont le nombre est le plus élevé dans ce cluster.

```
$ mysql -h big-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -u
 $MYUSER -p
mysql> select @@aurora_server_id;
+-------------------------+
| @@aurora_server_id      |
+-------------------------+
| custom-endpoint-demo-10 |
+-------------------------+

$ mysql -h big-instances.cluster-custom-c7tj4example.ca-central-1.rds.amazonaws.com -u
 $MYUSER -p
mysql> select @@aurora_server_id;
+-------------------------+
| @@aurora_server_id      |
+-------------------------+
| custom-endpoint-demo-09 |
+-------------------------+
```

Classes d’instance de base de données Amazon Aurora

La classe d’instance de base de données détermine la capacité de calcul et de mémoire d’une instance de base de données Amazon Aurora. La classe d’instance de base de données dont vous avez besoin varie selon vos exigences en mémoire et en puissance de traitement.

Une classe d’instance de base de données comprend à la fois le type de classe d’instance de base de données et la taille. Par exemple, db.r6g est une classe d’instance de base de données optimisée pour la mémoire, alimentée par les processeurs AWS Graviton2. Dans le type de classe d’instance db.m6g, db.r6g.2xlarge est une classe d’instance de base de données. La taille de cette classe est 2xlarge.

Pour plus d’informations sur la tarification des classes d’instance, consultez Tarification d’Amazon RDS.

Pour plus d’informations sur les types de classes d’instance de base de données, les moteurs de base de données pris en charge, les Régions AWS prises en charge, la ou les spécifications matérielles des classes d’instance de base de données, consultez les sections suivantes.

Rubriques

Types de classes d’instance de base de données

Amazon Aurora prend en charge les classes d’instance de base de données pour les cas d’utilisation suivants :

- Aurora Serverless v2
- Optimisé pour la mémoire
- À performances extensibles
- Optimized Reads

Pour plus d'informations sur les types d' EC2 instances Amazon, consultez la section Types d'instances dans la EC2 documentation Amazon.

Type de classe d’instance Aurora Serverless v2

Le type Aurora Serverless v2 suivant est disponible :

- db.serverless : un type spécial de classe d’instance de base de données utilisé par Aurora Serverless v2. Aurora ajuste dynamiquement les ressources de calcul, de mémoire et de mise en réseau en fonction de l’évolution de la charge de travail. Pour plus de détails sur l’utilisation, consultez Utiliser Aurora Serverless v2.

Types de classes d’instance à mémoire optimisée

La famille X à mémoire optimisée prend en charge les classes d’instance suivantes :

- db.x2g — Classes d'instance optimisées pour les applications gourmandes en mémoire et alimentées par les processeurs Graviton2. AWS Ces classes d’instance offrent un faible coût par Gio de mémoire.

  Vous pouvez modifier une instance de base de données pour utiliser l'une des classes d'instance de base de données alimentées par les processeurs AWS Graviton2. Pour ce faire, suivez les mêmes étapes que pour toute autre modification d’une instance de base de données.

La famille R à mémoire optimisée prend en charge les types de classe d’instance suivants :

- db.r8g : classes d’instance alimentées par des processeurs AWS Graviton4. Ces classes d’instance sont idéales pour exécuter des charges de travail exigeantes en mémoire. Ces instances offrent des tailles d'instance plus importantes avec jusqu'à 3 fois plus de v CPUs et de mémoire que les instances db.r7g basées sur AWS Graviton3 de septième génération. Ils sont alimentés par le système AWS Nitro, une combinaison de matériel dédié et d'hyperviseur léger.
- Vous pouvez modifier une instance de base de données pour utiliser l'une des classes d'instance de base de données alimentées par les processeurs AWS Graviton4. Pour ce faire, suivez les mêmes étapes que pour toute autre modification d’une instance de base de données.
- db.r7g — Classes d'instances alimentées par les processeurs Graviton3. AWS Ces classes d’instance sont idéales pour exécuter des charges de travail exigeantes en mémoire.

Vous pouvez modifier une instance de base de données pour utiliser l'une des classes d'instance de base de données alimentées par les processeurs AWS Graviton3. Pour ce faire, suivez les mêmes étapes que pour toute autre modification d’une instance de base de données. Ils sont alimentés par le système AWS Nitro, une combinaison de matériel dédié et d'hyperviseur léger.

- db.r7i : classes d’instance alimentées par des processeurs Intel Xeon Scalable de 4e génération. Ces classes d’instance sont certifiées SAP et sont idéales pour les charges de travail qui demandent beaucoup de mémoire. Vous pouvez modifier une instance de base de données de manière à utiliser l’une des classes d’instance de base de données alimentées par des processeurs Intel Xeon Scalable de 4e génération. Pour ce faire, suivez les mêmes étapes que pour toute autre modification d’une instance de base de données. Ils sont alimentés par le système AWS Nitro, une combinaison de matériel dédié et d'hyperviseur léger.
- db.r6g : classes d’instance alimentées par des processeurs AWS Graviton2. Ces classes d’instance sont idéales pour exécuter des charges de travail exigeantes en mémoire. Ils sont alimentés par le système AWS Nitro, une combinaison de matériel dédié et d'hyperviseur léger.
- Vous pouvez modifier une instance de base de données pour utiliser l'une des classes d'instance de base de données alimentées par les processeurs AWS Graviton2. Pour ce faire, suivez les mêmes étapes que pour toute autre modification d’une instance de base de données.
- db.r6i : classes d’instance alimentées par des processeurs Intel Xeon Scalable de 3e génération. Ces classes d’instance sont certifiées SAP et sont idéales pour les charges de travail qui demandent beaucoup de mémoire .
- db.r4 : classes d’instance optimisées pour les applications gourmandes en mémoire. Ces classes d’instance offrent une amélioration de la mise en réseau et des performances Amazon Elastic Block Store (Amazon EBS). Ils sont alimentés par le système AWS Nitro, une combinaison de matériel dédié et d'hyperviseur léger.
- db.r4 : ces classes d’instance sont uniquement prises en charge pour Aurora MySQL 2.x et Aurora PostgreSQL versions 11 et 12. Pour tous les clusters de bases de données Aurora qui utilisent des classes d’instance de base de données db.r4, nous vous recommandons de passer à une classe d’instance de base de données de génération supérieure dès que possible.

  Les classes d’instance db.r4 ne sont pas disponibles pour la configuration du stockage du cluster Aurora I/O-Optimized.

Types de classes d’instance à performances extensibles

Les types de classes d’instance de base de données à performances extensibles disponibles sont les suivants :

- db.t4g — Classes d'instance à usage général alimentées par des processeurs Graviton2 basés sur ARM. AWS Ces classes d’instance offrent de meilleures performances de prix que les précédentes classes d’instance de base de données de performance à performances extensible pour un large ensemble de charges de travail extensibles à usage général. Les instances Amazon RDS db.t4g sont configurées pour le mode illimité. Cela signifie qu’elles peuvent dépasser le niveau de base d’utilisation de l’UC sur une période de 24 heures moyennant des frais supplémentaires.

  Vous pouvez modifier une instance de base de données pour utiliser l'une des classes d'instance de base de données alimentées par les processeurs AWS Graviton2. Pour ce faire, suivez les mêmes étapes que pour toute autre modification d’une instance de base de données.
- db.t2 : classes d’instance qui fournissent un niveau de performance de base, avec la possibilité de transmission étendue jusqu’à une utilisation intégrale de l’UC. Les instances db.t3 sont configurées pour le mode Illimité. Ces classes d’instance offrent une plus grande capacité de calcul que les précédentes classes d’instance db.t2. Elles sont alimentées par le système AWS Nitro, qui allie un matériel dédié et un hyperviseur léger. Nous recommandons d’utiliser ces classes d’instance uniquement pour les serveurs de développement et de test, ou pour d’autres serveurs non dédiés à la production.
- db.t2 : classes d’instance qui fournissent un niveau de performance de base, avec la possibilité de transmission étendue jusqu’à une utilisation intégrale de l’UC. Les instances db.t2 sont configurées pour le mode Illimité. Nous recommandons d’utiliser ces classes d’instance uniquement pour les serveurs de développement et de test, ou pour d’autres serveurs non dédiés à la production.

  Les classes d’instance db.t2 ne sont pas disponibles pour la configuration du stockage du cluster Aurora I/O-Optimized.

> **Note**
>
> Nous recommandons d’utiliser uniquement les classes d’instance de base de données T pour les serveurs de développement, de test ou non dédiés à la production. Pour obtenir des recommandations plus détaillées pour les classes d’instance T, consultez Utilisation de classes d’instance T pour le développement et les tests.

Pour plus d'informations sur les spécifications matérielles de classe d'instance de base de données, consultez Spécifications matérielles pour les classes d'instance de base de données pour Aurora.

Type de classe d'instance Optimized Reads

Les types de classe d'instances Optimized Reads disponibles sont les suivants :

- db.r8gd — Classes d'instances alimentées par les processeurs Graviton4. Ces classes d'instances sont idéales pour exécuter des charges de travail gourmandes en mémoire et offrent un stockage SSD local au niveau NVMe des blocs pour les applications nécessitant un stockage local à haut débit et à faible latence. Ils offrent une mémoire maximale de 1,5 TiB et jusqu'à 11,4 To de stockage SSD à connexion directe NVMe.
- db.r6gd — Classes d'instances alimentées par les processeurs Graviton2. AWS Ces classes d'instances sont idéales pour exécuter des charges de travail gourmandes en mémoire et offrent un stockage SSD local au niveau NVMe des blocs pour les applications nécessitant un stockage local à haut débit et à faible latence.
- db.r6i : classes d'instance alimentées par des processeurs Intel Xeon Scalable de 3e génération. Ces classes d'instance sont certifiées SAP et sont idéales pour les charges de travail qui demandent beaucoup de mémoire. Ils offrent une mémoire maximale de 1 TiB et jusqu'à 7,6 To de stockage SSD à connexion directe. NVMe

Moteurs de base de données pris en charge pour les classes d'instance de base de données

Les tableaux suivants présentent les classes d'instance de base de données prises en charge pour les moteurs de base de données Amazon Aurora.

db.serverless : classe d'instance Aurora Serverless v2 avec mise à l'échelle automatique de la capacité

Classe d'instance	Aurora MySQL	Aurora PostgreSQL
db.serverless	Consultez Régions et moteurs de base de données Aurora pris en charge pour Aurora Serverless v2.	Consultez Régions et moteurs de base de données Aurora pris en charge pour Aurora Serverless v2.

db.x2g — classes d'instance optimisées pour la mémoire alimentées par les processeurs Graviton2 AWS

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.x2g.16xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.x2g.12xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.x2g.8xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.x2g.4xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.x2g.2xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.x2g.xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
		et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.x2g.large	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures

db.r6gd — Classes d'instance de lecture optimisées alimentées par les processeurs Graviton2 AWS

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r6g.16xlarge	Non	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.4 et ultérieures, 14.9 et ultérieures
db.r6g.12xlarge	Non	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.4 et ultérieures, 14.9 et ultérieures
db.r6g.8xlarge	Non	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.4 et ultérieures, 14.9 et ultérieures
db.r6g.4xlarge	Non	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.4 et ultérieures, 14.9 et ultérieures
db.r6g.2xlarge	Non	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.4 et ultérieures, 14.9 et ultérieures
db.r6gd.xlarge	Non	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.4 et ultérieures, 14.9 et ultérieures

db.r6id : classes d’instance Optimized Reads

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r6id.32xlarge	Non	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.4 et ultérieures, 14.9 et ultérieures
db.r6id.24xlarge	Non	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.4 et ultérieures, 14.9 et ultérieures

db.r8g — classes d'instance optimisées pour la mémoire alimentées par les processeurs Graviton4 AWS

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r8g.48xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r8g.24xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r8g.16xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r8g.12xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r8g.8xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r8g.4xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r8g.2xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r8g.xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r8g.large	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures

db.r8gd — classes d'instance de lecture optimisées alimentées par les processeurs Graviton4 AWS

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r8gd.48 x large	Non	17,4 et plus, 16,3 et plus, 15,7 et plus, 14,12 et plus
db.r8gd.24xlarge	Non	17,4 et plus, 16,3 et plus, 15,7 et plus, 14,12 et plus
db.r8gd.16 x large	Non	17,4 et plus, 16,3 et plus, 15,7 et plus, 14,12 et plus
db.r8gd.12xlarge	Non	17,4 et plus, 16,3 et plus, 15,7 et plus, 14,12 et plus
db.r8gd.8xlarge	Non	17,4 et plus, 16,3 et plus, 15,7 et plus, 14,12 et plus

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r8gd.4xlarge	Non	17,4 et plus, 16,3 et plus, 15,7 et plus, 14,12 et plus
db.r8gd.2xlarge	Non	17,4 et plus, 16,3 et plus, 15,7 et plus, 14,12 et plus
db.r8g.xlarge	Non	17,4 et plus, 16,3 et plus, 15,7 et plus, 14,12 et plus

db.r7g — classes d'instance optimisées pour la mémoire alimentées par les processeurs Graviton3 AWS

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r7g.16xlarge	2.12.0 et versions ultérieures, 3.03.1 et versions ultérieures	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.7 et ultérieures, 13.10 et ultérieures
db.r7g.12xlarge	2.12.0 et versions ultérieures, 3.03.1 et versions ultérieures	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.7 et ultérieures, 13.10 et ultérieures
db.r7g.8xlarge	2.12.0 et versions ultérieures, 3.03.1 et versions ultérieures	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.7 et ultérieures, 13.10 et ultérieures
db.r7g.4xlarge	2.12.0 et versions ultérieures, 3.03.1 et versions ultérieures	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.7 et ultérieures, 13.10 et ultérieures
db.r7g.2xlarge	2.12.0 et versions ultérieures, 3.03.1 et versions ultérieures	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.7 et ultérieures, 13.10 et ultérieures

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r7g.xlarge	2.12.0 et versions ultérieures, 3.03.1 et versions ultérieures	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.7 et ultérieures, 13.10 et ultérieures
db.r7g.large	2.12.0 et versions ultérieures, 3.03.1 et versions ultérieures	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.7 et ultérieures, 13.10 et ultérieures

db.r7i : classes d’instance à mémoire optimisée

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r7i.48xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r7i.24xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r7i.16xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r7i.12xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r7i.8xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r7i.4xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r7i.2xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r7i.xlarge	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures
db.r7i.large	3.08.0 et versions ultérieures	Versions 17.4 et ultérieures, 16.3 et ultérieures, 15.7 et ultérieures, 14.12 et ultérieures, 13.15 et ultérieures

db.r6g — classes d'instance optimisées pour la mémoire alimentées par les processeurs Graviton2 AWS

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r6g.16xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.r6g.12xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r6g.8xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.r6g.4xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.r6g.2xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.r6g.xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures
db.r6g.large	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.8 et ultérieures, 11.9, 11.12 et ultérieures

db.r6i : classes d’instance à mémoire optimisée

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r6i.32xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.5 et ultérieures et 12.9 et ultérieures
db.r6i.24xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.5 et ultérieures et 12.9 et ultérieures
db.r6i.16xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.5 et ultérieures et 12.9 et ultérieures
db.r6i.12xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.5 et ultérieures et 12.9 et ultérieures
db.r6i.8xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.5 et ultérieures et 12.9 et ultérieures
db.r6i.4xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.5 et ultérieures et 12.9 et ultérieures
db.r6i.2xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
		et ultérieures, 13.5 et ultérieures et 12.9 et ultérieures
db.r6i.xlarge	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.5 et ultérieures et 12.9 et ultérieures
db.r6i.large	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.5 et ultérieures et 12.9 et ultérieures

db.r5 : classes d’instance à mémoire optimisée

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r5.24xlarge	Toutes les versions actuellement disponibles	Toutes les versions actuellement disponibles
db.r5.16xlarge	Toutes les versions actuellement disponibles	Toutes les versions actuellement disponibles
db.r5.12xlarge	Toutes les versions actuellement disponibles	Toutes les versions actuellement disponibles
db.r5.8xlarge	Toutes les versions actuellement disponibles	Toutes les versions actuellement disponibles
db.r5.4xlarge	Toutes les versions actuellement disponibles	Toutes les versions actuellement disponibles
db.r5.2xlarge	Toutes les versions actuellement disponibles	Toutes les versions actuellement disponibles

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r5.xlarge	Toutes les versions actuellement disponibles	Toutes les versions actuellement disponibles
db.r5.large	Toutes les versions actuellement disponibles	Toutes les versions actuellement disponibles

db.r4 : classes d’instance à mémoire optimisée

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.r4.16xlarge	Toutes les versions 2.x ; non prises en charge dans les versions 3.x	Toutes les versions 11 et 12 prises en charge
db.r4.8xlarge	Toutes les versions 2.x ; non prises en charge dans les versions 3.x	Toutes les versions 11 et 12 prises en charge
db.r4.4xlarge	Toutes les versions 2.x ; non prises en charge dans les versions 3.x	Toutes les versions 11 et 12 prises en charge
db.r4.2xlarge	Toutes les versions 2.x ; non prises en charge dans les versions 3.x	Toutes les versions 11 et 12 prises en charge
db.r4.xlarge	Toutes les versions 2.x ; non prises en charge dans les versions 3.x	Toutes les versions 11 et 12 prises en charge
db.r4.large	Toutes les versions 2.x ; non prises en charge dans les versions 3.x	Toutes les versions 11 et 12 prises en charge

db.t4g — classes d'instance aux performances éclatantes alimentées par les processeurs Graviton2 AWS

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.t4g.2xlarge	Non	Non
db.t4g.xlarge	Non	Non
db.t4g.large	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.7 et ultérieures, 11.12 et ultérieures
db.t4g.medium	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.7 et ultérieures, 11.12 et ultérieures
db.t4g.small	Non	Non

db.t3 : classes d’instance à performances extensibles

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.t3.2xlarge	Non	Non
db.t3.xlarge	Non	Non
db.t3.large	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.7 et ultérieures, 11.12 et ultérieures

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.t3.medium	Toutes les versions actuellement disponibles	Versions 17.4 et ultérieures, 16.1 et ultérieures, 15.2 et ultérieures, 14.3 et ultérieures, 13.3 et ultérieures, 12.7 et ultérieures, 11.12 et ultérieures
db.t3.small	Toutes les versions 2.x ; non prises en charge dans les versions 3.x	Non
db.t3.micro	Non	Non

db.t2 : classes d’instance à performances extensibles

Classe d’instance	Aurora MySQL	Aurora PostgreSQL
db.t2.medium	Toutes les versions 2.x ; non prises en charge dans les versions 3.x	Non
db.t2.small	Toutes les versions 2.x ; non prises en charge dans les versions 3.x	Non

Détermination de la prise en charge des classes d’instance de base de données dans les Régions AWS

Pour déterminer les classes d’instance de base de données prises en charge par chaque moteur de base de données dans une Région AWS spécifique, vous pouvez adopter l’une des différentes approches. Vous pouvez utiliser AWS Management Console, la page Tarification d’Amazon RDS ou la commande AWS CLI describe-orderable-db-instance-options.

> **Note**
>
> Lorsque vous effectuez des opérations avec l'AWS Management Console, elle affiche automatiquement les classes d'instance de base de données prises en charge pour un moteur de base de données, une version de moteur de base de données et une Région AWS spécifique. Parmi les opérations que vous pouvez effectuer, citons la création et la modification d'une instance de base de données.

Table des matières

Utilisation de la page de tarification d'Amazon RDS pour déterminer la prise en charge des classes d'instance de base de données dans les Régions AWS

Vous pouvez utiliser la page Tarification d'Amazon Aurora pour connaître les classes d'instance de base de données prises en charge par chaque moteur de base de données dans une Région AWS spécifique.

Pour utiliser la page de tarification pour déterminer les classes d'instance de base de données prises en charge par chaque moteur dans une région

1. Accédez à la page Tarification d'Amazon Aurora.
2. Choisissez un moteur Amazon Aurora dans la section Calculateur de prix AWS.
3. Dans Choisir une région, choisissez une Région AWS.
4. Dans Option de configuration de cluster, choisissez une option de configuration.
5. Dans la section où figurent les instances compatibles, examinez les classes d'instance de base de données prises en charge.

6. (Facultatif) Choisissez d'autres options dans le calculateur, puis sélectionnez Enregistrer et afficher le récapitulatif ou Enregistrer et ajouter un service.

Utilisation de l'AWS CLI pour déterminer la prise en charge des classes d'instance de base de données dans les Régions AWS

Vous pouvez utiliser l'AWS CLI pour déterminer les classes d'instance de base de données qui sont prises en charge pour des moteurs de base de données spécifiques et des versions de moteur de base de données dans une Région AWS.

Pour utiliser les exemples d'AWS CLI suivants, saisissez des valeurs valides pour le moteur de base de données, la version du moteur de base de données, la classe d'instance de base de données et la Région AWS. Le tableau suivant présente les valeurs du moteur de base de données valides.

Nom du moteur	Valeur du moteur dans les commandes de la CLI	Plus d'informations sur les versions
Compatible MySQL 5.7 et Aurora 8.0	`aurora-mysql`	Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 2 et Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 3 dans Notes de mise à jour d'Aurora MySQL
Aurora PostgreSQL	`aurora-postgresql`	Notes de mise à jour d'Aurora PostgreSQL

Pour plus d'informations sur les noms d'Région AWS, consultez AWSRégions.

Les exemples suivants montrent comment déterminer la prise en charge des classes d'instance de base de données dans une Région AWS à l'aide de la commande AWS CLI describe-orderable-db-instance-options.

Rubriques

- Répertorier les classes d'instance de base de données prises en charge par une version de moteur de base de données spécifique dans une Région AWS

- Répertorier les versions de moteur de base de données qui prennent en charge une classe d'instance de base de données spécifique dans une Région AWS

Répertorier les classes d'instance de base de données prises en charge par une version de moteur de base de données spécifique dans une Région AWS

Pour répertorier les classes d'instance de base de données prises en charge par une version de moteur de base de données spécifique dans une Région AWS, exécutez la commande suivante.

Pour Linux, macOS ou Unix :

```
aws rds describe-orderable-db-instance-options --engine engine --engine-version version
 \
    --query "OrderableDBInstanceOptions[].
{DBInstanceClass:DBInstanceClass,SupportedEngineModes:SupportedEngineModes[0]}" \
    --output table \
    --region region
```

Pour Windows :

```
aws rds describe-orderable-db-instance-options --engine engine --engine-version version
 ^
    --query "OrderableDBInstanceOptions[].
{DBInstanceClass:DBInstanceClass,SupportedEngineModes:SupportedEngineModes[0]}" ^
    --output table ^
    --region region
```

La sortie affiche également les modes de moteur pris en charge pour chaque classe d'instance de base de données.

Par exemple, la commande suivante répertorie les classes d'instance de base de données prises en charge pour la version 13.6 du moteur de base de données Aurora PostgreSQL dans la région USA Est (Virginie du Nord).

Pour Linux, macOS ou Unix :

```
aws rds describe-orderable-db-instance-options --engine aurora-postgresql --engine-
version 15.3 \
    --query "OrderableDBInstanceOptions[].
{DBInstanceClass:DBInstanceClass,SupportedEngineModes:SupportedEngineModes[0]}" \
    --output table \
```

```
    --region us-east-1
```

Pour Windows :

```
aws rds describe-orderable-db-instance-options --engine aurora-postgresql --engine-
version 15.3 ^
    --query "OrderableDBInstanceOptions[].
{DBInstanceClass:DBInstanceClass,SupportedEngineModes:SupportedEngineModes[0]}"  ^
    --output table ^
    --region us-east-1
```

Répertorier les versions de moteur de base de données qui prennent en charge une classe d'instance de base de données spécifique dans une Région AWS

Pour répertorier les versions de moteur de base de données qui prennent en charge une classe d'instance de base de données spécifique dans une Région AWS, exécutez la commande suivante.

Pour Linux, macOS ou Unix :

```
aws rds describe-orderable-db-instance-options --engine engine --db-instance-
class DB_instance_class \
    --query "OrderableDBInstanceOptions[].
{EngineVersion:EngineVersion,SupportedEngineModes:SupportedEngineModes[0]}" \
    --output table \
    --region region
```

Pour Windows :

```
aws rds describe-orderable-db-instance-options --engine engine --db-instance-
class DB_instance_class ^
    --query "OrderableDBInstanceOptions[].
{EngineVersion:EngineVersion,SupportedEngineModes:SupportedEngineModes[0]}" ^
    --output table ^
    --region region
```

La sortie affiche également les modes de moteur pris en charge pour chaque version du moteur de base de données.

Par exemple, la commande suivante répertorie les versions du moteur de base de données du moteur de base de données Aurora PostgreSQL qui prennent en charge la classe d'instance de base de données db.r5.large dans la région USA Est (Virginie du Nord).

Pour Linux, macOS ou Unix :

```
aws rds describe-orderable-db-instance-options --engine aurora-postgresql --db-
instance-class db.r7g.large \
    --query "OrderableDBInstanceOptions[].
{EngineVersion:EngineVersion,SupportedEngineModes:SupportedEngineModes[0]}" \
    --output table \
    --region us-east-1
```

Pour Windows :

```
aws rds describe-orderable-db-instance-options --engine aurora-postgresql --db-
instance-class db.r7g.large ^
    --query "OrderableDBInstanceOptions[].
{EngineVersion:EngineVersion,SupportedEngineModes:SupportedEngineModes[0]}" ^
    --output table ^
    --region us-east-1
```

Spécifications matérielles pour les classes d’instance de base de données pour Aurora

Dans la table de cette section, vous trouverez des détails matériels sur les classes d’instance de base de données Amazon RDS pour Aurora.

Pour plus d’informations sur le moteur de base de données Aurora pris en charge pour chaque classe d’instance de base de données, consultez Moteurs de base de données pris en charge pour les classes d’instance de base de données.

Rubriques

- Terminologie matérielle pour les classes d’instance de base de données pour Aurora
- Spécifications matérielles pour les classes d’instance à mémoire optimisée
- Spécifications matérielles pour les classes d’instance à performances extensibles

Terminologie matérielle pour les classes d’instance de base de données pour Aurora

La terminologie suivante est utilisée pour décrire les spécifications matérielles des classes d’instance de base de données :

vCPU

Le nombre d'unités centrales virtuelles (CPUs). Un processeur virtuel est une unité de capacité que vous pouvez utiliser pour comparer les classes d'instance de base de données. Au lieu d'acheter ou de louer un processeur particulier pour l'utiliser pendant plusieurs mois ou plusieurs années, vous louez la capacité à l'heure. Notre but est de fournir une quantité constante et spécifique de capacité CPU, dans les limites du matériel sous-jacent.

ECU

Mesure relative de la puissance de traitement entière d'une EC2 instance Amazon. Pour permettre aux développeurs de comparer facilement la capacité du processeur entre différentes classes d'instances, nous avons défini une unité de EC2 calcul Amazon. La quantité de CPU allouée à une instance particulière est exprimée en termes de ces unités de EC2 calcul. Un ECU fournit actuellement une capacité de processeur équivalente à celle d'un processeur Opteron GHz 2007 ou Xeon 2007 1,0—1,2.

Mémoire (Gio)

Mémoire RAM, en gibioctets (Gio), allouée à l'instance de base de données. Il existe souvent un ratio cohérent entre la mémoire et le processeur virtuel. Citons, par exemple, la classe d'instance db.r4, qui a un ratio mémoire/processeur virtuel similaire à celui de la classe db.r5. Toutefois, dans la plupart des cas d'utilisation, la classe d'instance db.r5 fournit de meilleures performances, plus cohérentes, que la classe d'instance db.r4.

Taille max. Bande passante EBS (Mbit/s)

Bande passante EBS maximale en mégabits par seconde. Divisez cette valeur par 8 pour calculer le débit attendu en mégaoctets par seconde.

> **Note**
>
> Ce chiffre fait référence à la I/O bande passante pour le stockage local au sein de l'instance de base de données. Elle ne s'applique pas à la communication avec le volume du cluster Aurora.

Bande passante réseau

Vitesse du réseau par rapport à d'autres classes d'instance de base de données.

Pour plus d'informations sur l'utilisation CloudWatch des métriques Amazon pour surveiller le débit de votre instance de base de données Aurora, consultez Évaluation de l'utilisation de l'instance de base de données pour Aurora MySQL à l'aide des métriques Amazon CloudWatch et Évaluation de l'utilisation des instances de base de données pour Aurora CloudWatch PostgreSQL à l'aide de métriques.

Spécifications matérielles pour les classes d'instance à mémoire optimisée

Les tableaux suivants présentent les spécifications de calcul, de mémoire, de stockage et de bande passante pour les classes d'instance à mémoire optimisée.

db.x2g — classes d'instance optimisées pour la mémoire avec processeurs Graviton2 AWS

Classe d'instance	vCPU	ECU	Mémoire (Gio)	Stockage d'instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.x2g.16xlarge	64	—	1 024	Optimisé pour EBS uniquement	19 000	25
db.x2g.12xlarge	48	—	768	Optimisé pour EBS uniquement	14 250	20
db.x2g.8xlarge	32	—	512	Optimisé pour EBS uniquement	9 500	12
db.x2g.4xlarge	16	—	256	Optimisé pour EBS uniquement	4 750	Jusqu'à 10
db.x2g.2xlarge	8	—	128	Optimisé pour EBS uniquement	Jusqu'à 4 750	Jusqu'à 10

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.x2g.xlarge	4	—	64	Optimisé pour EBS uniquement	Jusqu’à 4 750	Jusqu’à 10
db.x2g.large	2	—	32	Optimisé pour EBS uniquement	Jusqu’à 4 750	Jusqu’à 10

db.r8gd — classes d'instance optimisées pour la mémoire alimentées par les processeurs Graviton4 et le stockage SSD AWS

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r8gd.48 x large	192	—	1536	6 disques NVMe SSD 1900	40 000	50
db.r8gd.24xlarge	96	—	768	3 disques NVMe SSD 1900	30 000	40
db.r8gd.16 x large	64	—	512	2 disques NVMe SSD 1900	20 000	30
db.r8gd.12xlarge	48	—	384	3 disques NVMe SSD 950	15 000	22,5

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r8gd.8xlarge	32	—	256	1 NVMe disque SSD de 1900	10 000	15
db.r8gd.4xlarge	16	—	128	1 NVMe disque SSD 950	Jusqu’à 10 000	Jusqu’à 15
db.r8gd.2xlarge	8	—	64	1 disque SSD 474 NVMe	Jusqu’à 10 000	Jusqu’à 15
db.r8gd.xlarge	4	—	32	1 NVMe disque SSD 237	Jusqu’à 10 000	Jusqu’à 12,5
db.r8gd.large	2	—	16	1 x 118 NVMe SSD	Jusqu’à 10 000	Jusqu’à 12,5

db.r8g — classes d'instance optimisées pour la mémoire alimentées par les processeurs Graviton4 AWS

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r8g.48xlarge	192	—	1536	Optimisé pour EBS uniquement	40 000	50

Classe d'instance	vCPU	ECU	Mémoire (Gio)	Stockage d'instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r8g.24xlarge	96	—	768	Optimisé pour EBS uniquement	30 000	40
db.r8g.16xlarge	64	—	512	Optimisé pour EBS uniquement	20 000	30
db.r8g.12xlarge	48	—	384	Optimisé pour EBS uniquement	15 000	22,5
db.r8g.8xlarge	32	—	256	Optimisé pour EBS uniquement	10 000	15
db.r8g.4xlarge	16	—	128	Optimisé pour EBS uniquement	Jusqu'à 10 000	Jusqu'à 15
db.r8g.2xlarge	8	—	64	Optimisé pour EBS uniquement	Jusqu'à 10 000	Jusqu'à 15
db.r8g.xlarge	4	—	32	Optimisé pour EBS uniquement	Jusqu'à 10 000	Jusqu'à 12,5
db.r8g.large	2	—	16	Optimisé pour EBS uniquement	Jusqu'à 10 000	Jusqu'à 12,5

db.r7i : classes d’instance à mémoire optimisée alimentées par des processeurs Intel Xeon Scalable de 4e génération

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r7i.48xlarge	192	—	1536	Optimisé pour EBS uniquement	40 000	50
db.r7i.24xlarge	96	—	768	Optimisé pour EBS uniquement	30 000	37,5
db.r7i.16xlarge	64	—	512	Optimisé pour EBS uniquement	20 000	25
db.r7i.12xlarge	48	—	384	Optimisé pour EBS uniquement	15 000	18,75
db.r7i.8xlarge	32	—	256	Optimisé pour EBS uniquement	10 000	12,5
db.r7i.4xlarge	16	—	128	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 12,5
db.r7i.2xlarge	8	—	64	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 12,5

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r7i.xlarge	4	—	32	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 12,5
db.r7i.large	2	—	16	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 12,5

db.r7g — classes d'instance optimisées pour la mémoire avec processeurs Graviton3 AWS

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r7g.16xlarge	64	—	512	Optimisé pour EBS uniquement	20 000	30
db.r7g.12xlarge	48	—	384	Optimisé pour EBS uniquement	15 000	22,5
db.r7g.8xlarge	32	—	256	Optimisé pour EBS uniquement	10 000	15
db.r7g.4xlarge	16	—	128	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 15

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r7g.2xlarge	8	—	64	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 15
db.r7g.xlarge	4	—	32	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 12,5
db.r7g.large	2	—	16	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 12,5

db.r6id : classe d’instance à mémoire optimisée avec processeurs Intel Xeon Scalable de 3e génération et stockage SSD

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r6id.32xlarge	128	—	1,024	SSD 4 x 1900 NVMe	40 000	50
db.r6id.24xlarge	96	—	768	SSD 4 x 1425 NVMe	30 000	37,5

db.r6gd — classes d'instance optimisées pour la mémoire avec processeurs Graviton2 et stockage SSD AWS

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r6g.16xlarge	64	—	512	2 disques NVMe SSD 1900	19 000	25
db.r6g.12xlarge	48	—	384	2 disques SSD 1425 NVMe	13 500	20
db.r6g.8xlarge	32	—	256	1 NVMe disque SSD de 1900	9 000	12
db.r6g.4xlarge	16	—	128	1 NVMe disque SSD 950	4 750	Jusqu’à 10
db.r6g.2xlarge	8	—	64	1 disque SSD 474 NVMe	Jusqu’à 4 750	Jusqu’à 10
db.r6gd.xlarge	4	—	32	1 NVMe disque SSD 237	Jusqu’à 4 750	Jusqu’à 10

db.r6g — classes d'instance optimisées pour la mémoire avec processeurs Graviton2 AWS

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r6g.16xlarge	64	—	512	Optimisé pour EBS uniquement	19 000	25
db.r6g.12xlarge	48	—	384	Optimisé pour EBS uniquement	13 500	20
db.r6g.8xlarge	32	—	256	Optimisé pour EBS uniquement	9 000	12
db.r6g.4xlarge	16	—	128	Optimisé pour EBS uniquement	4 750	Jusqu’à 10
db.r6g.2xlarge	8	—	64	Optimisé pour EBS uniquement	Jusqu’à 4 750	Jusqu’à 10
db.r6g.xlarge	4	—	32	Optimisé pour EBS uniquement	Jusqu’à 4 750	Jusqu’à 10
db.r6g.large	2	—	16	Optimisé pour EBS uniquement	Jusqu’à 4 750	Jusqu’à 10

db.r6id : classes d’instance à mémoire optimisée avec processeurs Intel Xeon Scalable de 3e génération

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r6i.32xlarge	128	—	1,024	Optimisé pour EBS uniquement	40 000	50
db.r6i.24xlarge	96	—	768	Optimisé pour EBS uniquement	30 000	37,5
db.r6i.16xlarge	64	—	512	Optimisé pour EBS uniquement	20 000	25
db.r6i.12xlarge	48	—	384	Optimisé pour EBS uniquement	15 000	18,75
db.r6i.8xlarge	32	—	256	Optimisé pour EBS uniquement	10 000	12,5
db.r6i.4xlarge	16	—	128	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 12,5
db.r6i.2xlarge	8	—	64	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 12,5
db.r6i.xlarge	4	—	32	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 12,5

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r6i.large	2	—	16	Optimisé pour EBS uniquement	Jusqu’à 10 000	Jusqu’à 12,5

db.r5 : classes d’instance à mémoire optimisée

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r5.24xlarge	96	347	768	Optimisé pour EBS uniquement	19 000	25
db.r5.16xlarge	64	264	512	Optimisé pour EBS uniquement	13 600	20
db.r5.12xlarge	48	173	384	Optimisé pour EBS uniquement	9 500	12
db.r5.8xlarge	32	132	256	Optimisé pour EBS uniquement	6 800	10
db.r5.4xlarge	16	71	128	Optimisé pour EBS uniquement	4 750	Jusqu’à 10

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r5.2xlarge	8	38	64	Optimisé pour EBS uniquement	Jusqu’à 4 750	Jusqu’à 10
db.r5.xlarge	4	19	32	Optimisé pour EBS uniquement	Jusqu’à 4 750	Jusqu’à 10
db.r5.large	2	10	16	Optimisé pour EBS uniquement	Jusqu’à 4 750	Jusqu’à 10

db.r4 : classes d’instance à mémoire optimisée avec processeurs Intel Xeon Scalable

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r4.16xlarge	64	195	488	Optimisé pour EBS uniquement	14 000	25
db.r4.8xlarge	32	99	244	Optimisé pour EBS uniquement	7 000	10
db.r4.4xlarge	16	53	122	Optimisé pour EBS uniquement	3 500	Jusqu’à 10

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.r4.2xlarge	8	27	61	Optimisé pour EBS uniquement	1 700	Jusqu’à 10
db.r4.xlarge	4	13,5	30,5	Optimisé pour EBS uniquement	850	Jusqu’à 10
db.r4.large	2	7	15,25	Optimisé pour EBS uniquement	425	Jusqu’à 10

Spécifications matérielles pour les classes d’instance à performances extensibles

Les tableaux suivants présentent les spécifications de calcul, de mémoire, de stockage et de bande passante pour les classes d’instance à performances extensibles.

db.t4g — classes d'instance aux performances éclatantes alimentées par les processeurs Graviton2 AWS

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.t4g.large	2	—	8	Optimisé pour EBS uniquement	Jusqu’à 2 780	Jusqu’à 5
db.t4g.medium	2	—	4	Optimisé pour EBS uniquement	Jusqu’à 2 085	Jusqu’à 5

db.t3 : classes d’instance à performances extensibles

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.t3.large	2	Variab	8	Optimisé pour EBS uniquement	Jusqu’à 2 048	Jusqu’à 5
db.t3.medium	2	Variab	4	Optimisé pour EBS uniquement	Jusqu’à 1 536	Jusqu’à 5
db.t3.small	2	Variab	2	Optimisé pour EBS uniquement	Jusqu’à 1 536	Jusqu’à 5

db.t2 : classes d’instance à performances extensibles

Classe d’instance	vCPU	ECU	Mémoire (Gio)	Stockage d’instances (Gio)	Taille max. Bande passante EBS (Mbit/s)	Bande passante du réseau (Gbit/s)
db.t2.medium	2	Variab	4	EBS uniquement	—	Modérée
db.t2.small	1	Variab	2	EBS uniquement	—	Faible

Stockage Amazon Aurora

Vous pouvez découvrir ci-après le sous-système de stockage Aurora. Aurora utilise une architecture de stockage distribuée et partagée qui est un facteur important en matière de performances, d'évolutivité et de fiabilité pour les clusters Aurora.

Rubriques

Présentation du stockage Amazon Aurora

Les données Aurora sont stockées dans le volume de cluster, qui est un seul volume virtuel et utilise des disques SSD. Un volume de cluster se compose de copies des données couvrant trois zones de disponibilité d'une même région AWS. Comme les données sont automatiquement répliquées à travers les zones de disponibilité, vos données sont hautement durables, avec une possibilité moindre de perte des données. Cette réplication garantit aussi que votre base de données est plus disponible pendant un basculement. En effet, les copies de données existent déjà dans les autres zones de disponibilité et continuent de traiter les demandes de données adressées aux instances de base de données de votre cluster de bases de données. Le volume de la réplication est indépendant du nombre d'instances DB de votre cluster.

Aurora utilise un stockage local séparé pour les fichiers temporaires non persistants. Il s'agit notamment des fichiers qui sont utilisés à des fins telles que le tri de grands jeux de données pendant le traitement des requêtes et la génération d'index. Pour plus d'informations, consultez Limites de stockage temporaires pour Aurora MySQL et Limites de stockage temporaires pour Aurora PostgreSQL.

Contenu du volume de cluster

Le volume de cluster Aurora contient toutes les données utilisateur, les objets de schéma et les métadonnées internes, telles que les tables système et le journal binaire. Par exemple, Aurora stocke, entre autres, les tables, les index, les objets BLOB et les procédures stockées d'un cluster Aurora dans un volume de cluster.

L’architecture de stockage partagée d’Aurora sépare vos données des instances de base de données du cluster. Par exemple, vous pouvez ajouter rapidement une instance de base de données, car Aurora n’effectue pas de nouvelle copie des données de la table. Au lieu de cela, l’instance de base de données se connecte au volume partagé qui contient déjà toutes les données. Vous pouvez supprimer une instance de base de données d’un cluster sans avoir à supprimer les données sous-jacentes du cluster. Aurora ne supprime les données que lorsque vous supprimez le cluster entier.

Configurations de stockage pour les clusters de bases de données Amazon Aurora

Amazon Aurora dispose de deux configurations de stockage pour les clusters de bases de données :

- Aurora I/O-Optimized : amélioration du rapport prix/performances et de la prévisibilité pour les applications gourmandes en E/S. Vous ne payez que pour l’utilisation et le stockage de vos clusters de bases de données, sans frais supplémentaires pour les opérations d’E/S en lecture et en écriture.

  Aurora I/O-Optimized est le meilleur choix lorsque vos dépenses d’E/S représentent 25 % ou plus de vos dépenses totales pour la base de données Aurora.

  Vous pouvez choisir Aurora I/O-Optimized lorsque vous créez ou modifiez un cluster de bases de données avec une version du moteur de base de données qui prend en charge la configuration du cluster Aurora I/O-Optimized. Vous pouvez passer du type Aurora I/O-Optimized au type Aurora Standard à tout moment.
- Aurora Standard : tarification rentable pour de nombreuses applications avec une utilisation modérée des E/S. Outre l’utilisation et le stockage de vos clusters de bases de données, vous payez également un tarif standard pour 1 million de demandes d’opérations d’E/S.

  Aurora Standard est le meilleur choix lorsque vos dépenses d’E/S représentent moins de 25 % de vos dépenses totales pour la base de données Aurora.

  Vous pouvez passer d’Aurora Standard à Aurora I/O-Optimized une fois tous les 30 jours. Lorsque vous basculez entre les options de stockage Aurora Standard et Aurora I/O-Optimized pour les instances de base de données non basées sur NVMe, il n’y a aucune durée d’indisponibilité. Toutefois, pour les instances de base de données basées sur NVMe, basculer entre les options de stockage Aurora I/O-Optimized et Aurora Standard nécessite le redémarrage du moteur de base de données, ce qui peut entraîner une brève durée d’indisponibilité.

Pour obtenir des informations sur la prise en charge de la versions et de la Région AWS, consultez Régions et moteurs de base de données Aurora pris en charge pour les configurations de stockage en cluster.

Pour plus d'informations sur la tarification des configurations de stockage Amazon Aurora, consultez Tarification d'Amazon Aurora.

Pour obtenir des informations sur le choix de la configuration de stockage lors de la création d'un cluster de bases de données, consultez Création d'un cluster de bases de données. Pour obtenir des informations sur la modification de la configuration de stockage pour un cluster de bases de données, consultez Paramètres pour Amazon Aurora.

Redimensionnement automatique du stockage Aurora

Les volumes de cluster Aurora croissent automatiquement au fur et à mesure que la quantité de données de votre base de données augmente. Pour plus d'informations sur les tailles de volume maximales des clusters Aurora pour chaque version de moteur, consultez Limites de taille Amazon Aurora. Cette mise à l'échelle automatique du stockage est associée à un sous-système de stockage hautement distribué et à hautes performances. Ceci fait d'Aurora un bon choix pour vos données importantes d'entreprise lorsque vos objectifs principaux sont la fiabilité et la haute disponibilité.

Pour afficher le statut du volume, reportez-vous à la section Affichage du statut du volume pour un cluster de base de données Aurora MySQL ou Affichage du statut du volume pour un cluster de bases de données Aurora PostgreSQL. Pour trouver des moyens d'équilibrer les coûts de stockage par rapport aux autres priorités, Dimensionnement du stockage décrit comment surveiller les métriques Amazon Aurora `AuroraVolumeBytesLeftTotal` et `VolumeBytesUsed` dans CloudWatch.

Lorsque des données Aurora sont supprimées, l'espace alloué à ces données est libéré. Parmi les exemples de suppression de données, on peut citer la suppression ou la troncation d'une table. Cette réduction automatique de l'utilisation du stockage vous aide à minimiser les frais de stockage.

> **Note**
>
> Les limites de stockage et le comportement de redimensionnement dynamique présentés ici s'appliquent aux tables persistantes et aux autres données stockées dans le volume de cluster.
>
> Pour Aurora PostgreSQL, les données de table temporaires sont stockées dans l'instance de base de données locale.

Pour Aurora MySQL version 2, les données de tables temporaires sont stockées par défaut dans le volume du cluster pour les instances d'écriture et dans le stockage local pour les instances de lecture. Pour plus d'informations, consultez Moteur de stockage pour des tables temporaires sur disque.
Pour Aurora MySQL version 3, les données de tables temporaires sont stockées dans l'instance de base de données locale ou dans le volume du cluster. Pour plus d'informations, consultez Nouveau comportement de table temporaire dans Aurora MySQL version 3.
La taille maximale des tables temporaires résidant dans le stockage local est limitée par la taille de stockage local maximale de l'instance de base de données. La taille de stockage local dépend de la classe d'instance que vous utilisez. Pour plus d'informations, consultez Limites de stockage temporaires pour Aurora MySQL et Limites de stockage temporaires pour Aurora PostgreSQL.

Certaines fonctions de stockage, telles que la taille maximale d'un volume de cluster et le redimensionnement automatique lorsque des données sont supprimées, dépendent de la version Aurora de votre cluster. Pour plus d'informations, consultez Dimensionnement du stockage. Vous pouvez également apprendre à éviter les problèmes de stockage et à surveiller le stockage alloué et l'espace libre dans votre cluster.

Facturation du stockage des données Aurora

Même si un volume de cluster Aurora peut atteindre 256 téraoctets (Tio) pour des versions de moteur spécifiques, vous n'êtes facturé que pour l'espace que vous utilisez dans un volume de cluster Aurora. Dans les versions antérieures d'Aurora, le volume de cluster pouvait réutiliser l'espace libéré lorsque vous supprimiez des données, mais l'espace de stockage alloué ne diminuait jamais. Désormais, lorsque des données Aurora sont supprimées (par exemple, en supprimant une table ou une base de données), l'espace alloué global diminue d'un montant comparable. Ainsi, vous pouvez réduire les frais de stockage en supprimant des tables, des index, des bases de données, etc. dont vous n'avez plus besoin.

Tip

Pour les versions antérieures sans fonction de redimensionnement dynamique, la réinitialisation de l'utilisation du stockage pour un cluster impliquait la réalisation d'un vidage logique et la restauration vers un nouveau cluster. Cette opération peut prendre beaucoup de temps pour un volume important de données. Le cas échéant, envisagez de mettre à

niveau votre cluster vers une version qui prend en charge le redimensionnement du volume dynamique.

Pour plus d'informations sur les versions d'Aurora qui prennent en charge le redimensionnement dynamique et sur la façon de réduire les frais de stockage en surveillant l'utilisation du stockage pour votre cluster, consultez Dimensionnement du stockage. Pour plus d'informations sur la facturation du stockage de sauvegarde Aurora, consultez Comprendre l'utilisation du stockage de sauvegarde Amazon Aurora. Pour obtenir des informations sur la tarification du stockage de données Aurora, consultez la section Tarification de Amazon RDS for Aurora.

Fiabilité Amazon Aurora

Aurora est conçu pour être fiable, durable et tolérant aux pannes. Vous pouvez définir l'architecture de votre cluster de bases de données Aurora afin d'améliorer la disponibilité en permettant des actions telles que l'ajout de réplicas Aurora et leur placement dans différentes zones de disponibilité. En outre, Aurora inclut plusieurs fonctions automatiques qui garantissent sa fiabilité en tant que solution de base de données.

Rubriques

- Réparation automatique du stockage
- Cache de page durable
- Reprise après un redémarrage imprévu

Réparation automatique du stockage

Comme Aurora gère plusieurs copies de vos données dans trois zones de disponibilité, le risque de perdre des données suite à une défaillance de disque est considérablement limité. Aurora détecte automatiquement les défaillances au niveau des volumes de disque qui composent le volume de cluster. En cas de défaillance d'un segment d'un volume disque, Aurora répare immédiatement le segment. Quand Aurora répare le segment disque, il utilise les données des autres volumes qui composent le volume de cluster pour garantir que les données du segment réparé sont actives. En conséquence, Aurora empêche la perte de données et réduit la nécessité d'exécuter une restauration à un instant dans le passé pour récupérer à partir d'une défaillance disque.

Cache de page durable

Dans Aurora, le cache de page de chaque instance de base de données est géré dans un processus distinct de la base de données, qui lui permet de « survivre » indépendamment de la base de données. (Le cache de page est également appelé pool de mémoires tampons InnoDB sur Aurora MySQL et cache de tampon sur Aurora PostgreSQL.)

Dans l'éventualité peu probable d'une défaillance de la base de données, le cache de page reste en mémoire, ce qui maintient les pages de données actuelles à l'état pré-initialisé dans le cache de page au redémarrage de la base de données. Cette approche fournit un gain de performance, car les requêtes initiales n'ont pas besoin d'exécuter d'opérations d'E/S en lecture pour préparer le cache de page.

Pour Aurora MySQL, le comportement du cache de page lors du redémarrage et du basculement est le suivant :

- Vous pouvez redémarrer l'instance d'enregistreur sans redémarrer les instances de lecteur.
  - Si les instances de lecteur ne redémarrent pas lors du redémarrage de l'instance d'enregistreur, elles ne perdent pas leurs caches de pages.
  - Si les instances de lecteur redémarrent lors du redémarrage de l'instance d'enregistreur, elles perdent leurs caches de pages.
- Lorsqu'une instance de lecteur redémarre, les caches de pages survivent à la fois sur les instances d'enregistreur et de lecteur.
- Lorsque le cluster de bases de données bascule, l'effet est similaire au redémarrage d'une instance d'enregistreur. Sur la nouvelle instance d'enregistreur (auparavant l'instance de lecteur), le cache de page survit, mais sur l'instance de lecteur (auparavant l'instance d'enregistreur), le cache de page ne survit pas.

Pour Aurora PostgreSQL, vous pouvez utiliser la gestion du cache de cluster pour préserver le cache de page d'une instance de lecteur spécifique qui devient l'instance d'enregistreur après basculement. Pour plus d'informations, consultez Récupération rapide après basculement avec la gestion des caches de clusters pour Aurora PostgreSQL.

Reprise après un redémarrage imprévu

Aurora est conçu pour reprendre presque instantanément après un redémarrage imprévu, et continuer de servir les données de votre application sans le journal binaire. Aurora reprend de

manière asynchrone sur des threads parallèles de telle sorte que votre base de données est ouverte et immédiatement accessible après un redémarrage imprévu.

Pour plus d'informations, consultez Tolérance aux pannes pour un cluster de bases de données Aurora et Optimisations destinées à réduire le temps de redémarrage de la base de données.

Les points ci-dessous sont à prendre en considération pour la journalisation binaire et la reprise d'Aurora MySQL après un redémarrage imprévu :

- L'activation de la journalisation binaire dans Aurora affecte directement le délai de reprise après un redémarrage imprévu, car elle force l'instance de base de données à récupérer les journaux binaires.
- Le type de journalisation binaire utilisé affecte la taille et l'efficacité de la journalisation. Pour un même niveau d'activité de la base de données, certains formats consignent plus d'informations que d'autres dans les journaux binaires. Les valeurs suivantes du paramètre `binlog_format` entraînent des quantités différentes de données de journalisation :
  - `ROW` – Maximum de données de journalisation
  - `STATEMENT` – Minimum de données de journalisation
  - `MIXED` – Quantité modérée de données de journalisation qui représente généralement la meilleure option de performances et d'intégrité des données

  La quantité de données consignée dans les journaux binaires affecte la durée de récupération. Si une plus grande quantité de données est consignée dans les journaux binaires, l'instance de base de données doit traiter plus de données au cours de la récupération, ce qui augmente la durée de récupération.
- Pour réduire la surcharge de calcul et améliorer les temps de restauration grâce à la journalisation binaire, vous pouvez utiliser un journal binaire amélioré. Celui-ci améliore le temps de restauration de la base de données de jusqu'à 99 %. Pour plus d'informations, consultez Configuration du binlog amélioré pour Aurora MySQL.
- Aurora n'a pas besoin des journaux binaires pour répliquer les données au sein d'un cluster de bases de données ou réaliser une restauration à un instant donné.
- Si vous n'avez pas besoin du journal binaire pour la réplication externe (ou un flux de journal binaire externe), nous vous recommandons de définir le paramètre `binlog_format` sur `OFF` pour désactiver la journalisation binaire. Cela a pour effet de réduire le temps de récupération.

Pour plus d'informations sur la réplication et la journalisation binaire Aurora, consultez Réplication avec Amazon Aurora. Pour plus d'informations sur les implications des différents types de réplication MySQL, consultez Advantages and Disadvantages of Statement-Based and Row-Based Replication dans la documentation de MySQL.

Sécurité Amazon Aurora

La sécurité d'Amazon Aurora est gérée à trois niveaux :

- Pour contrôler les personnes autorisées à exécuter des opérations de gestion Amazon RDS sur des clusters et des instances de base de données Aurora, vous utilisez Gestion des identités et des accès AWS (IAM). Lorsque vous vous connectez à AWS à l'aide des informations d'identification IAM, votre compte AWS doit disposer des politiques IAM qui accordent les autorisations requises pour exécuter les opérations de gestion Amazon RDS. Pour plus d'informations, consultez Identity and Access Management pour Amazon Aurora.

  Si vous utilisez un compte IAM pour accéder à la console Amazon RDS, vous devez d'abord vous connecter à la AWS Management Console avec vos informations d'identification, puis accéder à la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds.
- Les clusters de bases de données Aurora doivent être créés dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC. Pour contrôler les appareils et les instances Amazon EC2 qui peuvent ouvrir des connexions au point de terminaison et au port de l'instance de base de données pour les clusters de bases de données Aurora d'un VPC, vous utilisez un groupe de sécurité VPC. Avec ces points de terminaison et les connexions de port, vous pouvez utiliser TLS/SSL (Transport Layer Security/Secure Sockets Layer). En outre, les règles de pare-feu de votre entreprise peuvent contrôler si les appareils en cours d'exécution dans votre entreprise peuvent ouvrir des connexions à une instance de base de données. Pour plus d'informations sur les VPC, consultez Amazon VPC et Amazon Aurora.
- Pour authentifier les connexions et les autorisations d'un cluster de bases de données Amazon Aurora, vous pouvez adopter l'une des approches suivantes, ou les combiner.
  - Vous pouvez adopter la même approche qu'avec une instance de base de données autonome de MySQL ou PostgreSQL.

    Les techniques d'authentification des connexions et des autorisations pour les instances de base de données autonomes de MySQL ou PostgreSQL, telles que l'utilisation des commandes SQL ou la modification des tables de schéma de base de données, fonctionnent également avec

Aurora. Pour plus d'informations, consultez Sécurité avec Amazon Aurora MySQL ou Sécurité avec Amazon Aurora PostgreSQL.

- Vous pouvez utiliser l'authentification de base de données IAM.

  Avec l'authentification de base de données IAM, vous vous authentifiez auprès de votre cluster de bases de données Aurora en utilisant un utilisateur ou un rôle IAM et un jeton d'authentification. Un jeton d'authentification est une valeur unique qui est générée à l'aide du processus de signature Signature Version 4. L'authentification de base de données IAM vous permet d'utiliser les mêmes informations d'identification pour contrôler l'accès à vos ressources AWS et à vos bases de données. Pour plus d'informations, consultez Authentification de base de données IAM.

- Vous pouvez utiliser l'authentification Kerberos pour Aurora PostgreSQL et Aurora MySQL.

  Vous pouvez utiliser Kerberos pour authentifier les utilisateurs lorsqu'ils se connectent à votre cluster de bases de données Aurora PostgreSQL et Aurora MySQL. Dans ce cas, votre cluster de bases de données utilise AWS Directory Service for Microsoft Active Directory pour activer l'authentification Kerberos. AWS Directory Service for Microsoft Active Directory est également appelé AWS Managed Microsoft AD. Vous pouvez gagner du temps et de l'argent en conservant toutes les informations d'identification dans le même annuaire. Vous avez un endroit centralisé de stockage et de gestion des informations d'identification pour plusieurs clusters de bases de données. L'utilisation d'un annuaire peut également améliorer votre profil de sécurité global. Pour plus d'informations, consultez Utilisation de l'authentification Kerberos avec Aurora PostgreSQL et Utilisation de l'authentification Kerberos pour Aurora MySQL.

Pour plus d'informations sur la configuration de la sécurité, consultez Sécurité dans Amazon Aurora.

Utilisation de SSL avec les clusters de bases de données Aurora

Les clusters de bases de données Amazon Aurora prennent en charge les connexions Secure Sockets Layer (SSL) à partir des applications utilisant le même processus et la même clé publique que les instances de base de données Amazon RDS. Pour plus d'informations, consultez Sécurité avec Amazon Aurora MySQL, Sécurité avec Amazon Aurora PostgreSQL ou Utilisation de TLS/SSL avec Aurora Serverless v1.

Haute disponibilité pour Amazon Aurora

L'architecture Amazon Aurora implique une séparation entre le stockage et le calcul. Aurora comporte des fonctions à haute disponibilité qui s'appliquent aux données de votre cluster de bases de données. Les données sont en sécurité, même si certaines ou la totalité des instances de base de données du cluster deviennent indisponibles. D'autres fonctionnalités à haute disponibilité s'appliquent aux instances de base de données. Ces fonctionnalités permettent d'être sûr qu'une ou plusieurs instances de base de données sont prêtes à gérer les demandes adressées à la base de données à partir de votre application.

Rubriques

Haute disponibilité pour les données Aurora

Aurora stocke les copies des données d'un cluster de bases de données dans plusieurs zones de disponibilité d'une même Région AWS. Aurora stocke ces copies indépendamment du fait que les instances dans le cluster de bases de données recouvrent ou pas plusieurs zones de disponibilité. Pour plus d'informations sur Aurora, consultez Gestion d'un cluster de bases de données Amazon Aurora.

Lorsque des données sont écrites dans l'instance de base de données principale, Aurora réplique de façon synchronisée les données entre les zones de disponibilité sur six nœuds de stockage associés au volume de votre cluster. Cela permet de bénéficier de la redondance des données, d'éliminer les figements d'I/O et de minimiser les pics de latence pendant les sauvegardes du système. L'exécution d'une instance de base de données avec la haute disponibilité peut améliorer la disponibilité pendant la maintenance planifiée du système et contribuer à protéger vos bases de données contre toute défaillance ou perturbation d'une zone de disponibilité. Pour plus d'informations sur les zones de disponibilité, consultez Régions et zones de disponibilité.

Haute disponibilité pour les instances de base de données Aurora

Une fois que vous avez créé l'instance principale (d'écriture), vous pouvez créer jusqu'à 15 réplicas Aurora en lecture seule. Les réplicas Aurora sont aussi appelés instances de lecteur. Ils utilisent la réplication asynchrone pour assurer une haute disponibilité sans nuire aux performances de l'instance principale.

Au cours des opérations quotidiennes, vous pouvez décharger une partie du travail pour les applications à forte intensité de lecture en utilisant les instances du lecteur pour traiter les requêtes `SELECT`. Lorsqu'un problème affecte l'instance principale, l'une de ces instances de lecteur prend le relais en tant qu'instance principale. Ce mécanisme est connu sous le nom de basculement. De nombreuses fonctionnalités Aurora s'appliquent au mécanisme de basculement. Par exemple, Aurora détecte les problèmes de base de données et active automatiquement le mécanisme de basculement si nécessaire. Aurora disposent également de fonctions qui réduisent le temps de basculement. Ainsi, le temps pendant lequel la base de données n'est pas disponible pour l'écriture pendant un basculement est réduit.

Aurora est conçu pour optimiser la vitesse de restauration. Le moyen de restauration le plus rapide consiste souvent à basculer vers la même instance de base de données ou à la redémarrer. Le redémarrage est plus rapide et implique moins de frais que le basculement.

Pour utiliser une chaîne de connexion qui reste identique même lorsqu'un basculement favorise une nouvelle instance principale, vous vous connectez au point de terminaison du cluster. Le point de terminaison du cluster représente toujours l'instance principale actuelle dans le cluster. Pour plus d'informations sur le point de terminaison du cluster, consultez Connexions de point de terminaison Amazon Aurora.

> **Tip**
>
> Dans chaque Région AWS, les zones de disponibilité représentent des emplacements distincts les uns des autres pour assurer l'isolement en cas de panne. Nous vous recommandons de répartir l'instance principale et les instances de lecteur de votre cluster de bases de données sur plusieurs zones de disponibilité afin d'améliorer la disponibilité du cluster. De cette façon, un problème qui affecte toute une zone de disponibilité ne provoquera aucune panne du cluster.
> Vous pouvez configurer un cluster de bases de données multi-AZ en faisant un choix simple lors de sa création. Vous pouvez utiliser l'AWS Management Console, l'AWS CLI ou l'API Amazon RDS. Vous pouvez également convertir un cluster Aurora en cluster de bases de

données multi-AZ en ajoutant une nouvelle instance de base de données de lecteur et en spécifiant une autre zone de disponibilité.

Haute disponibilité dans les régions AWS avec des bases de données Aurora globales

Pour une haute disponibilité dans plusieurs Régions AWS, vous pouvez configurer des bases de données Aurora globales. Une base de données Aurora globale couvre plusieurs Régions AWS, ce qui assure une faible latence des lectures globales et la reprise après sinistre en cas de panne dans une Région AWS. Aurora réplique automatiquement toutes les données et mises à jour de la Région AWS principale vers chacune des régions secondaires. Pour plus d'informations, consultez Utilisation d'Amazon Aurora Global Database.

Tolérance aux pannes pour un cluster de bases de données Aurora

Un cluster de bases de données Aurora est tolérant aux pannes par définition. Le volume de cluster couvre plusieurs zones de disponibilité d'une même Région AWS et chacune d'elles contient une copie des données du volume de cluster. Cette fonctionnalité signifie que votre cluster de bases de données peut tolérer une défaillance d'une zone de disponibilité sans perte de données et uniquement une brève interruption de service.

En cas de défaillance de l'instance principale d'un cluster de bases de données, Aurora bascule automatiquement vers une nouvelle instance principale de l'une des deux façons suivantes :

- Par la promotion d'un réplica Aurora existant vers la nouvelle instance principale
- En créant une autre instance principale

Si le cluster de bases de données possède un ou plusieurs réplicas Aurora, un réplica Aurora est promu vers l'instance principale lors d'un événement d'échec. Un événement d'échec se traduit par une brève interruption, pendant laquelle les opérations de lecture et d'écriture échouent avec une exception. Cependant, le service est généralement restauré en moins de 60 secondes, et souvent en moins de 30 secondes. Pour augmenter la disponibilité de votre cluster de bases de données, nous vous recommandons de créer au moins un ou plusieurs réplicas Aurora dans deux zones de disponibilité ou plus.

> **Tip**
>
> Dans Aurora MySQL, vous pouvez améliorer la disponibilité lors d'un basculement en utilisant plusieurs instances de base de données de lecteur dans un cluster. Dans Aurora MySQL, Aurora redémarre uniquement l'instance de base de données d'enregistreur et l'instance de lecteur vers laquelle le basculement s'effectue. D'autres instances de lecteur dans le cluster restent disponibles au cours d'un basculement pour continuer le traitement des requêtes par le biais de connexions au point de terminaison de lecteur.
> Vous pouvez également améliorer la disponibilité lors d'un basculement à l'aide de RDS Proxy avec votre cluster de bases de données Aurora. Pour plus d'informations, consultez Haute disponibilité avec Proxy Amazon RDS.

Vous pouvez personnaliser l'ordre dans lequel vos réplicas Aurora sont promus vers l'instance principale après un échec, en affectant à chaque réplica une priorité. Les priorités s'étendent de la valeur 0 pour la plus haute priorité à la valeur 15 pour la plus basse priorité. Si l'instance principale échoue, Amazon RDS promeut le réplica Aurora ayant la priorité la plus élevée vers la nouvelle instance principale. Vous pouvez modifier la priorité d'un réplica Aurora à tout moment. La modification de la priorité ne déclenche pas un basculement.

Plusieurs réplicas Aurora peuvent partager la même priorité, ce qui se traduit par des niveaux de promotion. Si deux réplicas Aurora ou plus partagent la même priorité, Amazon RDS promeut le réplica le plus grand en taille. Si deux réplicas Aurora ou plus partagent les mêmes priorité et taille, Amazon RDS promeut un réplica arbitraire du même niveau de promotion.

> **Note**
>
> Plusieurs facteurs interviennent dans l'identification d'une cible de basculement. Après cinq tentatives de basculement infructueuses, les niveaux de promotion ne sont plus pris en compte.

Si le cluster de bases de données ne contient aucun réplica Aurora, l'instance principale est recréée dans la même AZ pendant un événement d'échec. Un événement d'échec se traduit par une interruption, pendant laquelle les opérations de lecture et d'écriture échouent avec une exception. Le service est rétabli quand la nouvelle instance principale est créée, ce qui prend généralement moins de 10 minutes. La promotion d'un réplica Aurora vers l'instance principale est beaucoup plus rapide que la création d'une nouvelle instance principale.

Supposons que l'instance principale de votre cluster soit indisponible en raison d'une panne affectant une zone de disponibilité entière. Dans ce cas, la façon de mettre en ligne une nouvelle instance principale dépend du fait que votre cluster utilise ou non une configuration Multi-AZ :

- Si votre cluster alloué ou Aurora Serverless v2 contient des instances de lecteur dans d'autres zones de disponibilité, Aurora utilise le mécanisme de basculement pour promouvoir l'une de ces instances de lecteur en tant que nouvelle instance principale.
- Si votre cluster alloué ou Aurora Serverless v2 contient une instance de base de données unique, ou si l'instance principale et toutes les instances de lecteur se trouvent dans la même zone de disponibilité, veillez à créer manuellement une ou plusieurs instances de base de données dans une autre zone de disponibilité.
- Si votre cluster utilise Aurora Serverless v1, Aurora crée automatiquement une instance de base de données dans une autre zone de disponibilité. Toutefois, ce processus implique un remplacement d'hôte et prend donc plus de temps qu'un basculement.

> **Note**
>
> Amazon Aurora prend aussi en charge la réplication avec une base de données MySQL externe ou une instance de base de données MySQL RDS. Pour plus d'informations, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

Haute disponibilité avec Proxy Amazon RDS

Avec RDS Proxy, vous pouvez créer des applications capables de tolérer de manière transparente les pannes de base de données sans avoir à écrire de code de gestion des défaillances complexe. Le proxy achemine automatiquement le trafic vers une nouvelle instance de base de données tout en préservant les connexions aux applications. Il contourne également les caches du système de nom de domaine (DNS) afin de réduire les temps de basculement jusqu'à 66 % pour les bases de données Aurora multi-AZ. Pour plus d'informations, consultez Proxy Amazon RDS pour Aurora.

Réplication avec Amazon Aurora

Aurora propose plusieurs options de réplication. Chaque cluster de bases de données Aurora dispose d'une réplication intégrée entre plusieurs instances dans le même cluster. Vous pouvez

également configurer la réplication avec votre cluster Aurora en tant que source ou cible. Lorsque vous répliquez des données dans ou hors d'un cluster Aurora, vous pouvez choisir entre des fonctionnalités intégrées telles que les bases de données Aurora globales ou les mécanismes de réplication traditionnels pour les moteurs de base de données MySQL ou PostgreSQL. Vous pouvez déterminer les options appropriées en recherchant un équilibre entre haute disponibilité, commodité et performances selon vos besoins. Les sections suivantes expliquent comment et quand choisir chaque technique.

Rubriques

Réplicas Aurora

Lorsque vous créez une deuxième, troisième, etc. instances de base de données dans un cluster de bases de données Aurora provisionné, Aurora configure automatiquement la réplication à partir de l'instance de base de données de scripteur vers toutes les autres instances. Ces autres instances sont en lecture seule et sont appelées des réplicas Aurora. Nous les désignons également comme des instances de lecteur lorsque vous nous parlons de la façon dont vous pouvez combiner des instances de scripteur et de lecteur au sein d'un cluster.

Les réplicas Aurora ont deux objectifs principaux. Vous pouvez émettre des requêtes vers ces derniers pour mettre à l'échelle les opérations de lecture de votre application. Cela se fait généralement en se connectant au point de terminaison du lecteur du cluster. De cette façon, Aurora peut répartir la charge pour les connexions en lecture seule sur l'ensemble des réplicas Aurora disponibles dans le cluster. Les réplicas Aurora aident également à augmenter la disponibilité. Si l'instance de scripteur dans un cluster devient indisponible, Aurora promeut automatiquement l'une des instances de lecteur pour qu'elle prenne sa place en tant que nouveau scripteur.

Un cluster de bases de données Aurora peut contenir jusqu'à 15 réplicas Aurora. Les réplicas Aurora peuvent être répartis entre les zones de disponibilité couvertes par un cluster de bases de données au sein d'une même région AWS.

Les données de votre cluster de bases de données possèdent leurs propres fonctions de haute disponibilité et de fiabilité, indépendantes des instances de base de données du cluster. Si vous n'êtes pas familier avec les fonctionnalités de stockage d'Aurora, reportez-vous à la section

Présentation du stockage Amazon Aurora. Le volume de cluster de bases de données est physiquement composé de plusieurs copies des données du cluster de bases de données. Les données du volume de cluster sont représentées comme un seul volume logique à l'instance principale et aux réplicas Aurora du cluster de bases de données.

En conséquence, tous les réplicas Aurora renvoient les mêmes données pour les résultats des requêtes avec un retard de réplica minimal. Ce retard est généralement bien inférieur à 100 ms après que l'instance principale a écrit une mise à jour. Le retard de réplica varie en fonction de la fréquence de modification de la base de données. Autrement dit, pendant les périodes où une importante quantité d'opérations d'écriture se produit pour la base de données, il se peut que vous constatiez un retard accru du réplica.

> **Note**
>
> Le réplica Aurora redémarre automatiquement lorsqu'il perd la communication avec l'instance de base de données d'enregistreur pendant plus de 60 secondes dans les versions d'Aurora PostgreSQL suivantes :
>
> - Versions 14.6 et antérieures
> - Versions 13.9 et antérieures
> - Versions 12.13 et antérieures
> - Toutes les versions d'Aurora PostgreSQL 11
>
> Grâce à la fonctionnalité de disponibilité en lecture, les réplicas Aurora ne redémarrent pas automatiquement. Pour plus d'informations sur la fonctionnalité de disponibilité en lecture et les versions dans lesquelles elle est applicable, consultez Amélioration de la disponibilité en lecture des réplicas Aurora.

Les réplicas Aurora fonctionnement parfaitement pour le dimensionnement en lecture, car ils sont intégralement dédiés aux opérations de lecture de votre volume de cluster. Les opérations d'écriture sont gérées par l'instance principale. Sachant que le volume de cluster est partagé entre toutes les instances de base de données de votre cluster de bases de données, la réplication d'une copie des données de chaque réplica Aurora nécessite peu d'efforts.

Pour accroître la disponibilité, vous pouvez utiliser les réplicas Aurora comme cibles de basculement. En d'autres termes, si l'instance principale est défaillante, un réplica Aurora peut être promu instance

principale. Il y a une brève interruption, pendant laquelle les demandes de lecture et d'écriture adressées à l'instance principale échouent en renvoyant une exception.

La promotion d'un réplica Aurora par basculement est beaucoup plus rapide que la recréation de l'instance principale. Si votre cluster de bases de données Aurora ne comporte pas de réplicas Aurora, il ne sera pas disponible pendant que votre instance de base de données récupérera de la défaillance.

Lors du basculement, certains réplicas Aurora peuvent être redémarrés, en fonction de la version du moteur de base de données. Par exemple, dans Aurora MySQL, Aurora redémarre uniquement l'instance de base de données d'enregistreur et la cible de basculement lors d'un basculement. Pour plus d'informations sur le comportement de redémarrage des différentes versions du moteur de base de données Aurora, consultez Redémarrage d'un cluster de bases de données Amazon Aurora ou d'une instance de base de données Amazon Aurora. Pour obtenir des informations sur ce qu'il advient des caches de pages lors du redémarrage ou du basculement, consultez Cache de page durable.

Pour les scénarios de haute disponibilité, il est recommandé de créer un ou plusieurs réplicas Aurora. Ceux-ci doivent avoir la même classe d'instance de base de données que l'instance principale, et se trouver dans des zones de disponibilité différentes de votre cluster de bases de données Aurora. Pour plus d'informations sur les réplicas Aurora comme cibles de basculement, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.

Vous ne pouvez pas créer de réplica Aurora chiffré pour un cluster de bases de données Aurora non chiffré. Vous ne pouvez pas créer de réplica Aurora non chiffré pour un cluster de bases de données Aurora chiffré.

> **Tip**
>
> Vous pouvez utiliser des réplicas Aurora au sein d'un cluster Aurora comme seule forme de réplication pour maintenir vos données hautement disponibles. Vous pouvez également combiner la réplication Aurora intégrée avec les autres types de réplication. Cela peut vous aider à assurer un niveau de disponibilité et de distribution géographique de vos données encore supérieur

Pour plus de détails sur la création d'un réplica Aurora, consultez Ajout de réplicas Aurora à un cluster de bases de données.

Réplication avec Aurora MySQL

En plus des réplicas Aurora, Aurora MySQL propose les options de réplication suivantes :

- clusters de bases de données Aurora MySQL dans différentes régions AWS.
  - Vous pouvez répliquer des données sur plusieurs régions à l'aide d'une base de données Aurora globale. Pour plus d'informations, consultez Haute disponibilité dans les régions AWS avec des bases de données Aurora globales.
  - Vous pouvez créer un réplica en lecture Aurora d'un cluster de bases de donnéesAurora MySQL dans une autre Région AWS à l'aide de la réplication de journal binaire (binlog) MySQL. Il est possible de créer de cette façon jusqu'à cinq réplicas en lecture par cluster, chacun dans une Région différente.
- Deux clusters de bases de données Aurora MySQL dans la même région , en utilisant la réplication du journal binaire (binlog) MySQL.
- Une instance source de données de base de données RDS for MySQL et un cluster de bases de données Aurora MySQL, en créant un réplica en lecture Aurora d'une instance de base de données RDS for MySQL. Cette approche est généralement utilisée pour une migration vers Aurora MySQL plutôt que pour une réplication continue.

Pour plus d'informations sur la réplication avec Aurora MySQL, consultez Réplication avec Amazon Aurora MySQL.

Réplication avec Aurora PostgreSQL

En plus des réplicas Aurora, Aurora PostgreSQL propose les options de réplication suivantes :

- Un cluster de bases de données Aurora principal dans une région et jusqu'à 10 clusters secondaires en lecture seule dans différentes régions en utilisant une base de données Aurora globale. Aurora PostgreSQL ne prend pas en charge les réplicas Aurora entre Régions. Toutefois, vous pouvez utiliser la base de données Aurora globale pour mettre à l'échelle les capacités de lecture de votre cluster de bases de données Aurora PostgreSQL sur plusieurs Régions AWS et pour atteindre les objectifs de disponibilité. Pour plus d'informations, consultez Utilisation d'Amazon Aurora Global Database.
- Deux clusters de bases de données Aurora PostgreSQL dans la même Région, à l'aide de la fonctionnalité de réplication logique de PostgreSQL.

- Une instance de base de données RDS pour PostgreSQL en tant que source de données et un cluster de bases de données Aurora PostgreSQL, en créant un réplica en lecture Aurora d'une instance de base de données RDS PostgreSQL. Cette approche est généralement utilisée pour une migration vers Aurora PostgreSQL plutôt que pour une réplication continue.

Pour plus d'informations sur la réplication avec Aurora PostgreSQL, consultez Réplication avec Amazon Aurora PostgreSQL.

Facturation d'une instance de base de données pour Aurora

Les instances Amazon RDS allouées dans un cluster Amazon Aurora sont facturées en fonction des composants suivants :

- Heures des instances de base de données (par heure) – En fonction de la classe de l'instance de base de données (par exemple, db.t2.small or db.m4.large). La tarification est indiquée selon une base horaire, mais les factures sont calculées à la seconde près et affichent les heures sous une forme décimale. L'utilisation de RDS est facturée par incréments de 1 seconde, avec un minimum de 10 minutes. Pour plus d'informations, consultez Classes d'instance de base de données Amazon Aurora.
- Stockage (par Gio, tous les mois) – Capacité de stockage provisionnée pour votre instance de base de données. Si vous mettez à l'échelle votre capacité de stockage provisionnée dans le mois, votre facture est calculée au prorata. Pour plus d'informations, consultez Stockage Amazon Aurora.
- Demandes d'entrée/sortie (E/S) (pour 1 million de demandes) : nombre total de demandes de stockage d'E/S que vous avez effectuées au cours d'un cycle de facturation, pour la configuration du cluster de bases de données Aurora Standard uniquement.

  Pour plus d'informations sur la facturation des E/S Amazon Aurora, consultez Configurations de stockage pour les clusters de bases de données Amazon Aurora.
- Stockage de sauvegarde (par Gio par mois) – Le stockage de sauvegarde est le stockage associé à vos sauvegardes de base de données automatisées et tout instantané de base de données active que vous avez pris. Augmenter votre période de rétention des sauvegardes ou prendre des instantanés de base de données supplémentaires augmente le stockage de sauvegarde consommé par votre base de données. La facturation à la seconde ne s'applique pas au stockage de sauvegarde (mesuré en Go/mois).

  Pour plus d'informations, consultez Sauvegarde et restauration d'un cluster de bases de données Amazon Aurora.

- Transfert de données (par Go) – Echange de données entre votre instance de base de données et Internet ou d'autres régions AWS. Pour des exemples utiles, consultez le post de blog AWS Exploring Data Transfer Costs for AWS Managed Databases.

Amazon RDS propose les options d'achat suivantes pour vous permettre d'optimiser vos coûts en fonction de vos besoins :

- On-Demand instances (Instances à la demande) : paiement à l'heure pour les heures d'instance de base de données que vous utilisez. La tarification est indiquée selon une base horaire, mais les factures sont calculées à la seconde près et affichent les heures sous une forme décimale. L'utilisation de RDS est facturée par incréments de 1 seconde, avec un minimum de 10 minutes.
- Reserved instances (Instances réservées) : réservez une instance de base de données pour un an ou trois ans, et bénéficiez d'une remise importante par rapport à la tarification des instances de base de données à la demande. Grâce aux instances réservées, vous pouvez lancer, supprimer, démarrer ou arrêter plusieurs instances pendant une heure, puis obtenir l'avantage d'instance réservé pour toutes les instances.
- Aurora Serverless v2 – Aurora Serverless v2 fournit une capacité à la demande où l'unité de facturation est l'unité de capacité Aurora (ACU) en heures au lieu des heures d'instance de base de données. La capacité Aurora Serverless v2 augmente et diminue, dans une fourchette que vous spécifiez, en fonction de la charge de votre base de données. Vous pouvez configurer un cluster où se trouve toute la capacité Aurora Serverless v2. Ou vous pouvez configurer une combinaison de Aurora Serverless v2 et d'instances allouées à la demande ou approvisionnées. Pour plus d'informations sur le fonctionnement des unités de capacité Aurora Serverless v2, consultez Fonctionnement d'Aurora Serverless v2.
- Base de données Aurora PostgreSQL Limitless : la base de données Aurora PostgreSQL Limitless est une fonctionnalité de mise à l'échelle horizontale automatisée qui permet de dépasser les limites de débit d'écriture et de stockage d'une instance de base de données unique. La base de données Limitless répartit la charge de travail sur plusieurs instances d'Aurora Writer, tout en préservant la facilité d'utilisation en tant que base de données unique. Limitless Database fournit une capacité à la demande où l'unité de facturation est le nombre d'heures d'unité de capacité Aurora (ACU) dans un groupe de partitions de base de données. Pour plus d'informations sur le fonctionnement des unités ACU de la base de données Limitless, consultez Création d'un cluster de bases de données utilisant Aurora PostgreSQL Limitless Database.

Pour obtenir des informations sur la tarification Aurora, consultez la page de tarification Aurora.

Rubriques

Instances de base de données à la demande pour Aurora

Les instances de base de données à la demande Amazon RDS sont facturées en fonction de la classe de l'instance de base de données (par exemple, db.t3.small ou db.m5.large). Pour plus d'informations sur la tarification de Amazon RDS, consultez la page produit de Amazon RDS.

La facturation commence pour une instance de base de données dès que cette dernière est disponible. La tarification est indiquée selon une base horaire, mais les factures sont calculées à la seconde près et affichent les heures sous une forme décimale. L'utilisation de Amazon RDS est facturée par incréments d'une seconde, avec un minimum de 10 minutes. Dans le cas d'une modification de configuration pouvant être facturée, comme le dimensionnement de la capacité de calcul ou de stockage, un minimum de 10 minutes vous est facturé. La facturation continue jusqu'à la résiliation de l'instance de base de données, qui a lieu lorsque vous supprimez cette dernière ou si elle échoue.

Si vous ne souhaitez plus être facturé pour votre instance de base de données, vous devez l'arrêter ou la supprimer afin d'éviter d'être facturé pour des heures d'instance de base de données supplémentaires. Pour plus d'informations sur les états des instances de base de données pour lesquelles vous êtes facturé, consultez Affichage du statut de l'instance de base de données dans un cluster Aurora.

Instances de base de données arrêtées

Pendant que votre instance de base de données est arrêtée, le stockage provisionné vous est facturé, y compris les IOPS provisionnés. Le stockage de sauvegarde vous est également facturé, notamment le stockage des instantanés manuels et des sauvegardes automatisées dans la fenêtre de conservation que vous avez spécifiée. Les heures de l'instance de base de données ne vous sont pas facturées.

Instances de base de données multi-AZ

Une configuration multi-AZ améliore la durabilité et la disponibilité des données en allouant et en maintenant automatiquement un réplica de secours synchrone dans une autre zone de disponibilité. En raison des ressources supplémentaires et de la disponibilité accrue, les déploiements multi-AZ sont plus chers que les déploiements mono-AZ, et peuvent coûter environ deux fois plus cher en raison de l'instance de secours supplémentaire et des ressources associées.

Notez les informations importantes suivantes sur la tarification multi-AZ :

- Coûts de calcul : facturés par heure d'instance de base de données pour les instances principales et de secours.
- Coûts de stockage : facturés par Go par mois pour le stockage provisionné pour les instances principales et de secours.
- Coûts de transfert de données : la réplication entre les instances principales et de secours est incluse dans le coût, mais d'autres frais de transfert de données peuvent s'appliquer en fonction de votre utilisation.

Pour estimer avec précision vos coûts mensuels en fonction de votre cas d'utilisation et Région AWS spécifiques, vous pouvez utiliser l'Calculateur de tarification AWS. Cet outil vous permet de saisir les détails de votre configuration et fournit une ventilation complète des coûts.

> **Note**
>
> La tarification est susceptible d'être modifiée. Consultez la page Tarification d'Amazon RDS pour obtenir les informations les plus à jour.

Instances de base de données réservées pour Amazon Aurora

En utilisant des instances de base de données réservées, vous pouvez réserver une instance de base de données pour une durée d'un an ou de trois ans. Ce type d'instance est beaucoup plus économique que les instances de bases de données à la demande. Les instances de bases de données réservées ne sont pas des instances physiques, mais correspondent à une remise sur la facturation appliquée à l'utilisation de certaines instances de bases de données à la demande dans votre compte. Les remises pour instances de base de données réservées sont liées au type d'instance et à la Région AWS.

En règle générale, pour utiliser des instances de bases de données réservées, commencez par recueillir des informations sur les offres disponibles, achetez l'offre qui vous convient, puis consultez le détail des instances de bases de données réservées existantes.

Pour plus d'informations sur l'achat d'instances de base de données réservées et sur la consultation de la facturation des instances de base de données réservées, consultez les sections suivantes.

- Achat d'instances de base de données réservées pour Amazon Aurora
- Affichage de la facturation relative aux instances de base de données réservées pour Amazon Aurora

Présentation des instances de base de données réservées

Lorsque vous achetez une instance de base de données réservée dans Amazon RDS, vous achetez la garantie d'obtenir un tarif réduit sur un type d'instance de bases de données spécifique pour la durée de l'instance de base de données réservée. Pour utiliser une instance de base de données réservée Amazon RDS, créez une instance de bases de données, tout comme vous le feriez pour une instance à la demande.

L'instance de base de données que vous créez doit comporter les mêmes spécifications que l'instance de base de données réservée pour les éléments suivants :

- Région AWS
- Moteur de base de données (Il n'est pas nécessaire que le numéro de version du moteur de base de données corresponde.)
- Type d'instance de base de données

Si les spécifications de la nouvelle instance de bases de données coïncident avec une instance de base de données réservée existante dans votre compte, vous êtes facturé au tarif réduit correspondant à cette dernière. Dans le cas contraire, l'instance de bases de données est facturée selon le tarif à la demande.

Vous pouvez modifier une instance de base de données que vous utilisez en tant qu'instance de base de données réservée. Si la modification est conforme aux spécifications de l'instance de base de données réservée, une partie ou la totalité de la remise s'applique toujours à l'instance de base de données modifiée. Si la modification est en dehors des spécifications, comme la modification de la classe d'instance, la remise ne s'applique plus. Pour plus d'informations, consultez Instances de base de données réservées de taille flexible.

Rubriques

- Types d'offres
- Flexibilité de configuration du cluster de bases de données Aurora
- Instances de base de données réservées de taille flexible
- Exemples de facturation d'une instance de base de données réservée Aurora
- Suppression d'une instance de base de données réservée

Pour plus d'informations sur les instances de bases de données réservées, ainsi que sur leur tarification, consultez Instances réservées Amazon RDS.

Types d'offres

Trois types d'instances de base de données réservées sont disponibles : sans paiement initial, avec paiement initial partiel et avec paiement initial total. Vous pouvez donc optimiser vos coûts Amazon RDS en vous basant sur votre utilisation prévue.

> **Note**
>
> Toutes les classes d'instance RDS ne prennent pas en charge tous les types d'offres d'instances réservées. Par exemple, certaines classes d'instance peuvent ne pas proposer l'option Pas à l'avance. Pour confirmer la disponibilité, consultez les offres d'instances réservées dans la commande AWS Management Console ou utilisez la `describe-reserved-db-instances-offerings` AWS CLI commande.

Sans frais initiaux

Cette option vous permet d'accéder à des instances de base de données réservées sans paiement initial. Les instances de bases de données réservées sans frais initiaux n'impliquent aucun paiement initial et sont facturées selon un taux horaire réduit pendant toute la durée de l'engagement, quelle que soit l'utilisation. Cette option est uniquement disponible dans le cadre d'une réservation d'un an.

Frais initiaux partiels

Cette option exige qu'une partie des instances de base de données réservées soit payée d'avance. Les heures restantes pendant la période sont facturées à un taux réduit, quelle que soit l'utilisation. Cette option remplace l'option précédente d'utilisation intensive.

Tous les frais initiaux

Le paiement complet est effectué en totalité au début de la période, sans aucun autre coût pour le reste de la réservation, quel que soit le nombre d'heures utilisé.

Si vous utilisez une facturation consolidée, tous les comptes de l'organisation sont traités comme s'il s'agissait d'un seul compte. Cela signifie que tous les comptes de l'organisation peuvent bénéficier d'un surplus d'heures correspondant aux instances de base de données réservées qui sont achetées par un autre compte. Pour plus d'informations sur la facturation consolidée, consultez Instances de base de données réservées Amazon RDS dans le Guide de l'utilisateur AWS Billing and Cost Management.

Flexibilité de configuration du cluster de bases de données Aurora

Vous pouvez utiliser des instances de bases de données réservées Aurora avec les deux configurations de clusters de bases de données :

- Aurora I/O-Optimized— Vous ne payez que pour l'utilisation et le stockage de vos clusters de base de données, sans frais supplémentaires pour les I/O opérations de lecture et d'écriture.
- Aurora Standard— Outre l'utilisation et le stockage de vos clusters de base de données, vous payez également un tarif standard pour 1 million de demandes d' I/O opérations.

Aurora prend automatiquement en compte la différence de prix entre ces configurations. Aurora I/O-Optimized consomme 30 % d'unités normalisées en plus par heure par rapport à Aurora Standard.

Pour plus d'informations sur les configurations du stockage de cluster Aurora, consultez Configurations de stockage pour les clusters de bases de données Amazon Aurora. Pour plus d'informations sur la tarification des configurations du stockage de cluster Aurora, consultez Tarification Amazon Aurora.

Instances de base de données réservées de taille flexible

Lorsque vous achetez une instance de base de données réservée, vous devez spécifier la classe d'instance, par exemple, db.r5.large. Pour plus d'informations sur les classes d'instance de base de données, consultez Classes d'instance de base de données Amazon Aurora.

Si vous devez augmenter la capacité d'une instance de base de données, l'instance réservée est automatiquement appliquée à l'instance de base de données que vous avez dimensionnée. En d'autres termes, les instances de base de données réservées sont appliquées automatiquement aux classes d'instance de bases de données, quelle que soit leur taille. Des instances de base de données réservées de taille flexible sont disponibles pour les instances de base de données dotées du même moteur de base Région AWS de données. Des instances de bases de données réservées de taille flexible peuvent uniquement être mises à l'échelle dans leur type de classe d'instance. Par exemple, une instance de base de données réservée pour un fichier db.r6i.large peut s'appliquer à un fichier db.r6i.xlarge, mais pas à un fichier db.r6id.large ou db.r7g.large, car db.r6id.large et db.r7g.large sont des types de classes d'instance différents.

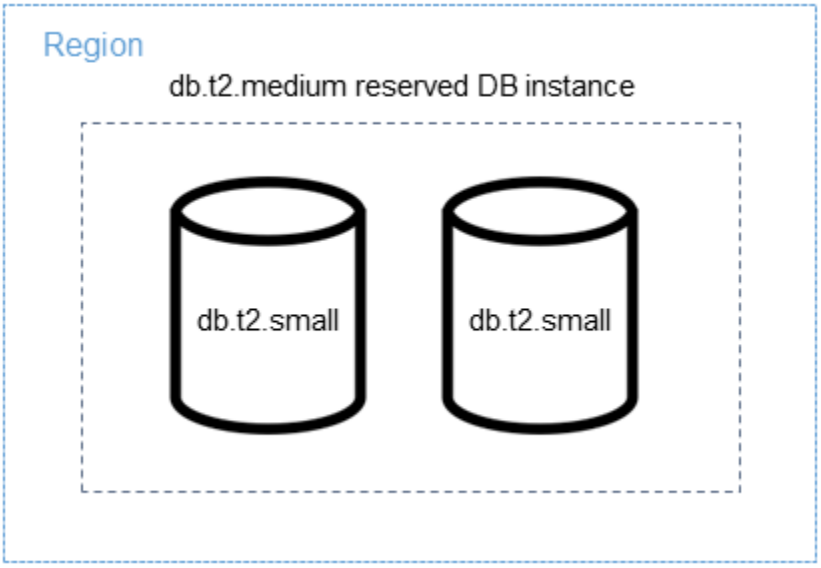
Des instances de base de données réservées de taille flexible sont disponibles pour les moteurs de base de données Aurora suivants :

- Aurora MySQL
- Aurora PostgreSQL

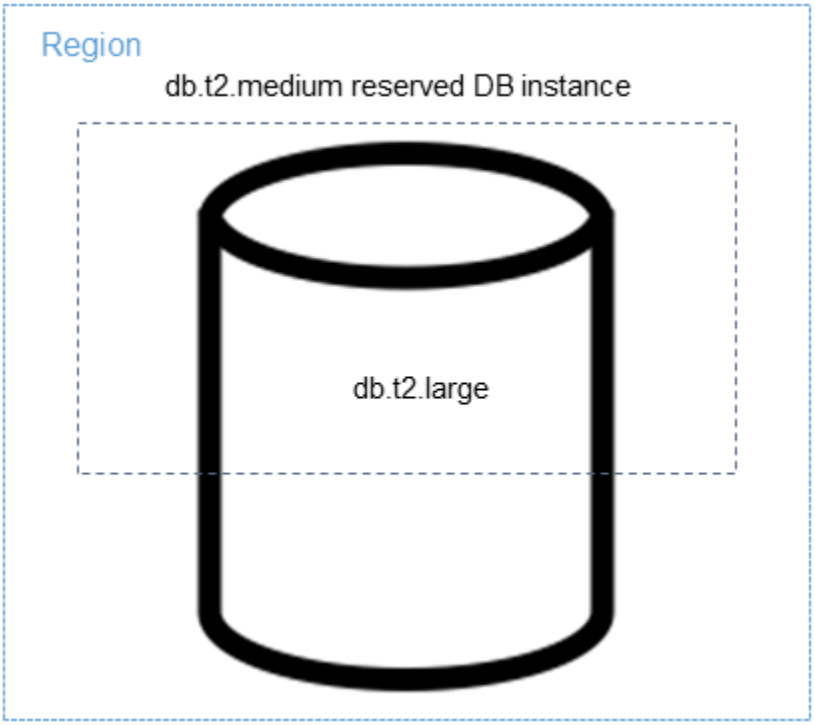
Les unités normalisées par heure permettent de comparer l'utilisation pour différentes tailles d'instances de base de données réservées. Par exemple, une unité d'utilisation sur deux instances de bases de données db.r3.large équivaut à huit unités normalisées par heure d'utilisation sur une instance db.r3.small. La table suivante indique le nombre d'unités normalisées par heure pour chaque taille d'instance de bases de données.

Taille d’instance	Unités normalisées par heure pour une instance de base de données, Aurora Standard	Unités normalisées par heure pour une instance de base de données, Aurora I/O-Optimized	Unités normalisées par heure pour trois instances de base de données (un rédacteur et deux lecteurs), Aurora Standard	Unités normalisées par heure pour trois instances de base de données (un rédacteur et deux lecteurs), Aurora I/O-Optimized
petit	1	1.3	3	3.9
medium	2	2.6	6	7.8
large	4	5.2	12	15,6
xlarge	8	10.4	24	31,2
2xlarge	16	20,8	48	62,4
4xlarge	32	41,6	96	124,8
8xlarge	64	83,2	192	249,6
12xlarge	96	124,8	288	374,4
16xlarge	128	166,4	384	499,2
24xlarge	192	249,6	576	748,8
32xlarge	256	332,8	768	998,4

Par exemple, supposons que vous achetez une instance de base de données réservée `db.t2.medium` et que deux instances de base de données `db.t2.small` sont exécutées dans votre compte dans la même Région AWS. Dans ce cas, l’avantage de facturation est appliqué entièrement à ces deux instances.



Sinon, si une `db.t2.large` instance est exécutée sur votre compte dans le même compte Région AWS, l'avantage de facturation est appliqué à 50 % de l'utilisation de l'instance de base de données.



> **Note**
>
> Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à la

production. Pour plus de détails sur les classes d'instance T, consultez Types de classes d'instance de base de données.

Exemples de facturation d'une instance de base de données réservée Aurora

Les exemples suivants illustrent la tarification des instances de bases de données réservées pour les clusters de bases de données Aurora utilisant à la fois les configurations de clusters de bases de données Aurora Standard et Aurora I/O-Optimized.

Exemple avec Aurora Standard

Le prix d'une instance de base de données réservée n'offre pas de réduction sur les coûts associés au stockage, aux sauvegardes et aux E/S. L'exemple suivant illustre le coût total mensuel d'une instance de base de données réservée :

- Classe d'instance de base de données db.r5.large mono-AZ réservée Aurora MySQL dans la région USA Est (Virginie du Nord) au coût de 0,19 USD par heure ou de 138,70 USD par mois
- Stockage Aurora au coût de 0,10 USD par mois (soit 45,60 USD par mois présumés pour cet exemple)
- Aurora I/O au coût de 0,20 dollar par million de demandes (supposons 20 dollars par mois pour cet exemple)
- Stockage de sauvegarde Aurora au coût de 0,021 USD par Gio et par mois (soit 30 USD par mois présumés pour cet exemple)

Ajoutez toutes ces options (138,70 USD + 45,60 USD + 20 USD + 30 USD) à l'instance de base de données réservée : le coût total mensuel est de 234,30 USD.

Si vous choisissez d'utiliser une instance de base de données à la demande au lieu d'une instance de base de données réservée, une classe d'instance de base de données db.r5.large mono-AZ Aurora MySQL dans la région USA Est (Virginie du Nord) coûte 0,29 USD par heure ou 217,50 USD par mois. Pour une instance de base de données à la demande, ajoutez toutes ces options (217,50 USD + 45,60 USD + 20 USD + 30 USD) ; le coût total mensuel est de 313,10 USD. L'instance de base de données réservée vous permet d'économiser près de 79 USD par mois.

Exemple avec un cluster de bases de données Aurora Standard et deux instances de lecteur

Pour utiliser des instances réservées pour les clusters de bases de données Aurora, achetez simplement une instance réservée pour chaque instance de base de données du cluster.

Dans le prolongement du premier exemple, vous avez un cluster de bases de données Aurora MySQL avec une instance de base de données d'écriture et deux réplicas Aurora, soit un total de trois instances de base de données dans le cluster. Les deux réplicas Aurora n'entraînent pas de frais de stockage ou de sauvegarde supplémentaires. Si vous achetez trois instances de base de données réservées Aurora MySQL db.r5.large, votre coût sera de 234,30$ (pour l'instance de base de données Writer) + 2* (138,70$ + 20$ par réplique Aurora), pour un total de 551,70$ I/O par mois.

Le coût à la demande correspondant pour un cluster de base de données Aurora MySQL avec une instance de base de données d'écriture et deux répliques Aurora est de 313,10$ + 2 * (217,50$ + 20$ I/O par instance) pour un total de 788,10$ par mois. Vous économisez 236,40 USD en utilisant les instances de base de données réservées.

Exemple avec Aurora I/O-Optimized

Vous pouvez réutiliser vos instances de base de données Aurora Standard réservées existantes avec Aurora I/O-Optimized. Pour profiter pleinement des avantages de vos remises sur les instances réservées avec Aurora I/O-Optimized, vous pouvez acheter 30 % d'instances réservées supplémentaires similaires à vos instances réservées actuelles.

La table suivante montre des exemples d'estimation des instances réservées supplémentaires avec Aurora I/O-Optimized. Si les instances réservées requises constituent une fraction, vous pouvez tirer parti de la flexibilité de taille offerte par les instances réservées pour obtenir un nombre entier. Dans ces exemples, le terme « en cours » fait référence aux instances réservées Aurora Standard que vous possédez actuellement. Les instances réservées supplémentaires correspondent au nombre d'instances réservées Aurora Standard que vous devez acheter pour conserver vos réductions actuelles sur les instances réservées avec Aurora I/O-Optimized.

Classe d'instance de base de données	Instances réservées Aurora Standard actuelles	Instances réservées requises pour Aurora I/O-Optimized	Instances réservées supplémentaires nécessaires	Instances réservées supplémentaires nécessaires, grâce à la flexibilité de la taille
db.r6g.large	10	10 x 1,3 = 13	3 x db.r6g.large	3 x db.r6g.large
db.r6g.4xlarge	20	20 x 1,3 = 26	6 x db.r6g.4xlarge	6 x db.r6g.4xlarge

Classe d’instance de base de données	Instances réservées Aurora Standard actuelles	Instances réservées requises pour Aurora I/O-Optimized	Instances réservées supplémentaires nécessaires	Instances réservées supplémentaires nécessaires, grâce à la flexibilité de la taille
db.r6g.12xlarge	5	5 x 1,3 = 6,5	1.5 * db.r6g.12xlarge	Un de chacun des formats db.r6g.12xlarge, r6g.4xlarge et r6g.2xlarge (0,5 x db.r6g.12xlarge = 1 x db.r6g.4xlarge + 1 x db.r6g.2xlarge)
db.r6i.24xlarge	15	15 x 1,3 = 19,5	4.5 * db.r6i.24xlarge	4 x db.r6i.24xlarge + 1 x db.r6i.12xlarge (0,5 x db.r6i.24xlarge = 1 x db.r6i.12xlarge)

Exemple avec un cluster de bases de données Aurora I/O-Optimized et deux instances de lecteur

Vous avez un cluster de bases de données Aurora MySQL avec une instance de base de données de rédacteur et deux réplicas Aurora, soit un total de trois instances de base de données dans le cluster. Ils utilisent la configuration du cluster de bases de données Aurora I/O-Optimized. Pour utiliser des instances de bases de données réservées pour ce cluster, vous devez acheter quatre instances de base de données réservées de la même classe d’instance de base de données.

3 instances de base de données qui utilisent Aurora I/O-Optimized consomment 3,9 unités normalisées par heure, contre 3 unités normalisées par heure pour 3 instances de base de données utilisant Aurora Standard. Cependant, vous économisez les I/O coûts mensuels pour chaque instance de base de données.

> **Note**
>
> Les prix indiqués ici sont des exemples et ne correspondent pas aux prix réels. Pour obtenir des informations sur la tarification d'Aurora, consultez Tarification d'Amazon Aurora.

Suppression d'une instance de base de données réservée

Les conditions d'une instance de base de données réservée impliquent un engagement d'un an ou de trois ans. Il n'est pas possible d'annuler une instance de base de données réservée. Toutefois, vous pouvez supprimer une instance de base de données à laquelle s'applique une remise d'instance de base de données réservée. Le processus de suppression d'une instance de base de données couverte par ce type de remise est le même que pour n'importe quelle autre instance de bases de données.

Vous êtes facturé pour les coûts initiaux, que vous utilisiez ou non les ressources.

Si vous supprimez une instance de base de données à laquelle s'applique une remise d'instance de base de données réservée, vous pouvez lancer toute autre instance de bases de données dont les spécifications sont compatibles. Dans ce cas, vous conservez le tarif réduit jusqu'à la fin de la période de réservation (d'un ou de trois ans).

Achat d'instances de base de données réservées pour Amazon Aurora

Vous pouvez utiliser l'APIAWS Management Console, leAWS CLI, et l'API RDS pour travailler avec des instances de base de données réservées.

Console

Vous pouvez utiliser le AWS Management Console pour travailler avec des instances de base de données réservées, comme indiqué dans les procédures suivantes.

Pour obtenir la tarification et les informations relatives aux offres d'instances de bases de données réservées disponibles

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le volet de navigation, choisissez Instances réservées.
3. Choisissez Purchase Reserved DB Instance (Instance de base de données réservée à l'achat).
4. Pour Description du produit, choisissez le moteur de base de données et le type de licence.
5. Pour Classe d'instance de base de données, choisissez la classe d'instance de base de données.
6. Pour Option de déploiement, choisissez si vous souhaitez un déploiement d'instance de base de données mono-AZ ou multi-AZ.

   > **Note**
   >
   > Les instances Amazon Aurora réservées ont toujours l'option de déploiement configurée sur Instance de base de données mono-AZ. Toutefois, lorsque vous créez un cluster de bases de données Aurora, l'option de déploiement par défaut est Créer un réplica Aurora ou lecteur dans une autre zone de disponibilité (multi-AZ).
   > Vous devez acheter une instance de base de données réservée pour chaque instance que vous prévoyez d'utiliser, y compris les réplicas Aurora. Par conséquent, pour les déploiements multi-AZ sur Aurora, vous devez acheter des instances de base de données réservées supplémentaires.

7. Pour Durée, choisissez la durée pendant laquelle vous souhaitez réserver l'instance de base de données.
8. Pour Type d'offre, choisissez le type d'offre.

   Les informations relatives à la tarification s'affichent après la sélection du type d'offre.

   > **Important**
   >
   > Choisissez Annuler pour éviter d'acheter l'instance de base de données réservée et d'avoir à payer des frais.

Une fois que vous disposez des informations requises sur les offres d'instances de bases de données réservées disponibles, vous pouvez utiliser ces informations pour acheter une offre, comme le montre la procédure suivante.

Pour acheter une instance de base de données réservée

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le volet de navigation, choisissez Instances réservées.
3.
> **Important**
>
> Avant de continuer, vérifiez que vous êtes dans la bonne Région AWS. Les instances de base de données réservées sont spécifiques à une région et ne peuvent pas être transférées entre les régions. Vérifiez le Sélecteur de région dans l'angle supérieur droit de la console pour être sûr que vous êtes en train d'acheter l'instance réservée dans la bonne région.

4. Choisissez Purchase reserved DB instance (Acheter une instance de base de données réservée).
5. Pour Description du produit, choisissez le moteur de base de données et le type de licence.
6. Pour Classe d'instance de base de données, choisissez la classe d'instance de base de données.
7. Pour Déploiement multi-AZ, choisissez si vous souhaitez un déploiement d'instance de base de données mono-AZ ou multi-AZ.

> **Note**
>
> Les instances Amazon Aurora réservées ont toujours l'option de déploiement configurée sur Instance de base de données mono-AZ. Lorsque vous créez un cluster de bases de données Amazon Aurora depuis l'instance de base de données réservée, il est automatiquement créé comme multi-AZ. Veillez à acheter une instance de base de données réservée pour chaque instance de base de données que vous prévoyez d'utiliser, y compris les réplicas Aurora.

8. Pour Durée, choisissez la durée pendant laquelle vous souhaitez que l'instance de base de données soit réservée.
9. Pour Type d'offre, choisissez le type d'offre.

Les informations relatives à la tarification s'affichent après que vous avez choisi le type d'offre. Console Acheter une instance de base de données réservée

10. (Facultatif) Afin de faciliter le suivi des instances de base de données réservées que vous achetez, vous pouvez leur attribuer un identifiant de votre choix. Dans Reserved Id (ID réservé), tapez un identifiant pour l'instance de bases de données réservée.
11. Sélectionnez Soumettre.

    Votre instance de base de données réservée est achetée, puis affichée dans la liste Reserved instances (Instances réservées).

Une fois que vous avez acheté une instance de bases de données réservée, suivez la procédure ci-dessous afin d'en consulter le détail.

Pour obtenir des informations sur les instances de base de données réservées pour votre AWS compte

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le volet Navigation, choisissez Instances réservées.

   Les instances de bases de données réservées pour votre compte s'affichent. Pour afficher des informations détaillées sur une instance de base de données réservée particulière, choisissez cette instance dans la liste. Vous pouvez alors consulter des informations détaillées sur cette instance dans le volet de détails au bas de la console.

AWS CLI

Vous pouvez utiliser le AWS CLI pour travailler avec des instances de base de données réservées, comme indiqué dans les exemples suivants.

Example d'obtention des offres d'instances de base de données réservées disponibles

Pour obtenir des informations sur les offres d'instances de base de données réservées disponibles, appelez la AWS CLI commande `describe-reserved-db-instances-offerings`.

```
aws rds describe-reserved-db-instances-offerings
```

Cet appel vous renvoie des informations semblables à ce qui suit :

```
OFFERING  OfferingId                            Class         Multi-AZ  Duration  Fixed
 Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  db.r3.large   y         1y
 1820.00 USD  0.368 USD    mysql        Partial  Upfront
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  db.r3.small   n         1y
 227.50 USD  0.046 USD    mysql        Partial  Upfront
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  db.r3.small   n         1y
 162.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  db.r3.large   y         1y
 700.00 USD   0.00 USD    mysql        All      Upfront
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  db.r3.xlarge  n         1y
 4242.00 USD   2.42 USD    mysql        No       Upfront
```

Une fois que vous disposez des informations requises sur les offres d'instances de base de données réservées disponibles, vous pouvez utiliser ces informations pour acheter une offre, comme le montre l'exemple suivant.

Pour acheter une instance de base de données réservée, utilisez la AWS CLI commande `purchase-reserved-db-instances-offering`avec les paramètres suivants :

- `--reserved-db-instances-offering-id` : identifiant de l'offre que vous voulez acheter. Reportez-vous à l'exemple précédent pour obtenir l'ID de l'offre.
- `--reserved-db-instance-id` : vous pouvez attribuer l'identifiant de votre choix aux instances de base de données réservées que vous achetez pour en faciliter le suivi.

Example d'achat d'une instance de base de données réservée

L'exemple suivant achète l'offre d'instance de base de données réservée avec ID *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f* et attribue l'identifiant de*MyReservation*.

Pour Linux, macOS ou Unix :

```
aws rds purchase-reserved-db-instances-offering \
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-db-instance-id MyReservation
```

Pour Windows :

```
aws rds purchase-reserved-db-instances-offering ^
    --reserved-db-instances-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-db-instance-id MyReservation
```

La commande renvoie un résultat semblable à ce qui suit :

```
RESERVATION  ReservationId      Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State           Description  Offering Type
RESERVATION  MyReservation      db.r3.small  y         2011-12-19T00:30:23.247Z  1y
    455.00 USD   0.092 USD    1      payment-pending  mysql        Partial  Upfront
```

Après avoir acheté des instances de bases de données réservées, vous pouvez en consulter le détail.

Pour obtenir des informations sur les instances de base de données réservées pour votre AWS compte, appelez la AWS CLI commande `describe-reserved-db-instances`, comme indiqué dans l'exemple suivant.

Example d’obtenir vos instances de bases de données réservées

```
aws rds describe-reserved-db-instances
```

La commande renvoie un résultat semblable à ce qui suit :

```
RESERVATION  ReservationId     Class        Multi-AZ  Start Time
 Duration  Fixed Price  Usage Price  Count  State    Description  Offering Type
RESERVATION  MyReservation     db.r3.small  y         2011-12-09T23:37:44.720Z  1y
   455.00 USD   0.092 USD    1      retired  mysql        Partial  Upfront
```

API RDS

Vous pouvez utiliser l’API RDS pour travailler avec des instances de base de données réservées :

- Pour obtenir des informations sur les offres d’instances de bases de données réservées disponibles, exécutez l’opération de l’API Amazon RDS `DescribeReservedDBInstancesOfferings`.
- Une fois que vous disposez des informations requises sur les offres d’instances de base de données réservées disponibles, vous pouvez utiliser ces informations pour

acheter une offre, comme le montre l'exemple suivant. Appelez l'opération d'API RDS `PurchaseReservedDBInstancesOffering` avec les paramètres suivants :

- `--reserved-db-instances-offering-id` : identifiant de l'offre que vous voulez acheter.
- `--reserved-db-instance-id` : vous pouvez attribuer l'identifiant de votre choix aux instances de base de données réservées que vous achetez pour en faciliter le suivi.

- Après avoir acheté des instances de bases de données réservées, vous pouvez en consulter le détail. Exécutez l'opération de l'API RDS `DescribeReservedDBInstances`.

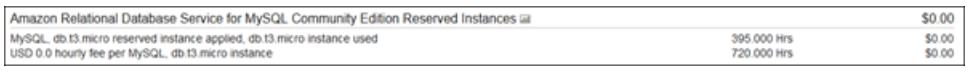
Affichage de la facturation relative aux instances de base de données réservées pour Amazon Aurora

Vous pouvez afficher la facturation de vos instances de base de données réservées dans le tableau de bord de facturation de la AWS Management Console.

Pour afficher la facturation des instances de base de données réservées

1. Connectez-vous à la AWS Management Console.
2. De le menu du compte, en haut à droite, choisissez Billing Dashboard (Tableau de bord de facturation).
3. Choisissez Bill Details (Détails de facturation) dans le coin supérieur droit du tableau de bord.
4. Sous AWS Service Charges (Frais de service), développez Relational Database Service (Service de base de données relationnelle).
5. Développez la Région AWS où se trouvent vos instances de base de données réservées, par exemple US West (Oregon) (USA Ouest (Oregon)).

   Vos instances de base de données réservées et leurs frais horaires pour le mois en cours sont affichés sous Amazon Relational Database Service (Service de base de données relationnelle) pour ***Database Engine (Moteur de base de données)***Reserved Instances (Instances réservées).

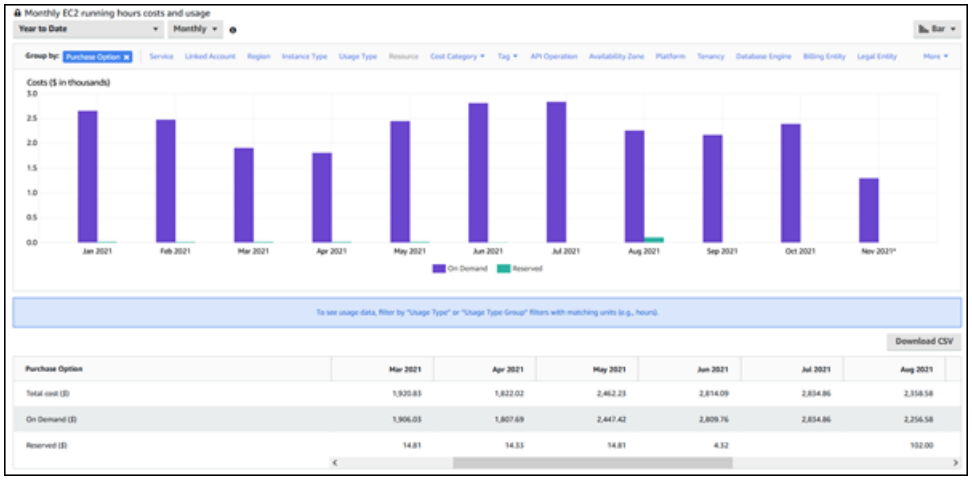
   

   L'instance de base de données réservée dans cet exemple a été achetée avec un paiement total anticipé et dès lors, il n'existe pas de frais horaires.
6. Cliquez sur l'icône Cost Explorer (graphique à barres) en regard de l'en-tête Reserved Instances.

Cost Explorer affiche le graphique Monthly EC2 running hours costs and usage (Coûts d'heures de fonctionnement et utilisation d'EC2 (base mensuelle)).

7. Effacez le filtre Usage Type Group (Groupe de type d'utilisation) situé à droite du graphique.
8. Choisissez la période et l'unité de temps pour lesquelles vous souhaitez examiner les coûts d'utilisation.

   L'exemple suivant illustre les coûts d'utilisation mensuels des instances de base de données à la demande et réservées pour l'année écoulée.



Les coûts des instances de base de données réservées de janvier à juin 2021 correspondent à des frais mensuels pour une instance avec frais initiaux partiels, tandis que les coûts d'août 2021 correspondent à des frais uniques pour une instance avec tous les frais initiaux.

La remise d'instance réservée pour l'instance avec frais initiaux partiels a expiré en juin 2021, mais l'instance de base de données n'a pas été supprimée. Après la date d'expiration, elle a simplement été facturée au tarif à la demande.

Configuration de votre environnement pour Amazon Aurora

Avant d'utiliser Amazon Aurora pour la première fois, exécutez les tâches suivantes.

Rubriques

Si vous possédez déjà un Compte AWS, connaissez vos exigences en matière d'Aurora et préférez utiliser les valeurs par défaut pour les groupes de sécurité IAM et VPC, passez directement à. Mise en route avec Amazon Aurora

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez l'https://portal.aws.amazon.com/billing/inscription.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique ou un SMS et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur, et à utiliser uniquement l'utilisateur racine pour effectuer les tâches nécessitant un accès utilisateur racine.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. À tout moment, vous pouvez consulter l'activité actuelle de votre compte et gérer votre compte en accédant à https://aws.amazon.com/et en choisissant Mon compte.

Création d’un utilisateur doté d’un accès administratif

Après vous être inscrit à un Compte AWS, sécurisez Utilisateur racine d'un compte AWS AWS IAM Identity Center, activez et créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l’aide pour vous connecter en utilisant l’utilisateur racine, consultez Connexion en tant qu’utilisateur racine dans le Guide de l’utilisateur Connexion à AWS .
2. Activez l’authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, voir Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d’un utilisateur doté d’un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d’ AWS IAM Identity Center dans le Guide de l’utilisateur AWS IAM Identity Center .
2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connexion en tant qu’utilisateur doté d’un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l’URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l’utilisateur IAM Identity Center.

Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribution d’un accès à d’autres utilisateurs

1. Dans IAM Identity Center, créez un ensemble d’autorisations qui respecte la bonne pratique consistant à appliquer les autorisations de moindre privilège.

   Pour obtenir des instructions, consultez Création d’un ensemble d’autorisations dans le Guide de l’utilisateur AWS IAM Identity Center .
2. Attribuez des utilisateurs à un groupe, puis attribuez un accès par authentification unique au groupe.

   Pour obtenir des instructions, consultez Ajout de groupes dans le Guide de l’utilisateur AWS IAM Identity Center .

Octroi d’un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS l'extérieur du AWS Management Console. La manière d'accorder un accès programmatique dépend du type d'utilisateur qui y accède AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l’une des options suivantes.

Quel utilisateur a besoin d’un accès programmatique ?	Pour	Méthode
IAM	(Recommandé) Utilisez les informations d'identification de la console comme informati ons d'identification temporair es pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l’interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Connexion pour le développement AWS local dans le guide de AWS

Quel utilisateur a besoin d’un accès programmatique ?	Pour	Méthode
		Command Line Interface l'utilisateur. • Pour AWS SDKs, voir Connexion pour le développement AWS local dans le guide de référence AWS SDKs and Tools.
Identité de la main-d’œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l’interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Configuration du AWS CLI à utiliser AWS IAM Identity Center dans le guide de AWS Command Line Interface l'utilisateur. • Pour AWS SDKs, outils, et AWS APIs, voir Authentification IAM Identity Center dans le guide de référence AWS SDKs et Tools.
IAM	Utilisez des informations d'identification temporaires pour signer les demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec AWS les ressources du Guide de l'utilisateur IAM.

Quel utilisateur a besoin d’un accès programmatique ?	Pour	Méthode
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes programmatiques adressées au AWS CLI AWS SDKs, ou AWS APIs.	Suivez les instructions de l’interface que vous souhaitez utiliser. • Pour le AWS CLI, voir Authentification à l'aide des informations d'identification utilisateur IAM dans le Guide de l'AWS Command Line Interface utilisateur. • Pour les outils AWS SDKs et, voir Authentifier à l'aide d'informations d'identif ication à long terme dans le guide de référence des outils AWS SDKs et. • Pour AWS APIs, voir Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Déterminer les exigences

La fondation de base d’Aurora est le cluster de bases de données. Une ou plusieurs instances de base de données peuvent appartenir à un cluster de bases de données. Un cluster de bases de données fournit une adresse réseau appelée point de terminaison de cluster. Vos applications se connectent au point de terminaison de cluster exposé par le cluster de bases de données lorsqu’elles doivent se connecter aux bases de données créées dans ce cluster de bases de données. Les informations que vous spécifiez lorsque vous créez le cluster de bases de données permettent de contrôler les éléments de la configuration, tels que le stockage, la mémoire, le moteur et la version de la base de données, la configuration réseau, la sécurité et les périodes de maintenance.

Avant de créer un cluster de bases de données et un groupe de sécurité, vous devez connaître vos besoins en termes de cluster de bases de données et de réseau. Voici quelques éléments importants à prendre en compte :

- Exigences en matière de ressources– Quelles sont les exigences de votre application ou de votre service en termes de mémoire et de processeur ? Vous utiliserez ces paramètres pour déterminer la classe d'instance de base de données à utiliser lors de la création de votre cluster de bases de données. Pour obtenir les caractéristiques des classes des instances de base de données, consultez Classes d'instance de base de données Amazon Aurora.
- VPC, sous-réseau et groupe de sécurité : votre cluster de bases de données se trouvera dans un cloud privé virtuel (VPC). Des règles de groupe de sécurité doivent être configurées pour la connexion à un cluster de bases de données. La liste suivante décrit les règles pour chaque option de VPC :
  - VPC par défaut : si votre AWS compte possède un VPC par défaut dans la région AWS , ce VPC est configuré pour prendre en charge les clusters de base de données. Si vous spécifiez le VPC par défaut lorsque vous créez le cluster de bases de données :
    - Assurez-vous de créer un groupe de sécurité VPC autorisant les connexions de l'application ou du service au cluster de bases de données Aurora. Utilisez l'option Groupe de sécurité sur la console VPC ou pour créer des groupes AWS CLI de sécurité VPC. Pour plus d'informations, consultez Étape 3 : créer un groupe de sécurité VPC.
    - Vous devez spécifier le groupe de sous-réseaux DB par défaut. S'il s'agit du premier cluster de base de données que vous créez dans la AWS région, Amazon RDS créera le groupe de sous-réseaux de base de données par défaut lors de la création du cluster de bases de données.
  - VPC défini par l'utilisateur — Si vous souhaitez spécifier un VPC défini par l'utilisateur lorsque vous créez un cluster de bases de données :
    - Assurez-vous de créer un groupe de sécurité VPC autorisant les connexions de l'application ou du service au cluster de bases de données Aurora. Utilisez l'option Groupe de sécurité sur la console VPC ou pour créer des groupes AWS CLI de sécurité VPC. Pour plus d'informations, consultez Étape 3 : créer un groupe de sécurité VPC.
    - Le VPC doit respecter certaines exigences afin d'héberger des clusters de bases de données. Il doit notamment comporter au moins deux sous-réseaux, dans deux zones de disponibilités distinctes. Pour plus d'informations, consultez Amazon VPC et Amazon Aurora.

    - Vous devez spécifier un groupe de sous-réseaux DB définissant les sous-réseaux de ce VPC pouvant être utilisés par le cluster de bases de données. Pour plus d'informations, consultez Groupe de sous-réseau DB de Utilisation d'un cluster de bases de données dans un VPC.
- Haute disponibilité : Avez-vous besoin de la prise en charge du basculement ? Sur Aurora, un déploiement multi-AZ crée une instance principale et des réplicas Aurora. Vous pouvez configurer l'instance principale et les réplicas Aurora afin qu'ils soient placés dans des zones de disponibilité différentes pour la prise en charge du basculement. Nous recommandons les déploiements multi-AZ pour les charges de travail de production afin de maintenir une haute disponibilité. À des fins de développement et de test, vous pouvez utiliser un non-Multi-AZ déploiement. Pour de plus amples informations, veuillez consulter Haute disponibilité pour Amazon Aurora.
- Politiques IAM : votre AWS compte dispose-t-il de politiques qui accordent les autorisations nécessaires pour effectuer des opérations Amazon RDS ? Si vous vous connectez à AWS l'aide d'informations d'identification IAM, votre compte IAM doit disposer de politiques IAM qui accordent les autorisations requises pour effectuer des opérations Amazon RDS. Pour de plus amples informations, veuillez consulter Identity and Access Management pour Amazon Aurora.
- Ports ouverts : sur quel TCP/IP port votre base de données écoutera-t-elle ? Dans certaines entreprises, le pare-feu peut bloquer les connexions vers le port par défaut de votre moteur de base de données. Si le pare-feu de votre entreprise bloque le port par défaut, choisissez un autre port pour le nouveau cluster DB. Notez que lorsque vous créez un cluster de bases de données qui écoute sur un port spécifié par vos soins, vous pouvez changer de port en modifiant le cluster de bases de données.
- AWS Région : Dans quelle AWS région souhaitez-vous disposer de votre base de données ? La proximité entre la base de données et l'application ou le service Web service permet de réduire la latence du réseau. Pour plus d'informations, consultez Régions et zones de disponibilité.

Lorsque vous disposez de toutes les informations nécessaires pour créer le groupe de sécurité et le cluster de bases de données, passez à l'étape suivante.

Créer un groupe de sécurité qui autorise l'accès au cluster de bases de données dans le VPC

Votre cluster de bases de données sera créé dans un VPC. Les groupes de sécurité autorisent l'accès au cluster de bases de données dans le VPC. Ils font office de pare-feu pour le cluster de bases de données associé, en contrôlant le trafic entrant et le trafic sortant au niveau du cluster. Par défaut, les clusters de bases de données sont créés avec un pare-feu et un groupe de sécurité

par défaut empêchant d'accéder au cluster de bases de données. Vous devez donc ajouter des règles à un groupe de sécurité qui vous permettent de vous connecter à votre cluster de bases de données. Utilisez les informations relatives au réseau et à la configuration déterminées lors de l'étape précédente pour créer les règles autorisant l'accès à votre cluster de bases de données.

Par exemple, si l'une de vos applications doit accéder à une base de données de votre cluster de bases de données situé dans un VPC, vous devez ajouter une règle TCP personnalisée qui spécifie la plage de ports et les adresses IP utilisées par l'application pour accéder à la base de données. Si vous avez une application sur une EC2 instance Amazon, vous pouvez utiliser le groupe de sécurité VPC que vous avez configuré pour l'instance Amazon EC2.

Vous pouvez configurer la connectivité entre une EC2 instance Amazon et un cluster de bases de données lorsque vous créez le cluster de base de données. Pour de plus amples informations, veuillez consulter Configurer la connectivité réseau automatique avec une instance EC2.

> **Tip**
>
> Vous pouvez configurer automatiquement la connectivité réseau entre une EC2 instance Amazon et un cluster de base de données lorsque vous créez le cluster de base de données. Pour de plus amples informations, veuillez consulter Configurer la connectivité réseau automatique avec une instance EC2.

Pour plus d'informations sur la façon de connecter les ressources d'Amazon Lightsail à vos clusters de base de données, consultez Connect Lightsail resources to using VPC peering. Services AWS

Pour plus d'informations sur la création d'un VPC à utiliser avec Aurora, consultez Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement). Pour plus d'informations sur les scénarios courants d'accès à une instance de base de données, consultez Scénarios d'accès à un cluster de bases de données d'un VPC.

Pour créer un groupe de sécurité VPC

1. Connectez-vous à la console Amazon VPC AWS Management Console et ouvrez-la à https://console.aws.amazon.com l'adresse /vpc.

   > **Note**
   >
   > Assurez-vous que vous êtes dans la console VPC, et non dans la console RDS.

2. Dans le coin supérieur droit du AWS Management Console, choisissez la AWS région dans laquelle vous souhaitez créer votre groupe de sécurité VPC et votre cluster de base de données. Dans la liste des ressources Amazon VPC pour cette région AWS , vous devriez voir au moins un VPC et plusieurs sous-réseaux. Si ce n'est pas le cas, vous n'avez pas de VPC par défaut dans cette AWS région.
3. Dans le panneau de navigation, choisissez Groupes de sécurité.
4. Sélectionnez Create security group (Créer un groupe de sécurité).

   La page Create security group (Créer un groupe de sécurité) s’affiche.
5. DansBasic details (Détails de base), renseignez les champs Security group name (Nom du groupe de sécurité) et Description. Pour VPC, choisissez le VPC dans lequel vous souhaitez créer votre cluster de bases de données.
6. Dans Inbound rules (Règles entrantes), choisissez Add rule (Ajouter une règle).
   a. Pour Type, choisissez Custom TCP (TCP personnalisé).
   b. Pour Port range (Plage de ports), entrez la valeur de port à utiliser pour votre cluster de bases de données.
   c. Pour Source, choisissez un nom de groupe de sécurité ou tapez la plage d’adresses IP (valeur CIDR) à partir de laquelle vous accédez au cluster de bases de données. Si vous choisissez My IP (Mon IP), l’accès au cluster de bases de données est autorisé à partir de l’adresse IP détectée dans votre navigateur.
7. Si vous devez ajouter d’autres adresses IP ou des plages de ports différentes, choisissez Add rule (Ajouter une règle) et saisissez les informations relatives à la règle.
8. (Facultatif) Dans Outbound rules (Règles sortantes), ajoutez des règles pour le trafic sortant. Par défaut, tous les trafics sortant sont autorisés.
9. Sélectionnez Créer un groupe de sécurité.

Vous pouvez utiliser le groupe de sécurité VPC que vous venez de créer pour votre cluster de bases de données lors de sa création.

> **Note**
>
> Si vous utilisez un VPC par défaut, un groupe de sous-réseaux par défaut couvrant l’ensemble des sous-réseaux du VPC a déjà été créé pour vous. Lorsque vous créez un

cluster de bases de données, vous pouvez sélectionner le VPC par défaut et utiliser default (par défaut) pour DB Subnet Group (Groupe de sous-réseaux de base de données).

Une fois que vous avez terminé la configuration requise, vous pouvez créer un cluster de bases de données en utilisant votre configuration et votre groupe de sécurité en suivant les instructions de la section Création d'un cluster de bases de données Amazon Aurora. Pour plus d'informations sur la création d'un cluster de bases de données utilisant un moteur de base de données spécifique, consultez Mise en route avec Amazon Aurora.

Mise en route avec Amazon Aurora

Dans cette section, vous découvrirez comment créer un cluster de bases de données Aurora par l'intermédiaire d'Amazon RDS.

Les procédures suivantes sont des didacticiels qui présentent les bases de la mise en route avec Aurora. Les sections suivantes présentent des concepts et des procédures Aurora plus avancées, comme les différents types de point de terminaison et la manière de mettre à l'échelle des clusters Aurora.

> **Important**
>
> Vous devez réaliser les tâches de la section Configuration de votre environnement pour Amazon Aurora avant de créer un cluster de base de données ou de vous y connecter.

Rubriques

- Création et connexion à un cluster de bases de données Aurora MySQL
- Création et connexion à un cluster de bases de données Aurora PostgreSQL
- Didacticiel : Créer un serveur web et une cluster de base de données Amazon Aurora

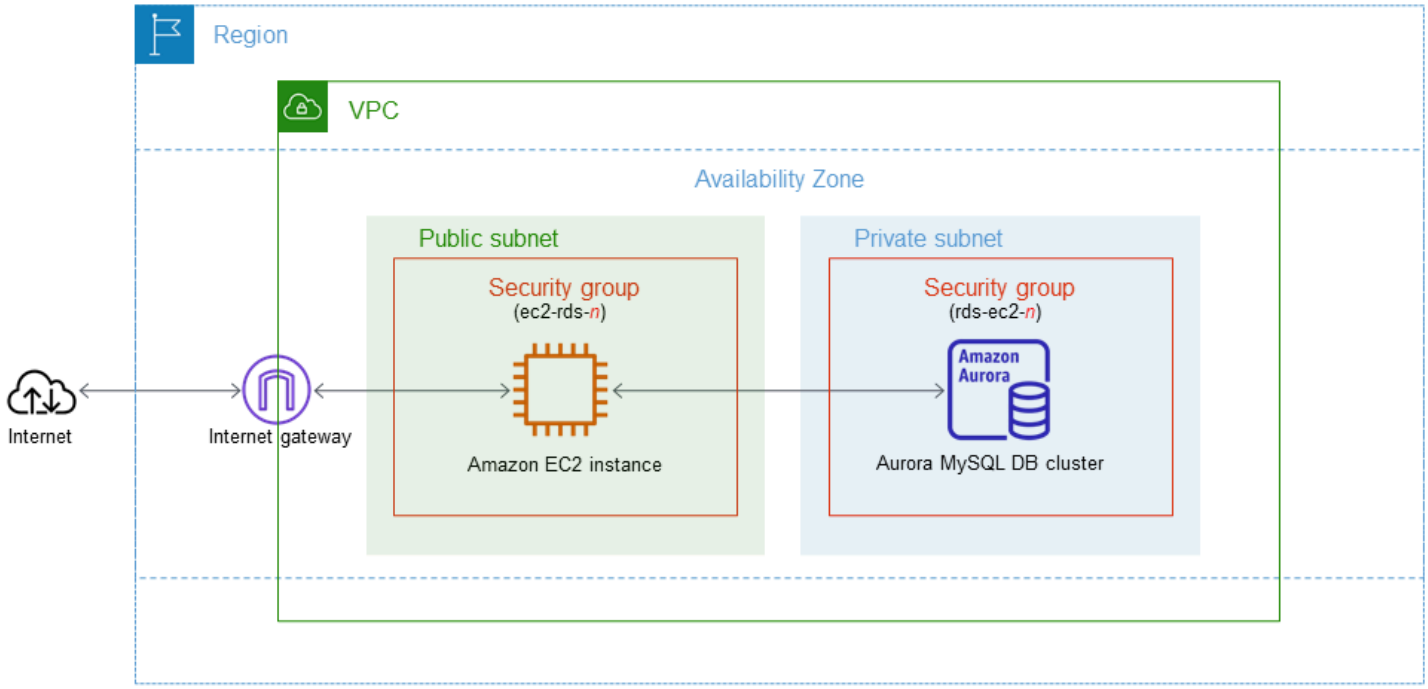
Création et connexion à un cluster de bases de données Aurora MySQL

Ce didacticiel crée une EC2 instance et un cluster de base de données Aurora MySQL. Le didacticiel explique comment accéder au cluster de base de données depuis l' EC2 instance à l'aide d'un client MySQL standard. En tant que bonne pratique, ce didacticiel crée un cluster de bases de donnée privé dans un cloud privé virtuel (VPC). Dans la plupart des cas, les autres ressources du même VPC, telles que les EC2 instances, peuvent accéder au cluster de base de données, mais les ressources extérieures au VPC ne peuvent pas y accéder.

Une fois le tutoriel terminé, chaque zone de disponibilité de votre VPC comporte un sous-réseau public et un sous-réseau privé. Dans une zone de disponibilité, l' EC2 instance se trouve dans le sous-réseau public et l'instance de base de données dans le sous-réseau privé.

> **Important**
>
> La création d'un AWS compte est gratuite. Cependant, en suivant ce didacticiel, les AWS ressources que vous utilisez peuvent vous coûter cher. Vous pouvez supprimer ces ressources après avoir terminé le didacticiel si elles ne sont plus nécessaires.

Le diagramme suivant affiche la configuration obtenue au terme de ce didacticiel.



Ce didacticiel vous permet de créer vos ressources en utilisant l'une des méthodes suivantes :

1. Utilisez le AWS Management Console - Étape 1 : créer une EC2 instance et Étape 2 : Créer un cluster de bases de données Aurora MySQL
2. CloudFormation À utiliser pour créer l'instance de base de données et l' EC2 instance - (Facultatif) Créez un VPC, une EC2 instance et un cluster Aurora MySQL à l'aide de CloudFormation

La première méthode utilise Création facile pour créer un cluster de bases de données Aurora MySQL privé avec la AWS Management Console. Ici, vous ne spécifiez que le type de moteur de base de données, la taille de l'instance de base de données et l'identifiant du cluster de bases de données. L'option Easy create (Création facile) utilise les paramètres par défaut pour les autres options de configuration.

Lorsque vous utilisez Création standard à la place, vous pouvez spécifier davantage d'options de configuration lorsque vous créez un cluster de bases de données. Ces options incluent les paramètres de disponibilité, de sécurité, de sauvegarde et de maintenance. Pour créer un cluster de bases de données public, vous devez utiliser Création standard. Pour plus d'informations, consultez the section called "Création d'un cluster de bases de données".

Rubriques

Prérequis

Avant de commencer, suivez les étapes détaillées dans les sections suivantes :

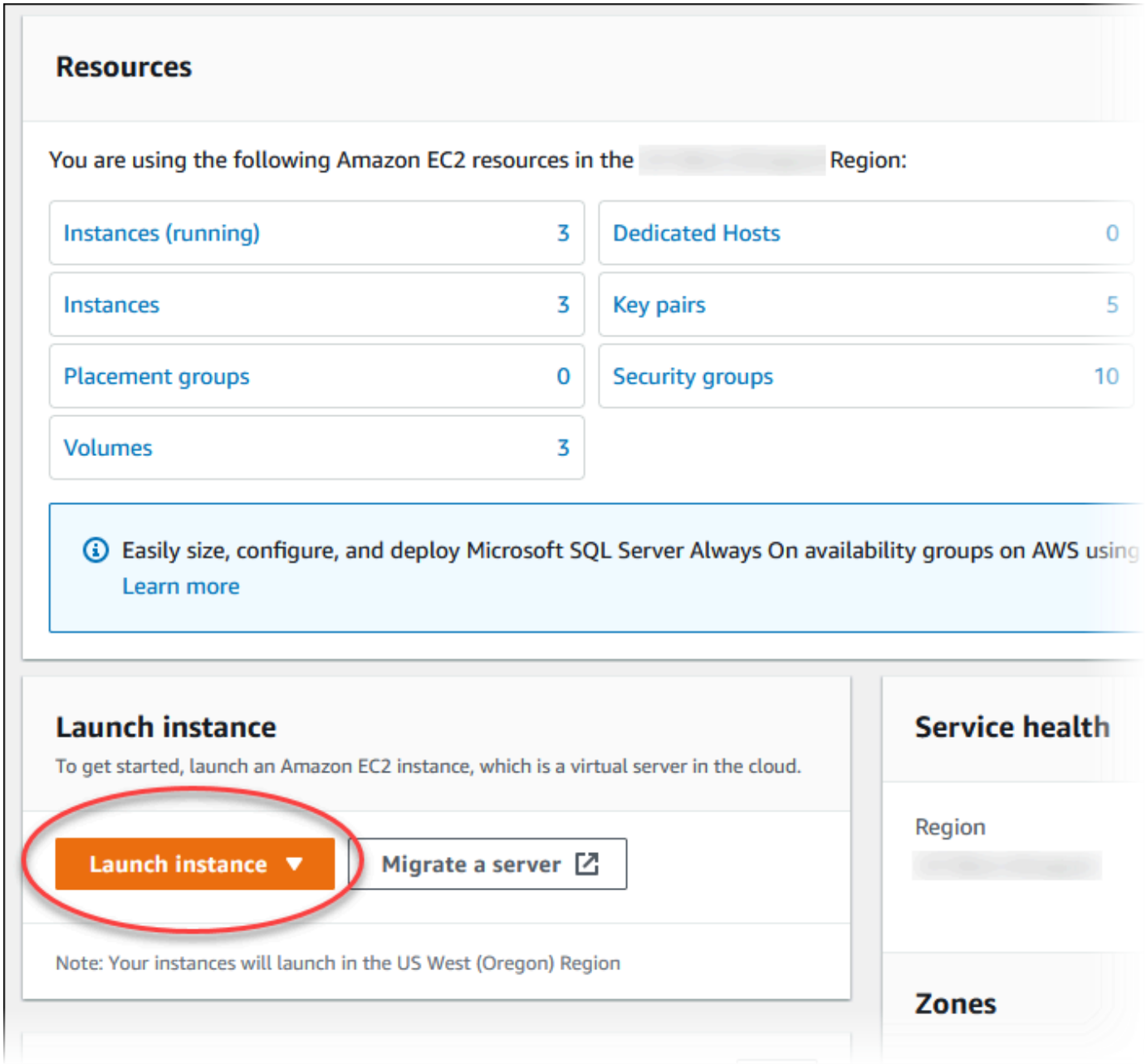
- Inscrivez-vous pour un Compte AWS
- Création d'un utilisateur doté d'un accès administratif

Étape 1 : créer une EC2 instance

Créez une EC2 instance Amazon que vous utiliserez pour vous connecter à votre base de données.

Pour créer une EC2 instance

1. Connectez-vous à la EC2 console Amazon AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le coin supérieur droit de AWS Management Console, choisissez l'instance Région AWS dans laquelle vous souhaitez créer l' EC2 instance.
3. Choisissez EC2 Dashboard, puis Launch instance, comme illustré dans l'image suivante.

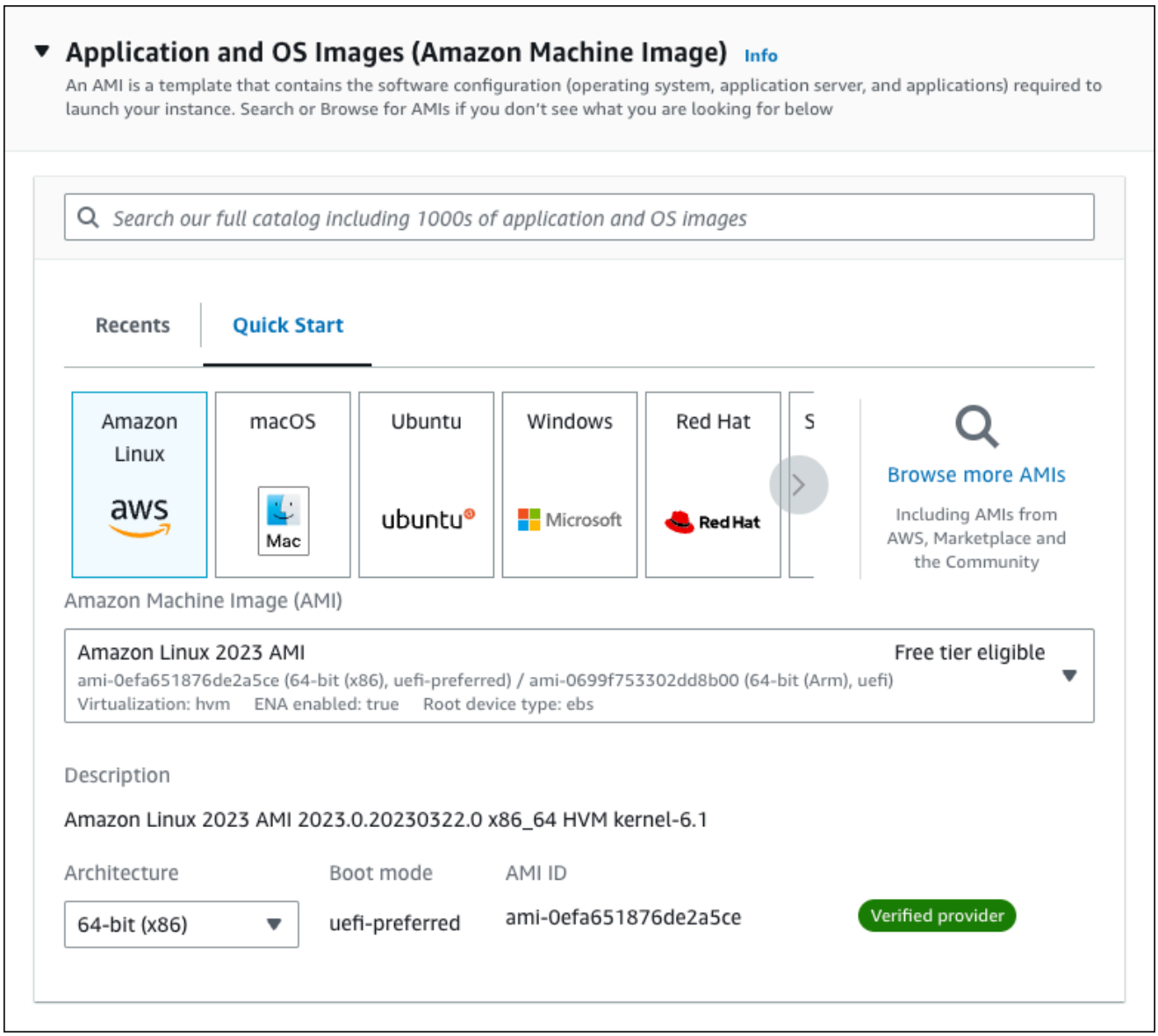


La page Lancer une instance s'ouvre.

4. Choisissez les paramètres suivants sur la page Lancer une instance.

   a. Sous Name and tags (Nom et identifications), pour Name (Nom), saisissez **`ec2-database-connect`**.

   b. Sous Application et images OS (Amazon Machine Image), choisissez Amazon Linux, puis Amazon Linux 2023 AMI. Conservez les sélections par défaut pour les autres choix.



c. Sous Instance type (Type d'instance), choisissez t2.micro.

d. Sous Key pair (login) [Paire de clés (connexion)], choisissez une valeur Key pair name (Nom de paire de clés) pour utiliser une paire de clés existante. Pour créer une nouvelle paire de clés pour l' EC2 instance Amazon, choisissez Create new key pair, puis utilisez la fenêtre Create key Pair pour la créer.

   Pour plus d'informations sur la création d'une nouvelle paire de clés, consultez la section Créer une paire de clés dans le guide de EC2 l'utilisateur Amazon.

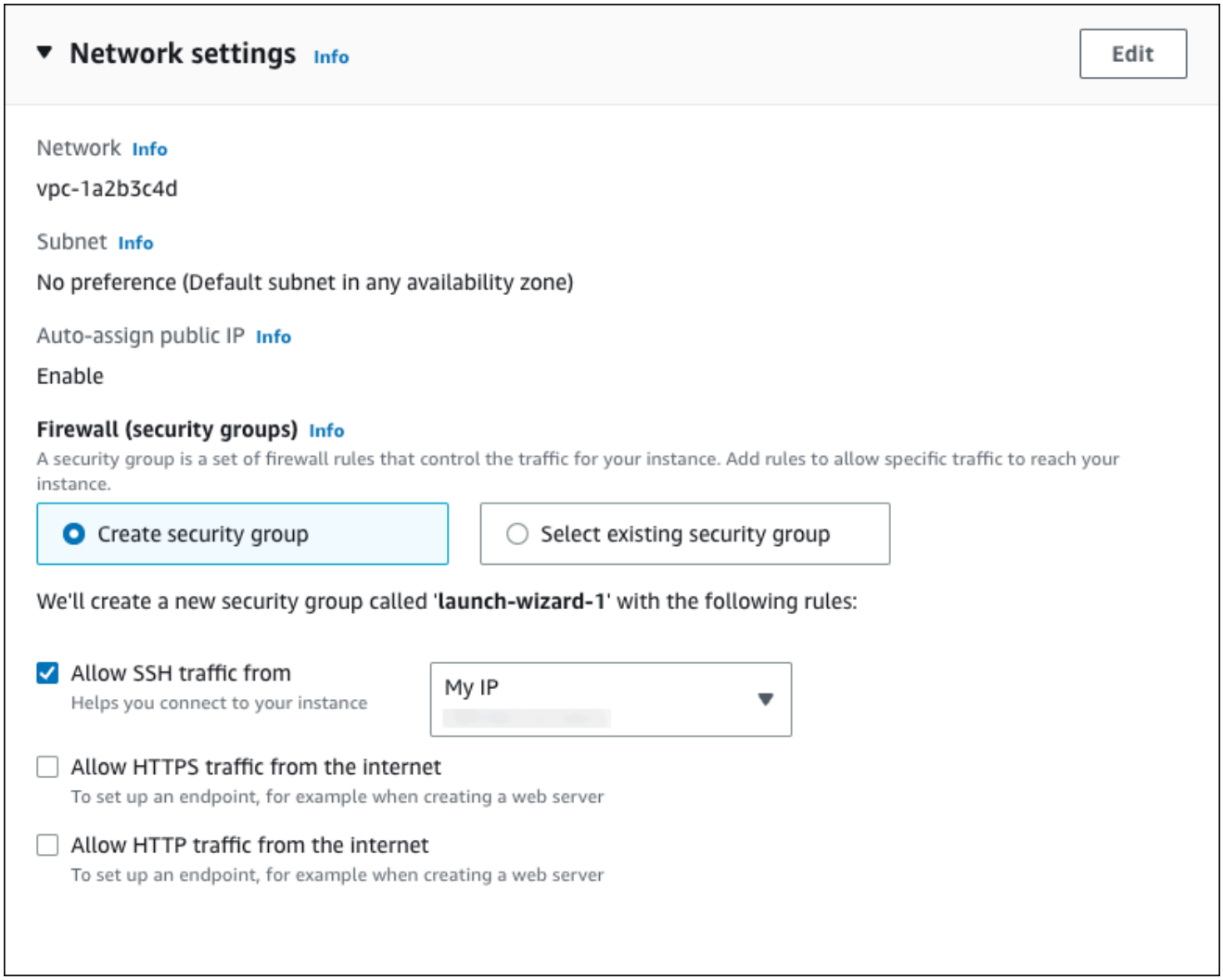
e. Pour Autoriser le trafic SSH dans les paramètres réseau, choisissez la source des connexions SSH à l' EC2instance.

Vous pouvez choisir My IP (Mon IP) si l’adresse IP affichée est correcte pour les connexions SSH. Sinon, vous pouvez déterminer l'adresse IP à utiliser pour vous connecter aux EC2 instances de votre VPC à l'aide de Secure Shell (SSH). Pour déterminer votre adresse IP publique, dans une autre fenêtre ou un autre onglet du navigateur, vous pouvez utiliser le service à l'adresse https://checkip.amazonaws.com. Exemple d’adresse IP : 192.0.2.1/32.

Dans de nombreux cas, votre connexion s’effectue via un fournisseur de services Internet (FSI) ou derrière votre pare-feu sans adresse IP statique. Si tel est le cas, assurez-vous de déterminer la plage d’adresses IP utilisées par les ordinateurs clients.

> **Warning**
>
> Si vous utilisez `0.0.0.0/0` l'accès SSH, vous permettez à toutes les adresses IP d'accéder à vos EC2 instances publiques via SSH. Cette approche est acceptable pour une brève durée dans un environnement de test, mais n’est pas sécurisée pour les environnements de production. En production, autorisez uniquement une adresse IP ou une plage d'adresses spécifiques à accéder à vos EC2 instances via SSH.

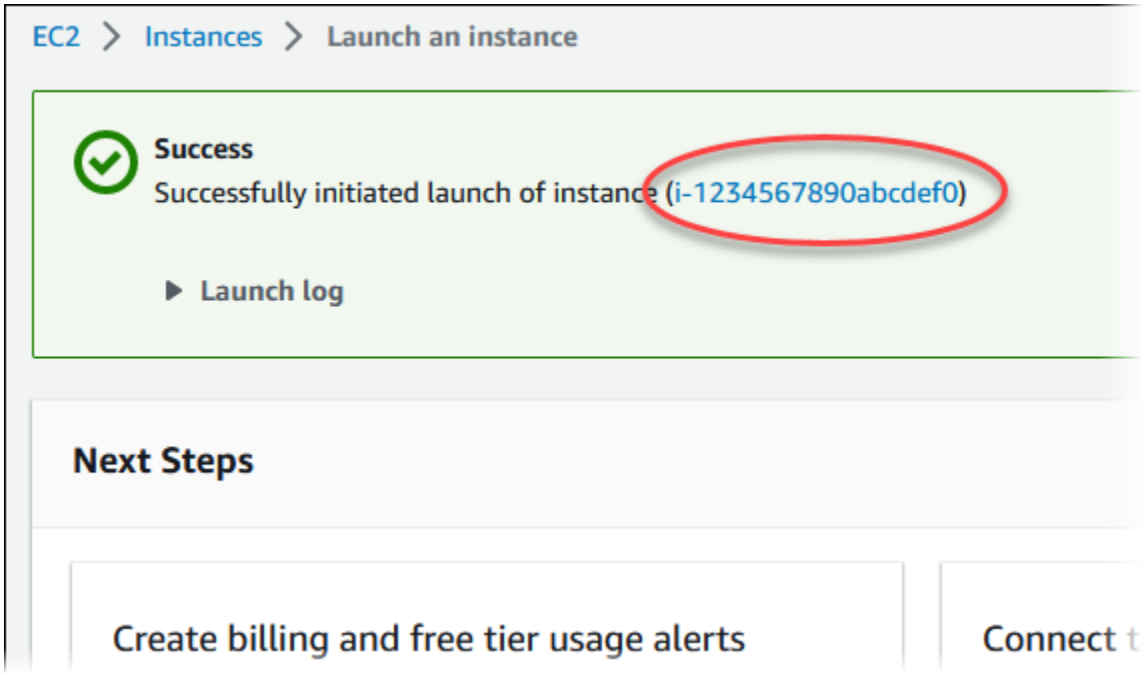
L’image suivante présente un exemple de la section Paramètres réseau.



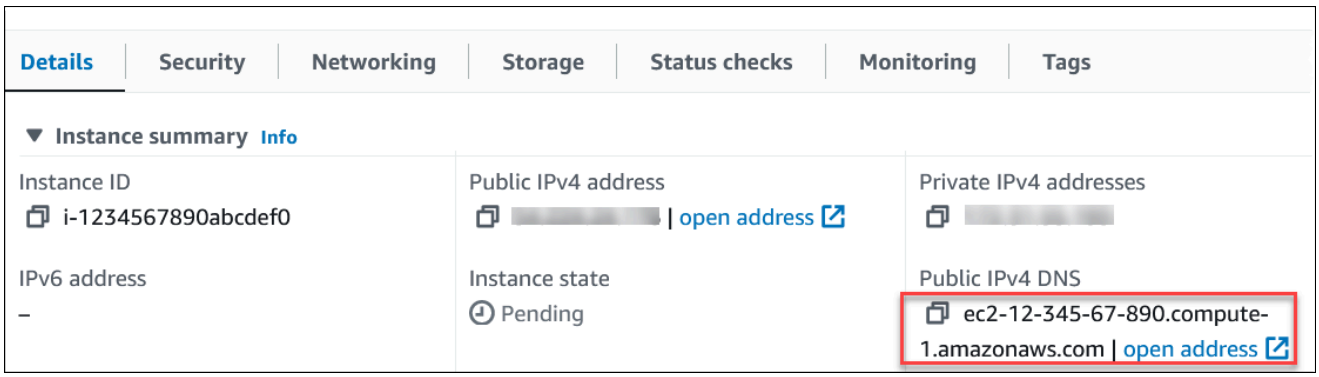
f. Laissez les valeurs par défaut pour les autres sections.

g. Consultez un résumé de la configuration de votre EC2 instance dans le panneau Résumé et, lorsque vous êtes prêt, choisissez Launch instance.

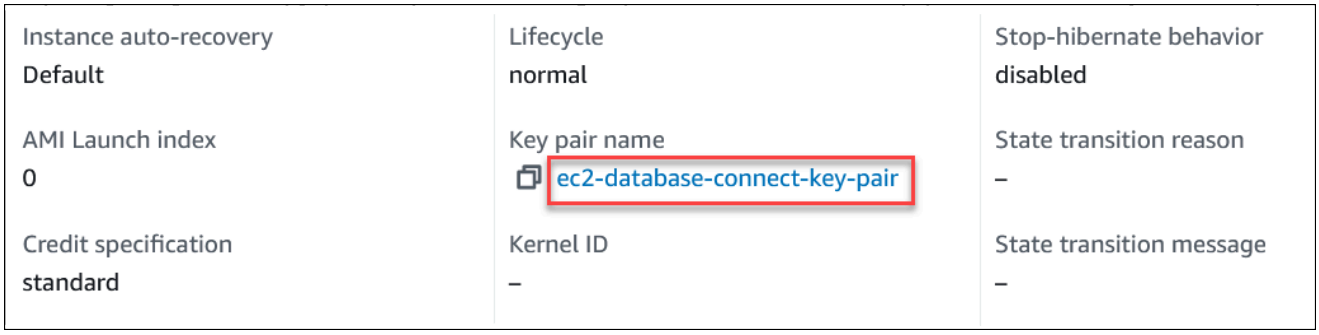
5. Sur la page État du lancement, notez l'identifiant de votre nouvelle EC2 instance, par exemple :`i-1234567890abcdef0`.



6. Choisissez l'identifiant d' EC2 instance pour ouvrir la liste des EC2 instances, puis sélectionnez votre EC2 instance.
7. Dans l’onglet Détails, notez les valeurs suivantes. Vous en aurez besoin lorsque vous vous connecterez via SSH :
   a. Dans Résumé de l'instance, notez la valeur du IPv4 DNS public.



   b. Dans Détails de l’instance, notez la valeur pour Nom de la paire de clés.



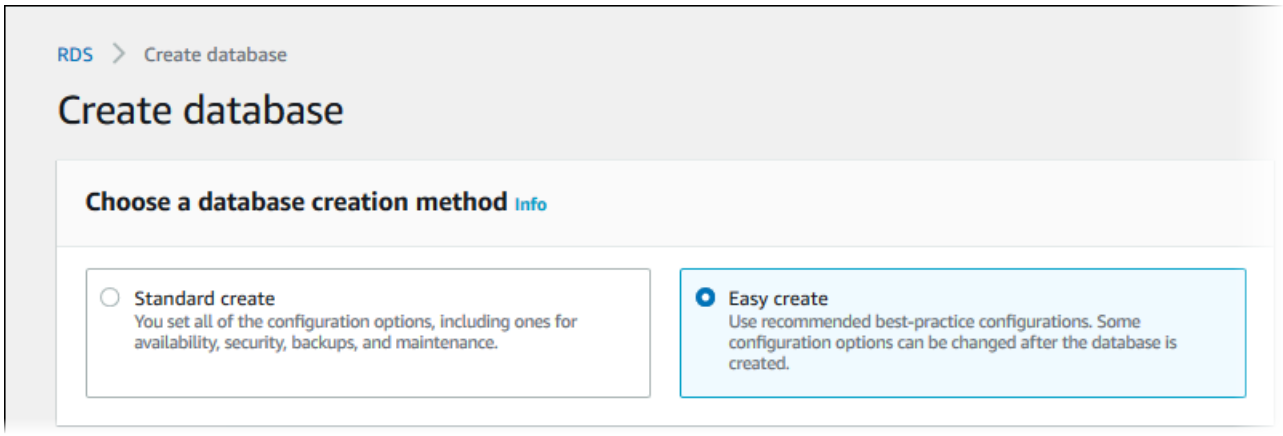
8. Attendez que l'état Instance de votre EC2 instance soit défini sur En cours d'exécution avant de continuer.

Étape 2 : Créer un cluster de bases de données Aurora MySQL

Dans cet exemple, vous utilisez l'option Création facile pour créer un cluster de bases de données Aurora MySQL avec une classe d'instance de base de données db.r6g.large.

Pour créer un cluster de bases de données Aurora MySQL avec l'option Création facile

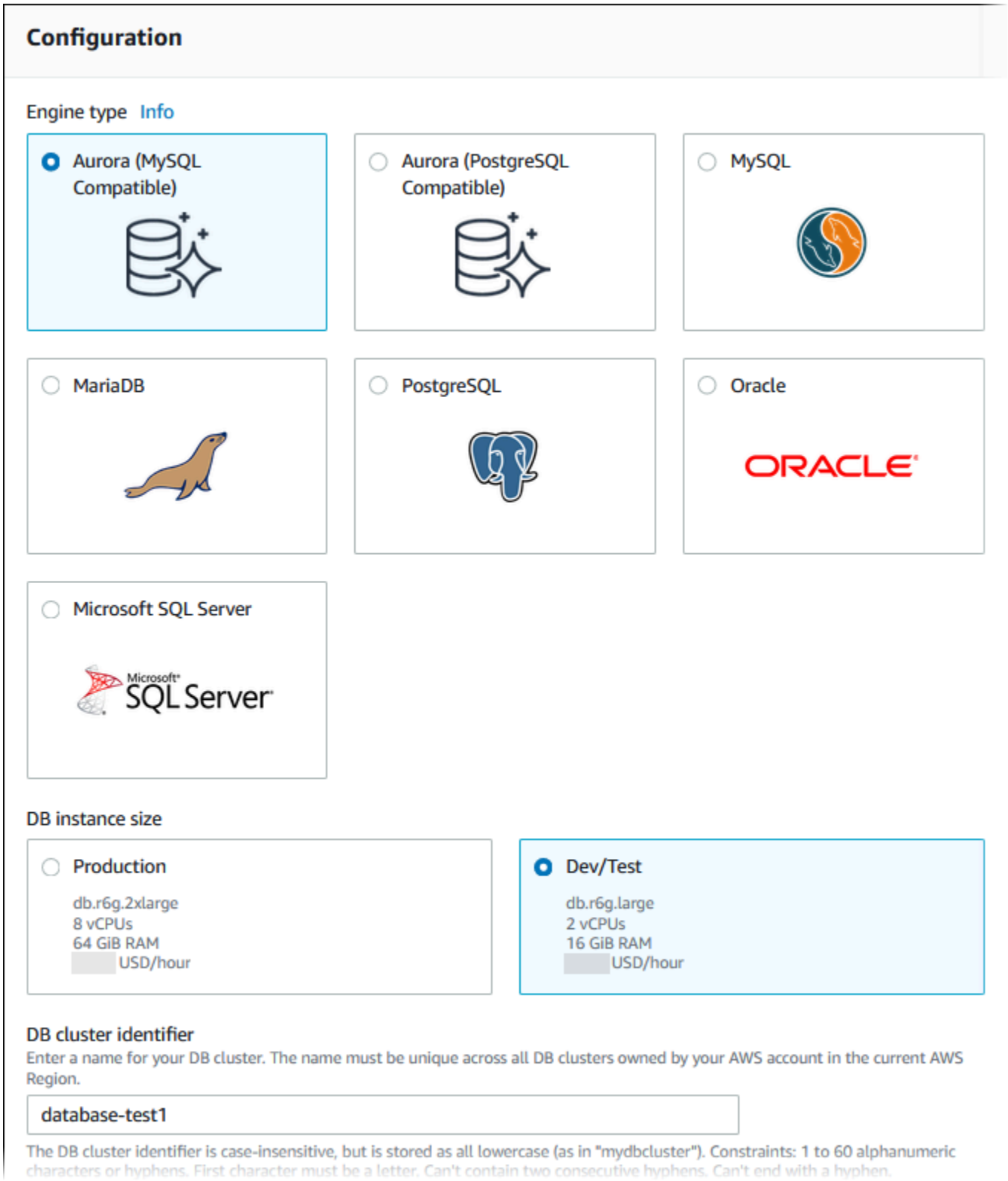
1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le coin supérieur droit de la console Amazon RDS, choisissez celui Région AWS dans lequel vous souhaitez créer le cluster de base de données.
3. Dans le panneau de navigation, choisissez Databases (Bases de données).
4. Choisissez Create database (Créer une base de données) et veillez à choisir Easy create (Création facile).



5. Dans Configuration, choisissez Aurora (compatible avec MySQL) pour Type de moteur.
6. Pour DB instance size (Taille de l'instance de base de données), choisissez Dev/Test.

7. Pour Identifiant du cluster de bases de données, saisissez **`database-test1`**.

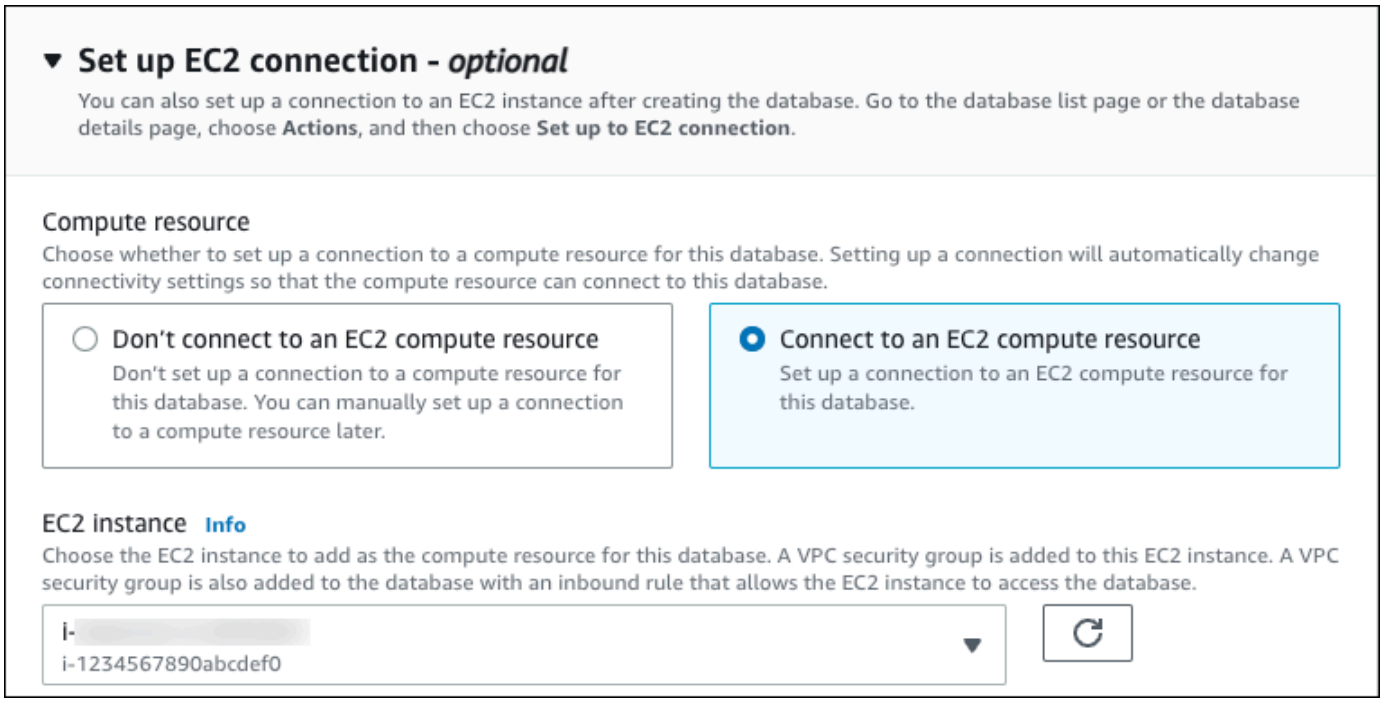
   La page Create database (Créer une base de données) doit ressembler à l'image suivante.



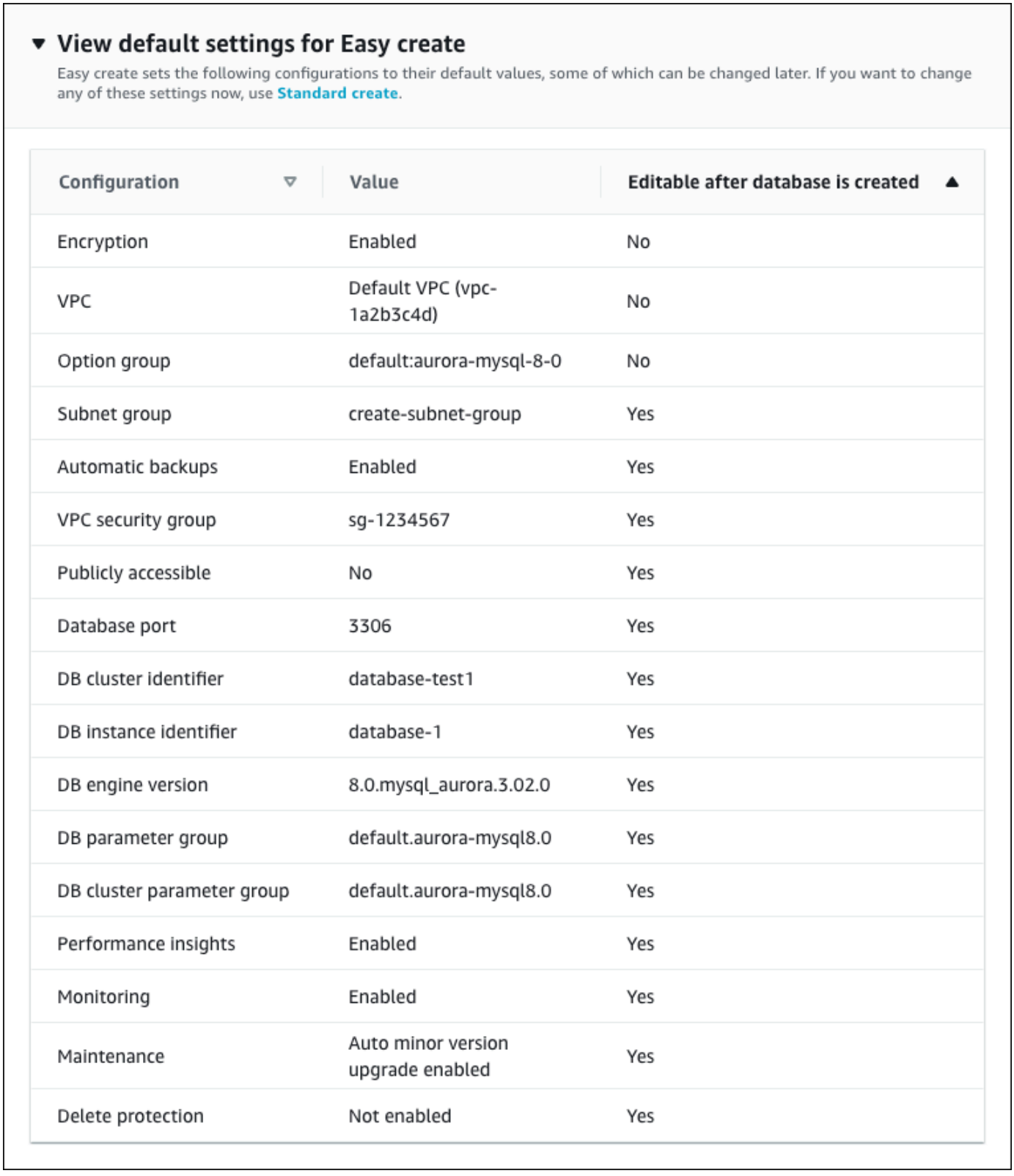
8. Pour Nom d’utilisateur principal, saisissez un nom pour l’utilisateur principal ou conservez le nom par défaut.
9. Pour utiliser un mot de passe principal généré automatiquement pour le cluster de bases de données, sélectionnez Générer automatiquement un mot de passe.

   Pour saisir votre mot de passe principal, veillez à ce que la case Générer automatiquement un mot de passe soit décochée, puis saisissez le même mot de passe dans Mot de passe principal et Confirmer le mot de passe.
10. Pour configurer une connexion avec l' EC2 instance que vous avez créée précédemment, ouvrez Configurer la EC2 connexion - facultatif.

    Sélectionnez Se connecter à une ressource EC2 de calcul. Choisissez l'EC2 instance que vous avez créée précédemment.



11. Ouvrez Afficher les paramètres par défaut pour Création facile.



View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:aurora-mysql-8-0	No
Subnet group	create-subnet-group	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	3306	Yes
DB cluster identifier	database-test1	Yes
DB instance identifier	database-1	Yes
DB engine version	8.0.mysql_aurora.3.02.0	Yes
DB parameter group	default.aurora-mysql8.0	Yes
DB cluster parameter group	default.aurora-mysql8.0	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

Vous pouvez examiner les paramètres par défaut utilisés quand l'option Easy create (Création facile) est activée. La colonne Modifiable après la création de la base de données indique les options que vous pouvez modifier après avoir créé la base de données.

- Si un paramètre contient Non dans cette colonne et que vous souhaitez un paramètre différent, vous pouvez utiliser Création standard pour créer le cluster de bases de données.
- Si un paramètre contient Oui dans cette colonne et que vous souhaitez un paramètre différent, vous pouvez utiliser Création standard pour créer le cluster de bases de données ou vous pouvez modifier le cluster de bases de données après l'avoir créé pour modifier le paramètre.

12. Choisissez Créer une base de données.

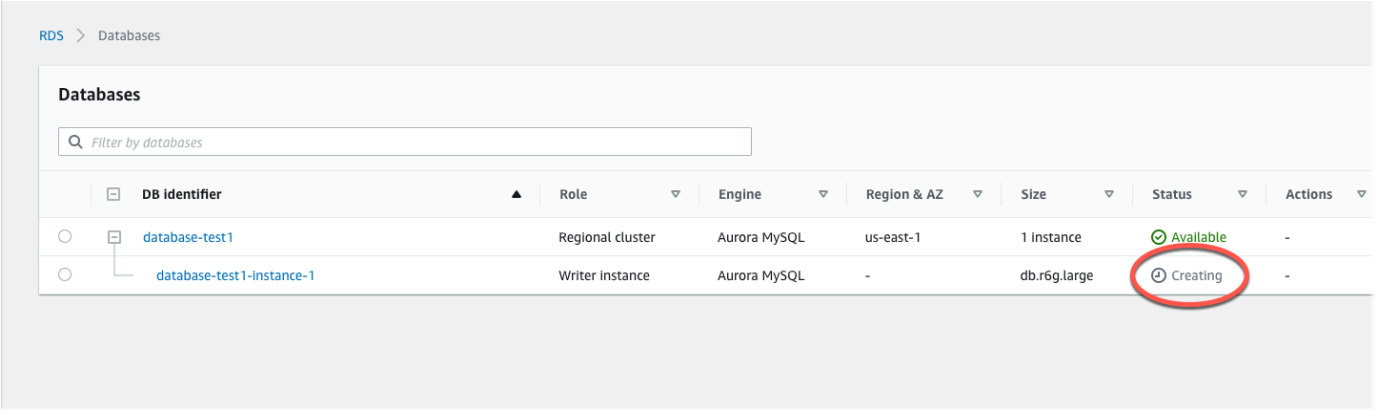
    Pour afficher le nom d'utilisateur principal et le mot de passe pour le cluster de bases de données, choisissez Afficher les détails des informations d'identification.

    Vous pouvez utiliser le nom d'utilisateur et le mot de passe affichés pour vous connecter au cluster de bases de données en tant qu'utilisateur principal.

    > **Important**
    >
    > Vous ne pourrez pas afficher le mot de passe de l'utilisateur principal de nouveau. Si vous ne l'enregistrez pas, il sera peut-être nécessaire de le modifier.
    > Si vous devez changer le mot de passe de l'utilisateur principal une fois le cluster de bases de données disponible, vous pouvez le faire en modifiant le cluster de bases de données. Pour plus d'informations sur la modification d'un cluster, consultez Modification d'un cluster de bases de données Amazon Aurora.

13. Dans la liste Bases de données, choisissez le nom du nouveau cluster de bases de données Aurora MySQL pour afficher ses détails.

    L'instance d'enregistreur a le statut Création en cours jusqu'à ce que le cluster de bases de données soit prêt à l'emploi.



Lorsque le statut de l'instance d'enregistreur passe à Disponible, vous pouvez vous connecter au cluster de bases de données. En fonction de la quantité de stockage et de la classe d'instance de base de données, la mise à disposition du nouveau cluster de bases de données peut prendre jusqu'à 20 minutes.

(Facultatif) Créez un VPC, une EC2 instance et un cluster Aurora MySQL à l'aide de CloudFormation

Au lieu d'utiliser la console pour créer votre VPC, votre EC2 instance et votre cluster de base de données Aurora MySQL, vous pouvez les utiliser CloudFormation pour provisionner AWS des ressources en traitant l'infrastructure comme du code. Pour vous aider à organiser vos AWS ressources en unités plus petites et plus faciles à gérer, vous pouvez utiliser la fonctionnalité de pile CloudFormation imbriquée. Pour plus d'informations, consultez les sections Création d'une pile sur la CloudFormation console et Utilisation de piles imbriquées.

> **Important**
>
> CloudFormation est gratuit, mais les ressources qui en CloudFormation découlent sont vivantes. Les frais d'utilisation standard pour ces ressources vous sont facturés jusqu'à ce que vous les résiliez. Pour plus d'informations, consultez Tarification Amazon Aurora.

Pour créer vos ressources à l'aide de la CloudFormation console, procédez comme suit :

- Étape 1 : Téléchargez le CloudFormation modèle
- Étape 2 : configurez vos ressources à l'aide de CloudFormation

Téléchargez le CloudFormation modèle

Un CloudFormation modèle est un fichier texte JSON ou YAML qui contient les informations de configuration relatives aux ressources que vous souhaitez créer dans la pile. Ce modèle crée également un VPC et un hôte bastion pour vous, en plus du cluster Aurora.

Pour télécharger le fichier modèle, ouvrez le lien suivant, CloudFormation Modèle Aurora MySQL.

Sur la page Github, cliquez sur le bouton Télécharger le fichier brut pour enregistrer le modèle de fichier YAML.

Configurez vos ressources à l'aide de CloudFormation

> **Note**
>
> Avant de commencer ce processus, assurez-vous que vous disposez d'une paire de clés pour une EC2 instance dans votre Compte AWS. Pour plus d'informations, consultez les paires de EC2 clés Amazon et les instances Linux.

Lorsque vous utilisez le CloudFormation modèle, vous devez sélectionner les paramètres appropriés pour vous assurer que vos ressources sont créées correctement. Procédez de la façon suivante :

1. Connectez-vous à la CloudFormation console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudformation.
2. Sélectionnez Créer une pile.
3. Dans la section Spécifier le modèle, sélectionnez Charger un fichier de modèle à partir de votre ordinateur, puis cliquez sur Suivant.
4. Sur la page Spécifier les détails de la pile, saisissez les paramètres suivants :
   a. Définissez le nom de la AurMySQLTestpile sur Stack.
   b. Sous Paramètres, définissez les zones de disponibilité en sélectionnant deux zones de disponibilité.
   c. Dans Configuration de l'hôte Linux Bastion, pour Nom de la clé, sélectionnez une paire de clés pour vous connecter à votre EC2 instance.
   d. Dans les paramètres de Configuration de l’hôte bastion Linux, définissez Plage d’adresses IP autorisées sur votre adresse IP. Pour vous connecter aux EC2 instances de votre VPC à l'aide de Secure Shell (SSH), déterminez votre adresse IP publique à l'aide du service à l'adresse. https://checkip.amazonaws.com Exemple d’adresse IP : 192.0.2.1/32.

> **Warning**
>
> Si vous utilisez `0.0.0.0/0` l'accès SSH, vous permettez à toutes les adresses IP d'accéder à vos EC2 instances publiques via SSH. Cette approche est acceptable pour une brève durée dans un environnement de test, mais n'est pas sécurisée pour les environnements de production. En production, autorisez uniquement une adresse IP ou une plage d'adresses spécifiques à accéder à vos EC2 instances via SSH.

e. Dans Configuration générale de la base de données, définissez la classe d'instance de base de données sur db.r6g.large.

f. Définissez Nom de la base de données sur **`database-test1`**.

g. Pour Nom d'utilisateur principal de la base de données, entrez le nom de l'utilisateur principal.

h. Définissez Gérer le mot de passe utilisateur principal de base de données avec Secrets Manager sur `false` pour ce didacticiel.

i. Pour Mot de passe de la base de données, définissez le mot de passe de votre choix. N'oubliez pas ce mot de passe pour pouvoir effectuer les étapes suivantes du didacticiel.

j. Définissez Déploiement multi-AZ sur `false`.

k. Conservez les valeurs par défaut de tous les autres paramètres. Cliquez sur Suivant pour continuer.

5. Sur la page Configurer les options de pile, conservez toutes les options par défaut. Cliquez sur Suivant pour continuer.
6. Sur la page Vérification de la pile, sélectionnez Soumettre après avoir vérifié les options de la base de données et de l'hôte bastion Linux.

Une fois le processus de création des piles terminé, visualisez les piles avec leurs noms BastionStacket AMSNS pour noter les informations dont vous avez besoin pour vous connecter à la base de données. Pour plus d'informations, consultez la section Affichage des données et des ressources de la CloudFormation pile sur le AWS Management Console.

Étape 3 : Se connecter à un cluster de bases de données Aurora MySQL

Vous pouvez utiliser n'importe quelle application client SQL standard pour vous connecter au cluster de bases de données. Dans cet exemple, vous vous connectez à un cluster de bases de données Aurora MySQL en utilisant le client de ligne de commande mysql.

Pour se connecter à un cluster de bases de données Aurora MySQL

1. Trouvez le point de terminaison (nom DNS) et le numéro de port de l'instance d'enregistreur pour votre cluster de bases de données.

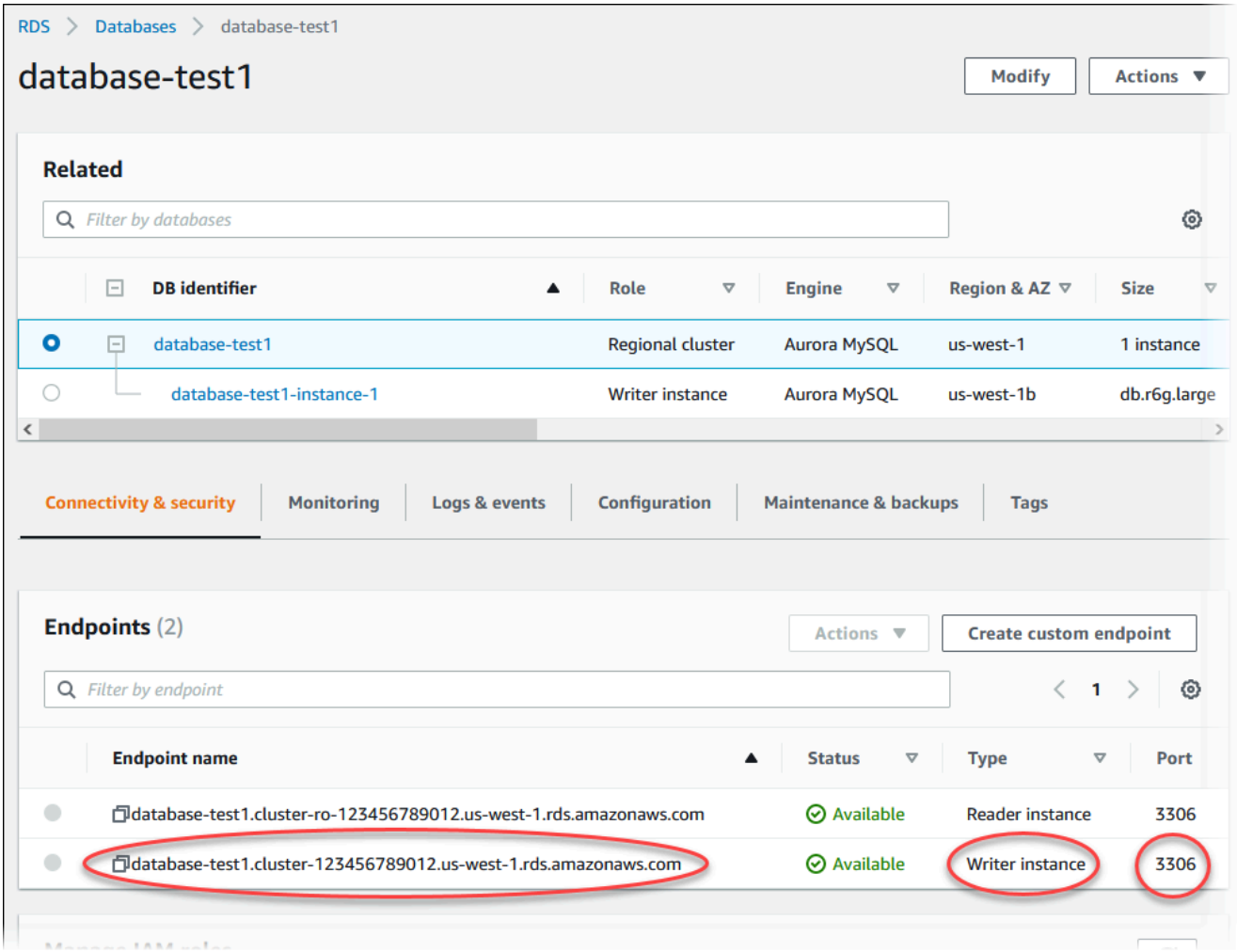
   a. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.

   b. Dans le coin supérieur droit de la console Amazon RDS, choisissez la Région AWS pour le cluster de bases de données.

   c. Dans le panneau de navigation, choisissez Databases (Bases de données).

   d. Choisissez le nom du cluster de bases de données Aurora MySQL pour en afficher les détails.

   e. Dans l'onglet Connectivité et sécurité, copiez le point de terminaison de l'instance d'enregistreur. Notez également le numéro du port. Vous avez besoin du point de terminaison et du numéro de port pour vous connecter au cluster de bases de données.



2. Connectez-vous à l' EC2 instance que vous avez créée précédemment en suivant les étapes décrites dans la section Connexion à votre instance Linux dans le guide de EC2 l'utilisateur Amazon.

   Nous vous recommandons de vous connecter à votre EC2 instance via SSH. Si l'utilitaire client SSH est installé sur Windows, Linux ou Mac, vous pouvez vous connecter à l'instance à l'aide du format de commande suivant :

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   Supposons, par exemple, qu'`ec2-database-connect-key-pair.pem`il soit `/dir1` stocké sous Linux et que le IPv4 DNS public de votre EC2 instance soit`ec2-12-345-678-90.compute-1.amazonaws.com`. Votre commande SSH se présenterait ensuite comme suit :

   ```
   ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
   user@ec2-12-345-678-90.compute-1.amazonaws.com
   ```

3. Obtenez les dernières corrections de bogues et mises à jour de sécurité en mettant à jour le logiciel de votre EC2 instance. Pour cela, utilisez la commande suivante.

   > **Note**
   >
   > L'option `-y` installe les mises à jour sans demander de confirmation. Pour examiner les mises à jour avant de les installer, omettez cette option.

   ```
   sudo dnf update -y
   ```

4. Pour installer le client de ligne de commande mysql depuis MariaDB sur Amazon Linux 2023, exécutez la commande suivante :

   ```
   sudo dnf install mariadb105
   ```

5. Connectez-vous à un cluster de bases de données Aurora MySQL. Par exemple, saisissez la commande suivante. Cette action vous permet de vous connecter au cluster de bases de données Aurora MySQL à l'aide du client MySQL.

Remplacez le point de terminaison de votre instance d'enregistreur pour *endpoint* et remplacez le nom d'utilisateur principal que vous avez utilisé pour *admin*. Indiquez le mot de passe principal que vous avez utilisé lorsque vous êtes invité à entrer un mot de passe.

```
mysql -h endpoint -P 3306 -u admin -p
```

Après avoir entré le mot de passe pour l'utilisateur, le résultat suivant devrait normalement s'afficher.

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 217
Server version: 8.0.23 Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]>
```

Pour plus d'informations sur la connexion à un cluster de bases de données Aurora MySQL, consultez Connexion à un cluster de bases de données Amazon Aurora MySQL. Si vous ne pouvez pas vous connecter à votre cluster de bases de données, consultez Impossible de se connecter à l'instance de base de données Amazon RDS.

Pour des raisons de sécurité, une bonne pratique consiste à recommander d'utiliser des connexions chiffrées. N'utilisez une connexion MySQL non chiffrée que quand le client et le serveur sont dans le même VPC et que le réseau est approuvé. Pour plus d'informations sur l'utilisation de connexions chiffrées, consultez Connexion à Aurora MySQL avec SSL.

6. Exécutez des commandes SQL.

   Par exemple, la commande SQL suivante indique la date et l'heure actuelles :

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

Étape 4 : Supprimer l' EC2 instance et le cluster de base de données

Après vous être connecté et exploré l'exemple d' EC2 instance et de cluster de base de données que vous avez créés, supprimez-les afin de ne plus être facturés pour eux.

Si vous aviez CloudFormation l'habitude de créer des ressources, ignorez cette étape et passez à l'étape suivante.

Pour supprimer l' EC2 instance

1. Connectez-vous à la EC2 console Amazon AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le panneau de navigation, choisissez Instances.
3. Sélectionnez l' EC2 instance, puis choisissez État de l'instance, Terminer l'instance.
4. Choisissez Résilier lorsque vous êtes invité à confirmer l’action.

Pour plus d'informations sur la suppression d'une EC2 instance, consultez Résilier votre instance dans le guide de EC2 l'utilisateur Amazon.

Pour supprimer un cluster de bases de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Choisissez Bases de données et sélectionnez ensuite l’instance de base de données associée au cluster de bases de données.
3. Pour Actions, choisissez Supprimer.
4. Décochez la case Créer un instantané final ?.
5. Terminez la confirmation et choisissez Supprimer.

Une fois que vous avez supprimé toutes les instances de base de données associées à un cluster de bases de données, ce dernier est automatiquement supprimé.

(Facultatif) Supprimez l' EC2 instance et le cluster de base de données créés avec CloudFormation

Si vous aviez l'habitude de CloudFormation créer des ressources, supprimez la CloudFormation pile après vous être connecté et exploré l'exemple d' EC2 instance et de cluster de base de données, afin de ne plus être facturé pour ces éléments.

Pour supprimer les CloudFormation ressources

1. Ouvrez la CloudFormation console.
2. Sur la page Stacks de la CloudFormation console, sélectionnez la pile racine (la pile sans nom VPCStack, BastionStack ou AMSNS).
3. Sélectionnez Delete (Supprimer).
4. Sélectionnez Supprimer la pile lorsque vous êtes invité à confirmer l’action.

Pour plus d'informations sur la suppression d'une pile dans CloudFormation, voir Supprimer une pile sur la CloudFormation console dans le Guide de AWS CloudFormation l'utilisateur.

(Facultatif) Connecter votre cluster de bases de données à une fonction Lambda

Vous pouvez également connecter votre cluster de bases de données Aurora MySQL à une ressource de calcul sans serveur Lambda. Les fonctions Lambda vous permettent d’exécuter du code sans provisionner ni gérer l’infrastructure. Une fonction Lambda vous permet également de répondre automatiquement aux demandes d’exécution de code à n’importe quelle échelle, d’une douzaine d’événements par jour à des centaines par seconde. Pour plus d’informations, consultez Connexion automatique d’une fonction Lambda et d’un cluster de bases de données Aurora.

Création et connexion à un cluster de bases de données Aurora PostgreSQL

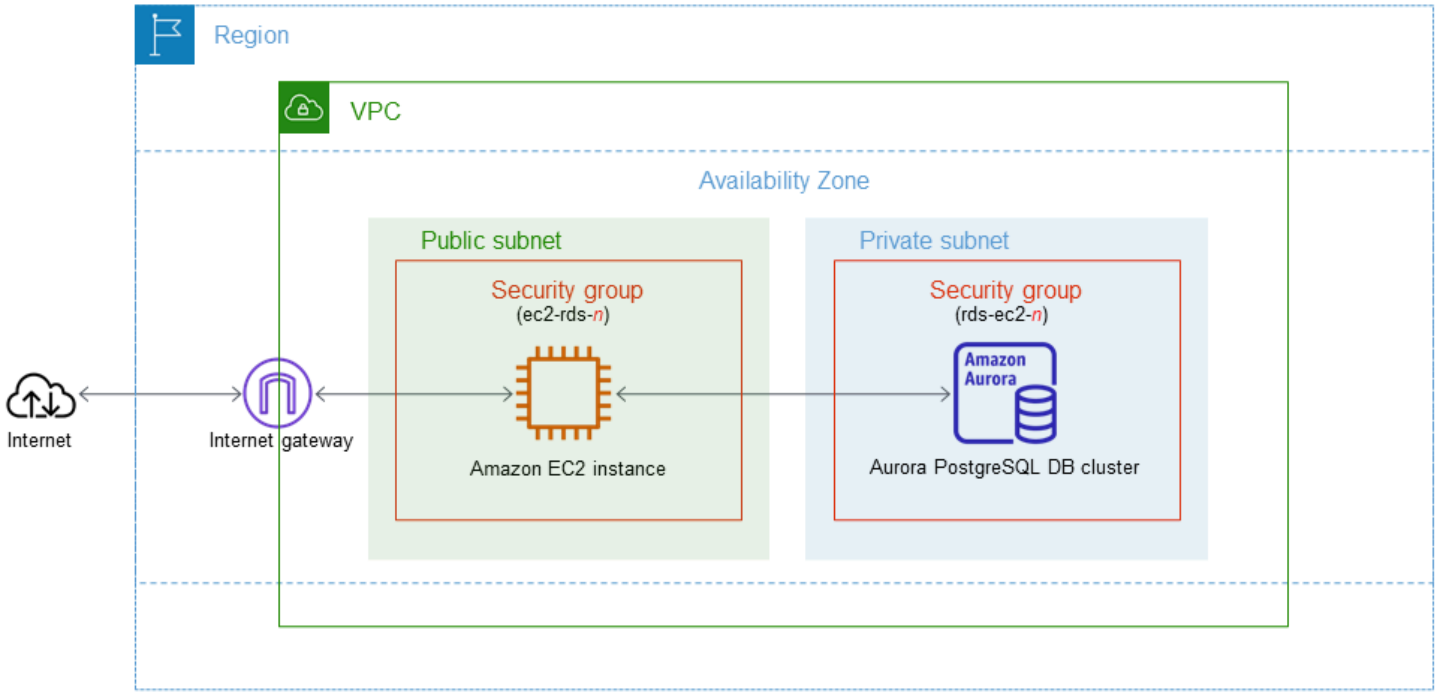
Ce didacticiel crée une EC2 instance et un cluster de base de données Aurora PostgreSQL. Le didacticiel explique comment accéder au cluster de base de données depuis l' EC2 instance à l'aide d'un client PostgreSQL standard. En tant que bonne pratique, ce didacticiel crée un cluster de bases de donnée privé dans un cloud privé virtuel (VPC). Dans la plupart des cas, les autres ressources du

même VPC, telles que les EC2 instances, peuvent accéder au cluster de base de données, mais les ressources extérieures au VPC ne peuvent pas y accéder.

Une fois le tutoriel terminé, chaque zone de disponibilité de votre VPC comporte un sous-réseau public et un sous-réseau privé. Dans une zone de disponibilité, l' EC2 instance se trouve dans le sous-réseau public et l'instance de base de données dans le sous-réseau privé.

> **Important**
>
> La création d'un AWS compte est gratuite. Cependant, en suivant ce didacticiel, les AWS ressources que vous utilisez peuvent vous coûter cher. Vous pouvez supprimer ces ressources après avoir terminé le didacticiel si elles ne sont plus nécessaires.

Le diagramme suivant affiche la configuration obtenue au terme de ce didacticiel.



Ce didacticiel vous permet de créer vos ressources en utilisant l'une des méthodes suivantes :

1. Utilisez le AWS Management Console - Étape 1 : créer une EC2 instance et Étape 2 : Créer un cluster de bases de données Aurora PostgreSQL
2. CloudFormation À utiliser pour créer l'instance de base de données et l' EC2 instance - (Facultatif) Créez un VPC, une EC2 instance et un cluster Aurora PostgreSQL à l'aide de CloudFormation

La première méthode utilise Création facile pour créer un cluster de bases de données Aurora PostgreSQL privé avec la AWS Management Console. Ici, vous ne spécifiez que le type de moteur de base de données, la taille de l'instance de base de données et l'identifiant du cluster de bases de données. L'option Easy create (Création facile) utilise les paramètres par défaut pour les autres options de configuration.

Lorsque vous utilisez Création standard à la place, vous pouvez spécifier davantage d'options de configuration lorsque vous créez un cluster de bases de données. Ces options incluent les paramètres de disponibilité, de sécurité, de sauvegarde et de maintenance. Pour créer un cluster de bases de données public, vous devez utiliser Création standard. Pour plus d'informations, consultez the section called "Création d'un cluster de bases de données".

Rubriques

Prérequis

Avant de commencer, suivez les étapes détaillées dans les sections suivantes :

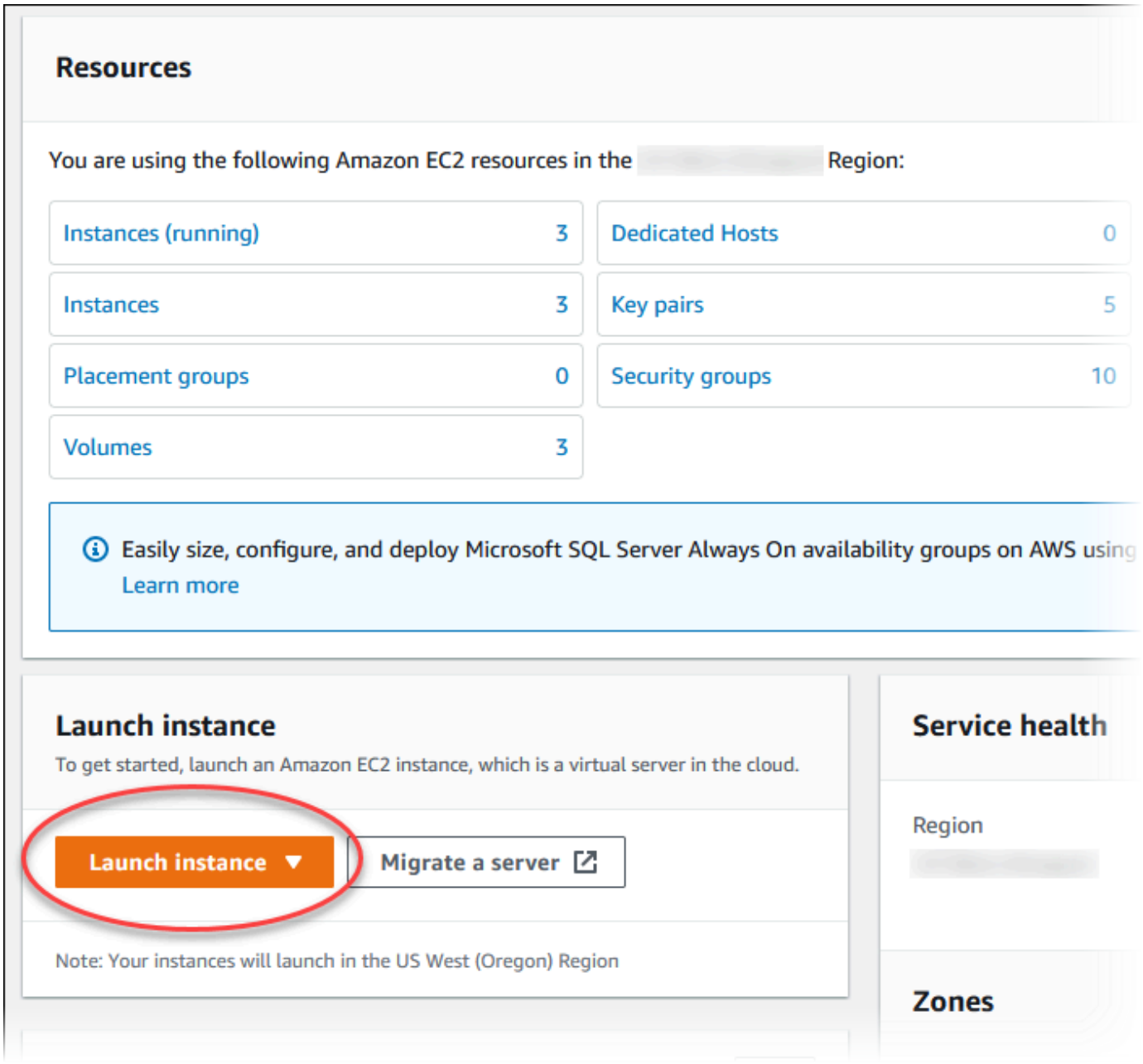
- Inscrivez-vous pour un Compte AWS
- Création d'un utilisateur doté d'un accès administratif

Étape 1 : créer une EC2 instance

Créez une EC2 instance Amazon que vous utiliserez pour vous connecter à votre base de données.

Pour créer une EC2 instance

1. Connectez-vous à la EC2 console Amazon AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le coin supérieur droit de AWS Management Console, choisissez l'instance Région AWS dans laquelle vous souhaitez créer l' EC2 instance.
3. Choisissez EC2 Dashboard, puis Launch instance, comme illustré dans l'image suivante.

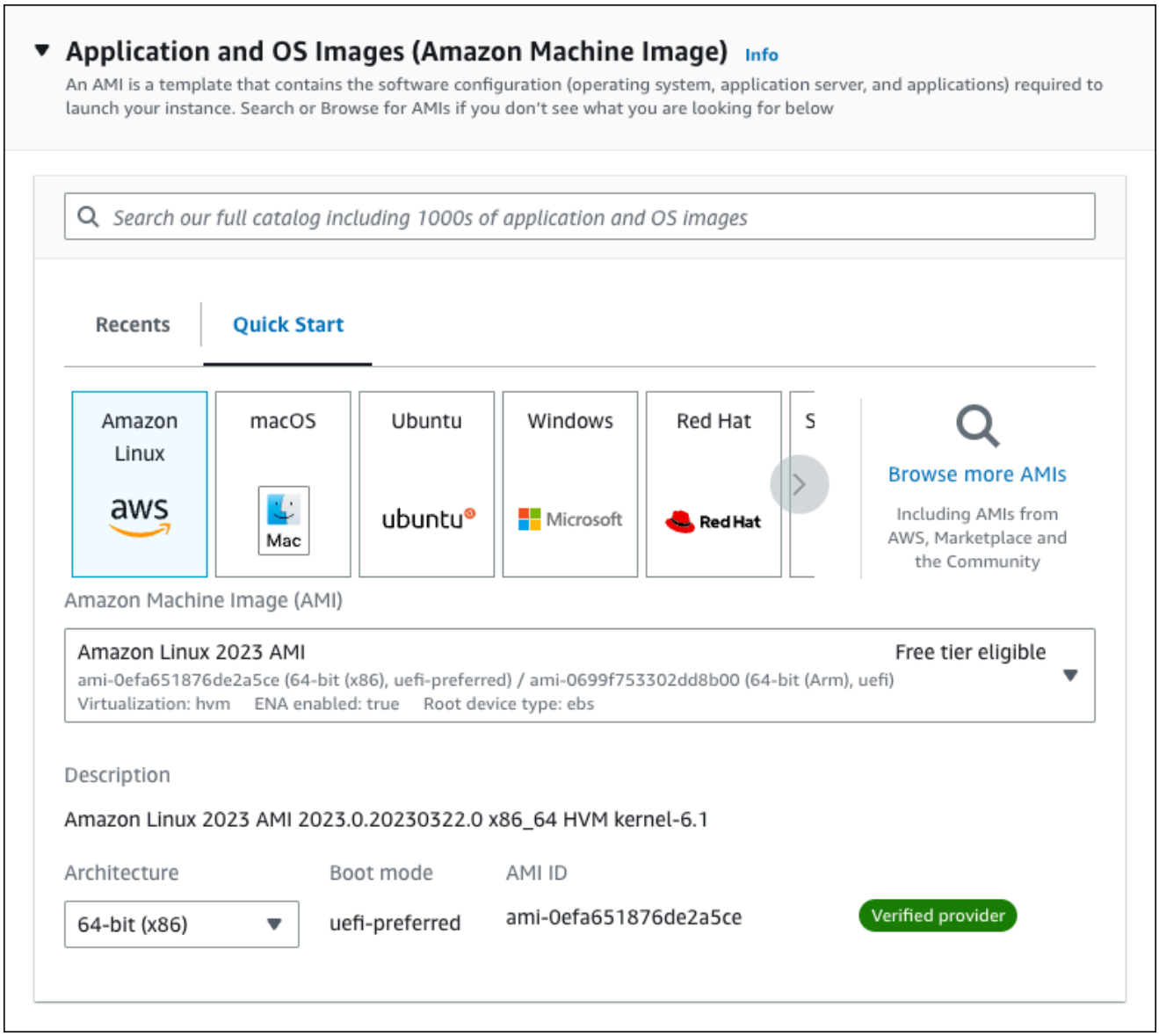


La page Lancer une instance s’ouvre.

4. Choisissez les paramètres suivants sur la page Lancer une instance.

   a. Sous Name and tags (Nom et identifications), pour Name (Nom), saisissez **`ec2-database-connect`**.

   b. Sous Application et images OS (Amazon Machine Image), choisissez Amazon Linux, puis Amazon Linux 2023 AMI. Conservez les sélections par défaut pour les autres choix.



   c. Sous Instance type (Type d'instance), choisissez t2.micro.

   d. Sous Key pair (login) [Paire de clés (connexion)], choisissez une valeur Key pair name (Nom de paire de clés) pour utiliser une paire de clés existante. Pour créer une nouvelle paire de clés pour l' EC2 instance Amazon, choisissez Create new key pair, puis utilisez la fenêtre Create key Pair pour la créer.

Pour plus d'informations sur la création d'une nouvelle paire de clés, consultez la section Créer une paire de clés dans le guide de EC2 l'utilisateur Amazon.

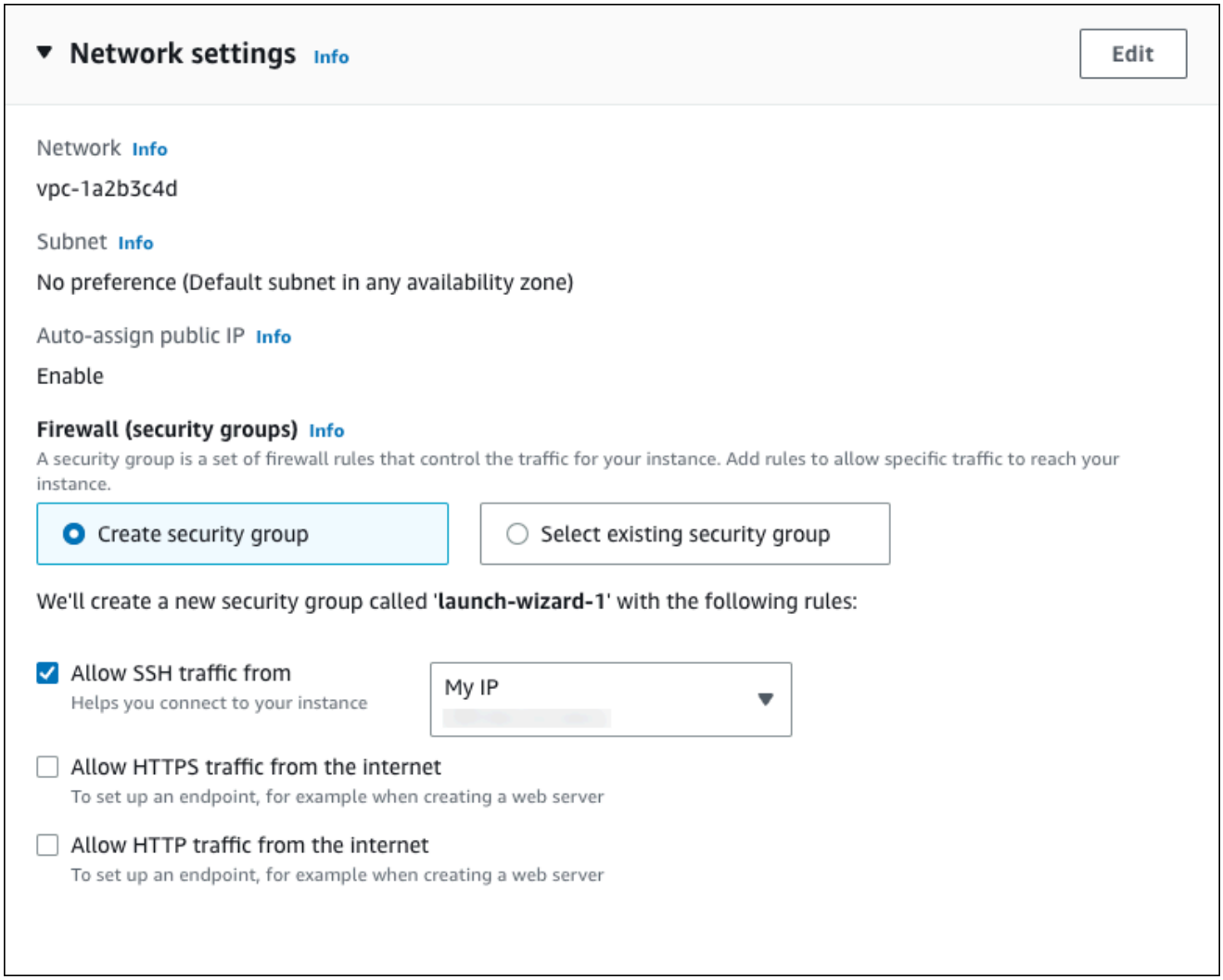
e. Pour Autoriser le trafic SSH dans les paramètres réseau, choisissez la source des connexions SSH à l' EC2instance.

Vous pouvez choisir My IP (Mon IP) si l'adresse IP affichée est correcte pour les connexions SSH. Sinon, vous pouvez déterminer l'adresse IP à utiliser pour vous connecter aux EC2 instances de votre VPC à l'aide de Secure Shell (SSH). Pour déterminer votre adresse IP publique, dans une autre fenêtre ou un autre onglet du navigateur, vous pouvez utiliser le service à l'adresse https://checkip.amazonaws.com. Exemple d'adresse IP : 192.0.2.1/32.

Dans de nombreux cas, votre connexion s'effectue via un fournisseur de services Internet (FSI) ou derrière votre pare-feu sans adresse IP statique. Si tel est le cas, assurez-vous de déterminer la plage d'adresses IP utilisées par les ordinateurs clients.

> **Warning**
>
> Si vous utilisez `0.0.0.0/0` l'accès SSH, vous permettez à toutes les adresses IP d'accéder à vos EC2 instances publiques via SSH. Cette approche est acceptable pour une brève durée dans un environnement de test, mais n'est pas sécurisée pour les environnements de production. En production, autorisez uniquement une adresse IP ou une plage d'adresses spécifiques à accéder à vos EC2 instances via SSH.

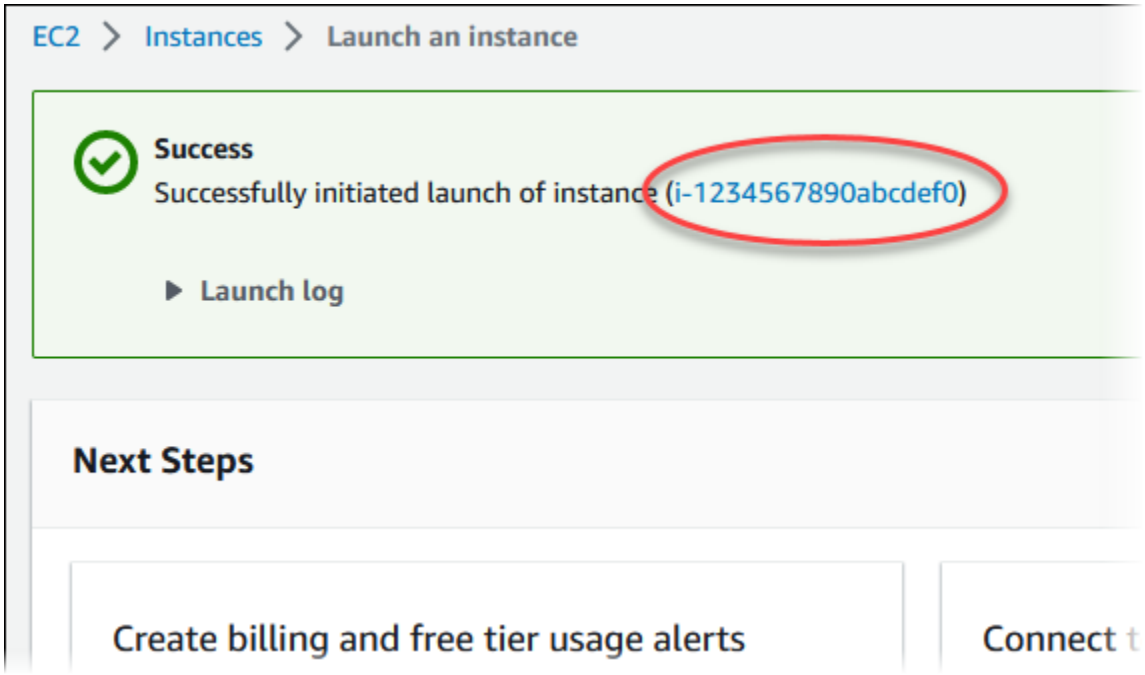
L'image suivante présente un exemple de la section Paramètres réseau.



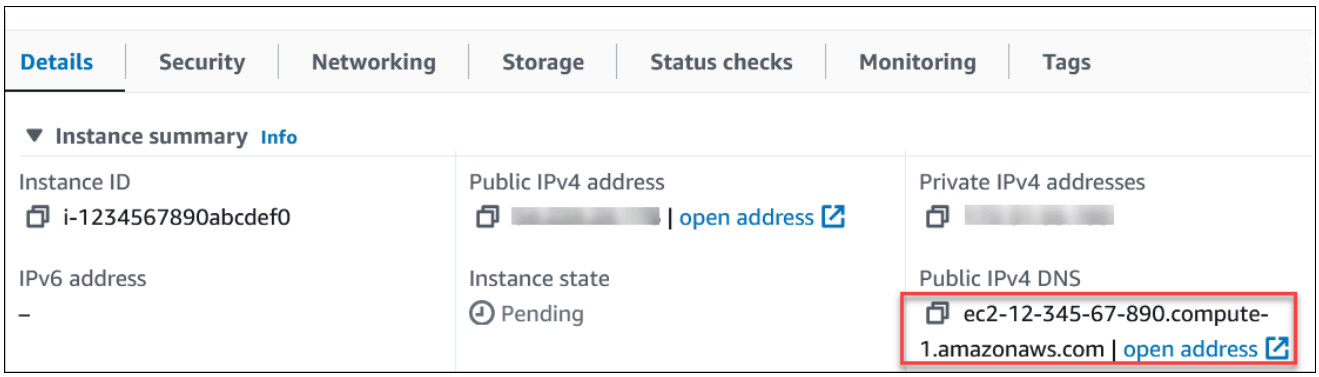
f. Laissez les valeurs par défaut pour les autres sections.

g. Consultez un résumé de la configuration de votre EC2 instance dans le panneau Résumé et, lorsque vous êtes prêt, choisissez Launch instance.

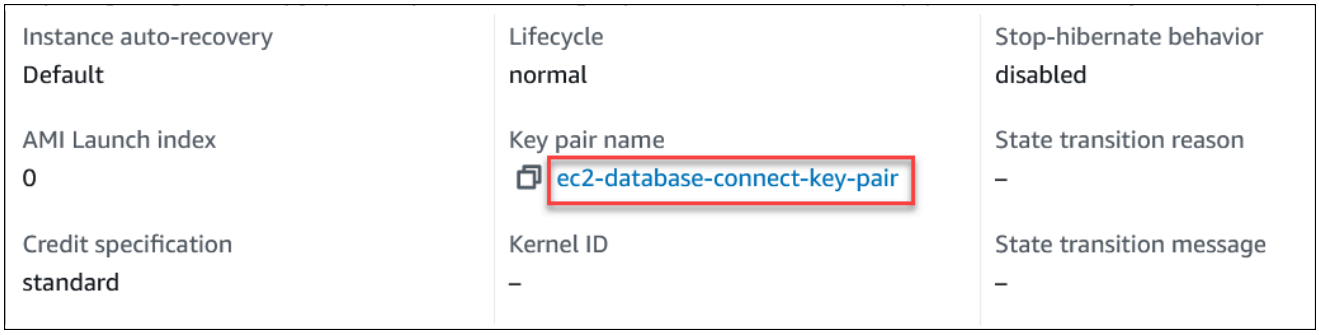
5. Sur la page État du lancement, notez l'identifiant de votre nouvelle EC2 instance, par exemple :`i-1234567890abcdef0`.



6. Choisissez l'identifiant d' EC2 instance pour ouvrir la liste des EC2 instances, puis sélectionnez votre EC2 instance.
7. Dans l’onglet Détails, notez les valeurs suivantes. Vous en aurez besoin lorsque vous vous connecterez via SSH :
   a. Dans Résumé de l'instance, notez la valeur du IPv4 DNS public.



   b. Dans Détails de l’instance, notez la valeur pour Nom de la paire de clés.



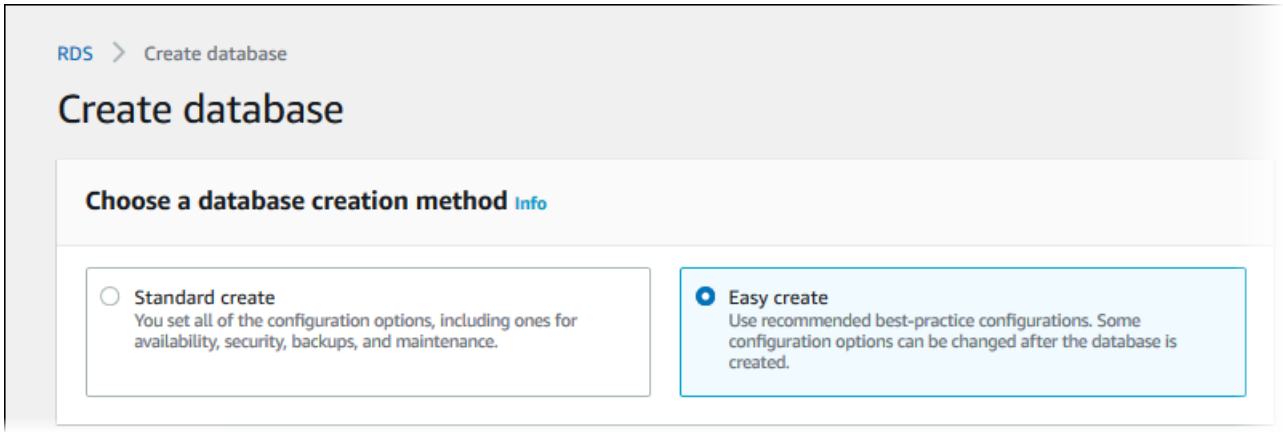
8. Attendez que l'état Instance de votre EC2 instance soit défini sur En cours d'exécution avant de continuer.

Étape 2 : Créer un cluster de bases de données Aurora PostgreSQL

Dans cet exemple, vous utilisez l'option Création facile pour créer un cluster de bases de données Aurora PostgreSQL avec une classe d'instance de base de données db.t4g.large.

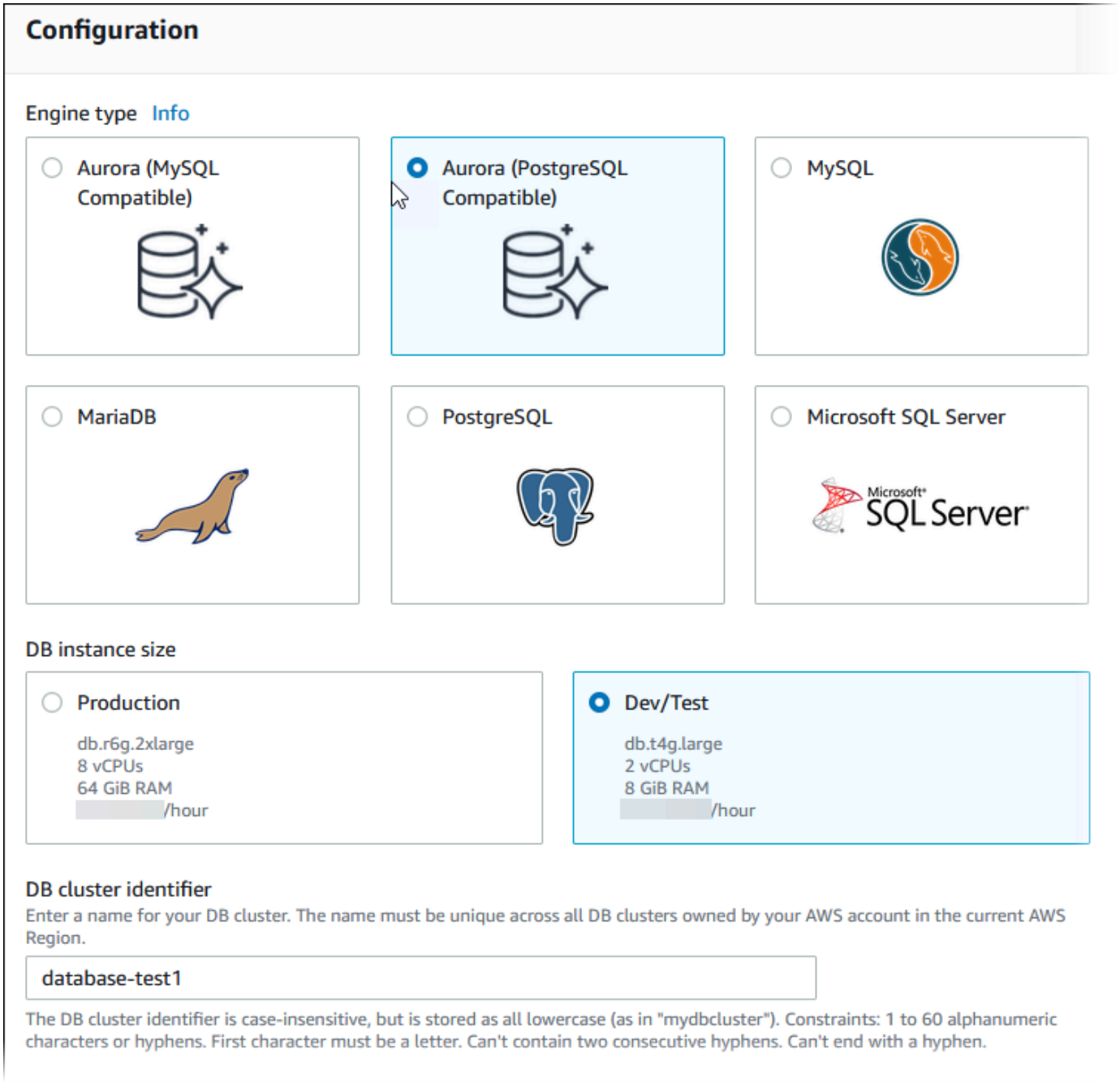
Pour créer un cluster de bases de données Aurora PostgreSQL avec l'option Création facile

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le coin supérieur droit de la console Amazon RDS, choisissez la Région AWS dans laquelle vous voulez créer le cluster de bases de données.
3. Dans le panneau de navigation, choisissez Databases (Bases de données).
4. Choisissez Créer une base de données et veillez à choisir Création facile.



5. Dans Configuration, choisissez Aurora (compatible avec PostgreSQL) pour Type de moteur.
6. Pour DB instance size (Taille de l'instance de base de données), choisissez Dev/Test.
7. Pour Identifiant du cluster de bases de données, saisissez **`database-test1`**.

La page Create database (Créer une base de données) doit ressembler à l'image suivante.

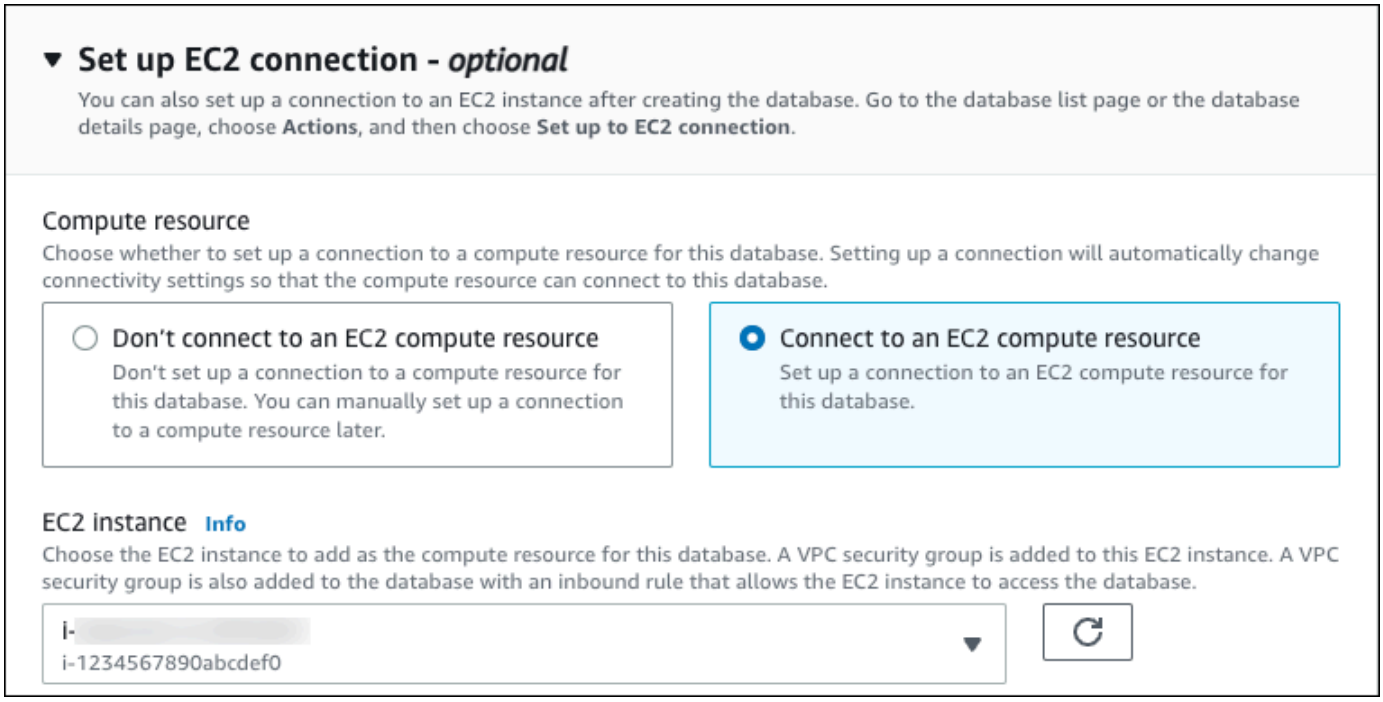


8. Pour Identifiant principal, saisissez un nom pour l'utilisateur ou conservez le nom par défaut (**`postgres`**).
9. Pour utiliser un mot de passe principal généré automatiquement pour le cluster de bases de données, sélectionnez Générer automatiquement un mot de passe.

Pour entrer votre mot de passe principal, veillez à ce que la case Générer automatiquement un mot de passe soit décochée, puis saisissez le même mot de passe dans Mot de passe principal et Confirmer le mot de passe.

10. Pour configurer une connexion avec l' EC2 instance que vous avez créée précédemment, ouvrez Configurer la EC2 connexion - facultatif.

    Sélectionnez Se connecter à une ressource EC2 de calcul. Choisissez l'EC2 instance que vous avez créée précédemment.



11. Ouvrez Afficher les paramètres par défaut pour Création facile.

View default settings for Easy create

Easy create sets the following configurations to their default values, some of which can be changed later. If you want to change any of these settings now, use **Standard create**.

Configuration	Value	Editable after database is created
Encryption	Enabled	No
VPC	Default VPC (vpc-1a2b3c4d)	No
Option group	default:aurora-postgresql-13	No
Subnet group	create-subnet-group	Yes
Automatic backups	Enabled	Yes
VPC security group	sg-1234567	Yes
Publicly accessible	No	Yes
Database port	5432	Yes
DB cluster identifier	database-test1	Yes
DB instance identifier	database-1	Yes
DB engine version	13.6	Yes
DB parameter group	default.aurora-postgresql13	Yes
DB cluster parameter group	default.aurora-postgresql13	Yes
Performance insights	Enabled	Yes
Monitoring	Enabled	Yes
Maintenance	Auto minor version upgrade enabled	Yes
Delete protection	Not enabled	Yes

Vous pouvez examiner les paramètres par défaut utilisés quand l'option Easy create (Création facile) est activée. La colonne Modifiable après la création de la base de données indique les options que vous pouvez modifier après avoir créé la base de données.

- Si un paramètre contient Non dans cette colonne et que vous souhaitez un paramètre différent, vous pouvez utiliser Création standard pour créer le cluster de bases de données.
- Si un paramètre contient Oui dans cette colonne et que vous souhaitez un paramètre différent, vous pouvez utiliser Création standard pour créer le cluster de bases de données ou vous pouvez modifier le cluster de bases de données après l'avoir créé pour modifier le paramètre.

12. Choisissez Créer une base de données.

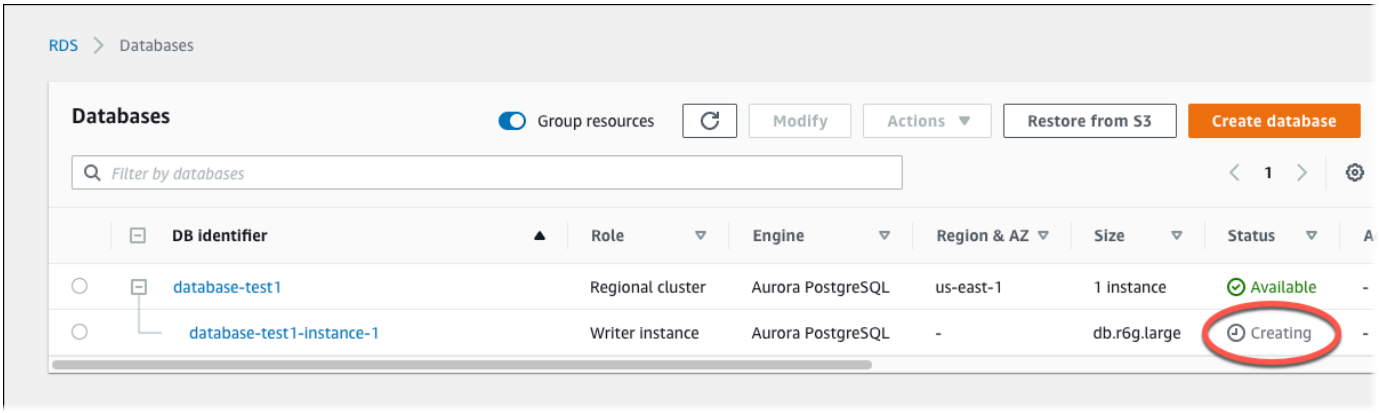
    Pour afficher le nom d'utilisateur principal et le mot de passe pour le cluster de bases de données, choisissez Afficher les détails des informations d'identification.

    Vous pouvez utiliser le nom d'utilisateur et le mot de passe affichés pour vous connecter au cluster de bases de données en tant qu'utilisateur principal.

    > **Important**
    >
    > Vous ne pourrez pas afficher le mot de passe de l'utilisateur principal de nouveau. Si vous ne l'enregistrez pas, il sera peut-être nécessaire de le modifier.
    > Si vous devez changer le mot de passe de l'utilisateur principal une fois le cluster de bases de données disponible, vous pouvez le faire en modifiant le cluster de bases de données. Pour plus d'informations sur la modification d'un cluster, consultez Modification d'un cluster de bases de données Amazon Aurora.

13. Dans la liste Bases de données, choisissez le nom du nouveau cluster de bases de données Aurora PostgreSQL pour afficher ses détails.

    L'instance d'enregistreur a le statut Création en cours jusqu'à ce que le cluster de bases de données soit prêt à l'emploi.



Lorsque le statut de l'instance d'enregistreur passe à Disponible, vous pouvez vous connecter au cluster de bases de données. En fonction de la quantité de stockage et de la classe d'instance de base de données, la mise à disposition du nouveau cluster de bases de données peut prendre jusqu'à 20 minutes.

(Facultatif) Créez un VPC, une EC2 instance et un cluster Aurora PostgreSQL à l'aide de CloudFormation

Au lieu d'utiliser la console pour créer votre VPC, votre EC2 instance et votre cluster de base de données Aurora PostgreSQL, vous pouvez les utiliser CloudFormation pour provisionner des AWS ressources en traitant l'infrastructure comme du code. Pour vous aider à organiser vos AWS ressources en unités plus petites et plus faciles à gérer, vous pouvez utiliser la fonctionnalité de pile CloudFormation imbriquée. Pour plus d'informations, consultez les sections Création d'une pile sur la CloudFormation console et Utilisation de piles imbriquées.

> **Important**
>
> CloudFormation est gratuit, mais les ressources qui en CloudFormation découlent sont vivantes. Les frais d'utilisation standard pour ces ressources vous sont facturés jusqu'à ce que vous les résiliez. Pour plus d'informations, consultez Tarification Amazon Aurora.

Pour créer vos ressources à l'aide de la CloudFormation console, procédez comme suit :

- Étape 1 : Téléchargez le CloudFormation modèle
- Étape 2 : configurez vos ressources à l'aide de CloudFormation

Téléchargez le CloudFormation modèle

Un CloudFormation modèle est un fichier texte JSON ou YAML qui contient les informations de configuration relatives aux ressources que vous souhaitez créer dans la pile. Ce modèle crée également un VPC et un hôte bastion pour vous, en plus du cluster Aurora.

Pour télécharger le fichier modèle, ouvrez le lien suivant, Modèle Aurora CloudFormation PostgreSQL.

Sur la page Github, cliquez sur le bouton Télécharger le fichier brut pour enregistrer le modèle de fichier YAML.

Configurez vos ressources à l'aide de CloudFormation

> **Note**
>
> Avant de commencer ce processus, assurez-vous que vous disposez d'une paire de clés pour une EC2 instance dans votre Compte AWS. Pour plus d'informations, consultez les paires de EC2 clés Amazon et les instances Linux.

Lorsque vous utilisez le CloudFormation modèle, vous devez sélectionner les paramètres appropriés pour vous assurer que vos ressources sont créées correctement. Procédez de la façon suivante :

1. Connectez-vous à la CloudFormation console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/cloudformation.
2. Sélectionnez Créer une pile.
3. Dans la section Spécifier le modèle, sélectionnez Charger un fichier de modèle à partir de votre ordinateur, puis cliquez sur Suivant.
4. Sur la page Spécifier les détails de la pile, saisissez les paramètres suivants :
   a. Définissez le nom de la AurPostgreSQLTestpile sur Stack.
   b. Sous Paramètres, définissez les zones de disponibilité en sélectionnant deux zones de disponibilité.
   c. Dans Configuration de l'hôte Linux Bastion, pour Nom de la clé, sélectionnez une paire de clés pour vous connecter à votre EC2 instance.
   d. Dans les paramètres de Configuration de l’hôte bastion Linux, définissez Plage d’adresses IP autorisées sur votre adresse IP. Pour vous connecter aux EC2 instances de votre VPC à l'aide

de Secure Shell (SSH), déterminez votre adresse IP publique à l'aide du service à l'adresse. https://checkip.amazonaws.com Exemple d'adresse IP : 192.0.2.1/32.

> **Warning**
>
> Si vous utilisez `0.0.0.0/0` l'accès SSH, vous permettez à toutes les adresses IP d'accéder à vos EC2 instances publiques via SSH. Cette approche est acceptable pour une brève durée dans un environnement de test, mais n'est pas sécurisée pour les environnements de production. En production, autorisez uniquement une adresse IP ou une plage d'adresses spécifiques à accéder à vos EC2 instances via SSH.

e. Dans Configuration générale de la base de données, définissez la classe d'instance de base de données sur db.t4g.large.

f. Définissez Nom de la base de données sur **`database-test1`**.

g. Pour Nom d'utilisateur principal de la base de données, entrez le nom de l'utilisateur principal.

h. Définissez Gérer le mot de passe utilisateur principal de base de données avec Secrets Manager sur `false` pour ce didacticiel.

i. Pour Mot de passe de la base de données, définissez le mot de passe de votre choix. N'oubliez pas ce mot de passe pour pouvoir effectuer les étapes suivantes du didacticiel.

j. Définissez Déploiement multi-AZ sur `false`.

k. Conservez les valeurs par défaut de tous les autres paramètres. Cliquez sur Suivant pour continuer.

5. Sur la page Configurer les options de pile, conservez toutes les options par défaut. Cliquez sur Suivant pour continuer.

6. Sur la page Vérification de la pile, sélectionnez Soumettre après avoir vérifié les options de la base de données et de l'hôte bastion Linux.

Une fois le processus de création des piles terminé, visualisez les piles avec leurs noms BastionStacket leurs APGNS pour noter les informations dont vous avez besoin pour vous connecter à la base de données. Pour plus d'informations, consultez la section Affichage des données et des ressources de la CloudFormation pile sur le AWS Management Console.

Étape 3 : Se connecter à un cluster de bases de données Aurora PostgreSQL

Vous pouvez utiliser n'importe quelle application client PostgreSQL standard pour vous connecter au cluster de bases de données. Dans cet exemple, vous vous connectez à un cluster de bases de données Aurora PostgreSQL en utilisant le client de ligne de commande psql.

Pour se connecter à un cluster de bases de données Aurora PostgreSQL

1. Trouvez le point de terminaison (nom DNS) et le numéro de port de l'instance d'enregistreur pour votre cluster de bases de données.

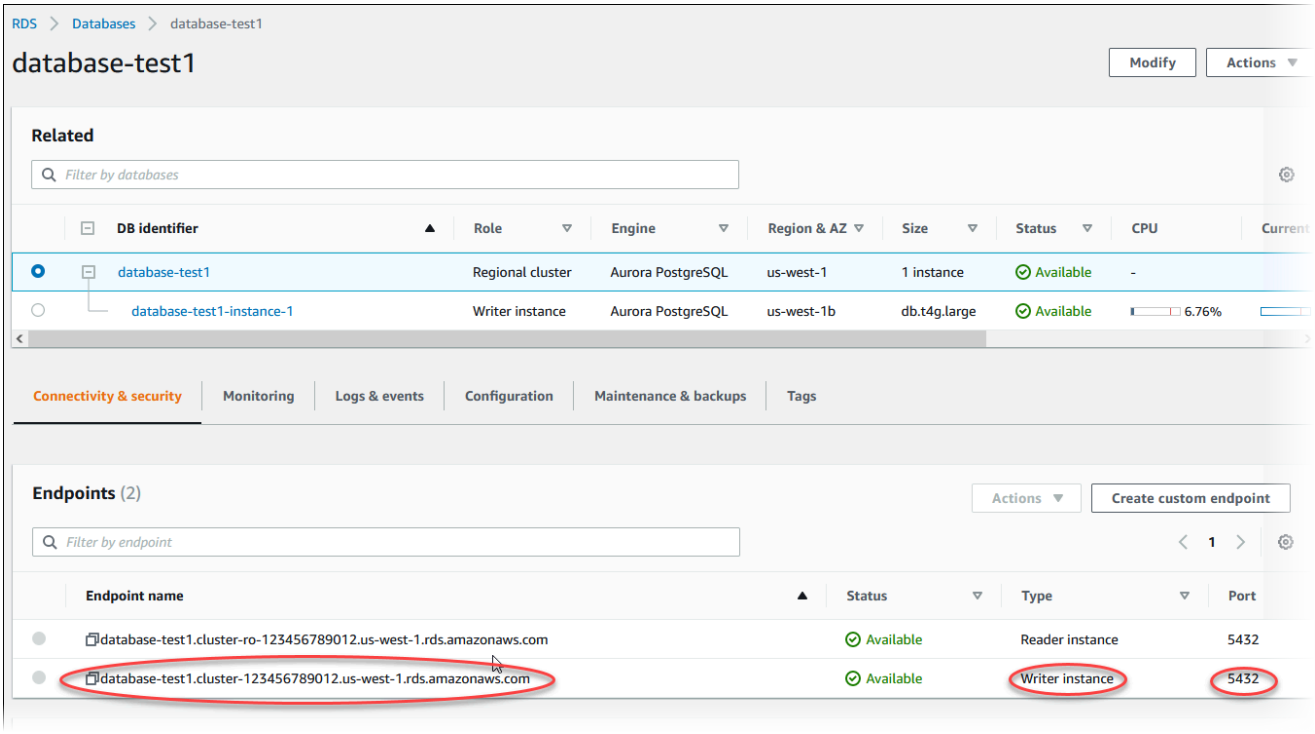
   a. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.

   b. Dans le coin supérieur droit de la console Amazon RDS, choisissez le cluster de base Région AWS de données.

   c. Dans le panneau de navigation, choisissez Databases (Bases de données).

   d. Choisissez le nom du cluster de bases de données Aurora PostgreSQL pour afficher ses détails.

   e. Dans l'onglet Connectivité et sécurité, copiez le point de terminaison de l'instance d'enregistreur. Notez également le numéro du port. Vous avez besoin du point de terminaison et du numéro de port pour vous connecter au cluster de bases de données.



2. Connectez-vous à l' EC2 instance que vous avez créée précédemment en suivant les étapes décrites dans la section Connexion à votre instance Linux dans le guide de EC2 l'utilisateur Amazon.

   Nous vous recommandons de vous connecter à votre EC2 instance via SSH. Si l'utilitaire client SSH est installé sur Windows, Linux ou Mac, vous pouvez vous connecter à l'instance à l'aide du format de commande suivant :

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   Supposons, par exemple, qu'`ec2-database-connect-key-pair.pem`il soit stocké `/dir1` sous Linux et que le IPv4 DNS public de votre EC2 instance l'est`ec2-12-345-678-90.compute-1.amazonaws.com`. Votre commande SSH se présenterait comme suit :

   ```
   ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
   user@ec2-12-345-678-90.compute-1.amazonaws.com
   ```

3. Obtenez les dernières corrections de bogues et mises à jour de sécurité en mettant à jour le logiciel de votre EC2 instance. Pour cela, utilisez la commande suivante.

> **Note**
>
> L'option `-y` installe les mises à jour sans demander de confirmation. Pour examiner les mises à jour avant de les installer, omettez cette option.

```
sudo dnf update -y
```

4. Pour installer le client de ligne de commande mysql depuis PostgreSQL sur Amazon Linux 2023, exécutez la commande suivante :

```
sudo dnf install postgresql15
```

5. Connectez-vous à un cluster de bases de données Aurora PostgreSQL. Par exemple, saisissez la commande suivante. Cette action vous permet de vous connecter au cluster de bases de données Aurora PostgreSQL à l'aide du client psql.

   Remplacez le point de terminaison de l'instance d'enregistreur pour *`endpoint`*, remplacez le nom de la base de données `--dbname` à laquelle vous voulez vous connecter pour *`postgres`* et remplacez le nom d'utilisateur principal que vous avez utilisé pour *`postgres`*. Indiquez le mot de passe principal que vous avez utilisé lorsque vous êtes invité à entrer un mot de passe.

```
psql --host=endpoint --port=5432 --dbname=postgres --username=postgres
```

   Après avoir entré le mot de passe pour l'utilisateur, le résultat suivant devrait normalement s'afficher.

```
psql (14.3, server 14.6)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

   Pour plus d'informations sur la connexion à un cluster de bases de données Aurora PostgreSQL, consultez Connexion à un cluster de bases de données Amazon Aurora PostgreSQL. Si vous ne pouvez pas vous connecter à votre cluster de bases de données, consultez Impossible de se connecter à l'instance de base de données Amazon RDS.

Pour des raisons de sécurité, une bonne pratique consiste à recommander d'utiliser des connexions chiffrées. N'utilisez une connexion PostgreSQL non chiffrée que quand le client et le serveur sont dans le même VPC et que le réseau est approuvé. Pour plus d'informations sur l'utilisation de connexions chiffrées, consultez Sécurisation des données Aurora PostgreSQL avec SSL/TLS.

6. Exécutez des commandes SQL.

   Par exemple, la commande SQL suivante indique la date et l'heure actuelles :

   ```
   SELECT CURRENT_TIMESTAMP;
   ```

Étape 4 : Supprimer l' EC2instance et le cluster de base de données

Une fois que vous vous êtes connecté et que vous avez exploré l'exemple d' EC2 instance et de cluster de base de données que vous avez créés, supprimez-les afin qu'ils ne vous soient plus facturés.

Si vous aviez CloudFormation l'habitude de créer des ressources, ignorez cette étape et passez à l'étape suivante.

Pour supprimer l' EC2 instance

1. Connectez-vous à la EC2 console Amazon AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/ec2/.
2. Dans le panneau de navigation, choisissez Instances.
3. Sélectionnez l' EC2 instance, puis choisissez État de l'instance, Terminer l'instance.
4. Choisissez Résilier lorsque vous êtes invité à confirmer l'action.

Pour plus d'informations sur la suppression d'une EC2 instance, consultez Résilier votre instance dans le guide de EC2 l'utilisateur Amazon.

Pour supprimer un cluster DB

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.

2. Choisissez Bases de données et sélectionnez ensuite l’instance de base de données associée au cluster de bases de données.
3. Pour Actions, choisissez Supprimer.
4. Sélectionnez Delete.

   Une fois que vous avez supprimé toutes les instances de base de données associées à un cluster de bases de données, ce dernier est automatiquement supprimé.

(Facultatif) Supprimez l' EC2 instance et le cluster de base de données créés avec CloudFormation

Si vous aviez l'habitude de CloudFormation créer des ressources, supprimez la CloudFormation pile après vous être connecté et exploré l'exemple d' EC2 instance et de cluster de base de données, afin de ne plus être facturé pour ces éléments.

Pour supprimer les CloudFormation ressources

1. Ouvrez la CloudFormation console.
2. Sur la page Stacks de la CloudFormation console, sélectionnez la pile racine (la pile sans nom VPCStack, BastionStack ou APGNS).
3. Sélectionnez Delete (Supprimer).
4. Sélectionnez Supprimer la pile lorsque vous êtes invité à confirmer l’action.

Pour plus d'informations sur la suppression d'une pile dans CloudFormation, consultez la section Supprimer une pile sur la CloudFormation console dans le Guide de AWS CloudFormation l'utilisateur.

(Facultatif) Connecter votre cluster de bases de données à une fonction Lambda

Vous pouvez également connecter votre cluster de bases de données Aurora PostgreSQL à une ressource de calcul sans serveur Lambda. Les fonctions Lambda vous permettent d’exécuter du code sans provisionner ni gérer l’infrastructure. Une fonction Lambda vous permet également de répondre automatiquement aux demandes d’exécution de code à n’importe quelle échelle, d’une douzaine d’événements par jour à des centaines par seconde. Pour plus d’informations, consultez Connexion automatique d’une fonction Lambda et d’un cluster de bases de données Aurora.

Didacticiel : Créer un serveur web et une cluster de base de données Amazon Aurora

Ce didacticiel vous montre comment installer un serveur web Apache avec PHP et créer une base de données MariaDB, MySQL ou PostgreSQL. Le serveur web s'exécute sur une instance Amazon EC2 utilisant Amazon Linux 2023 et vous pouvez choisir entre un cluster de bases de données Aurora MySQL ou Aurora PostgreSQL. L'instance Amazon EC2 et l'cluster de base de données s'exécutent tous deux dans un Virtual Private Cloud (VPC) basé sur le service Amazon VPC.

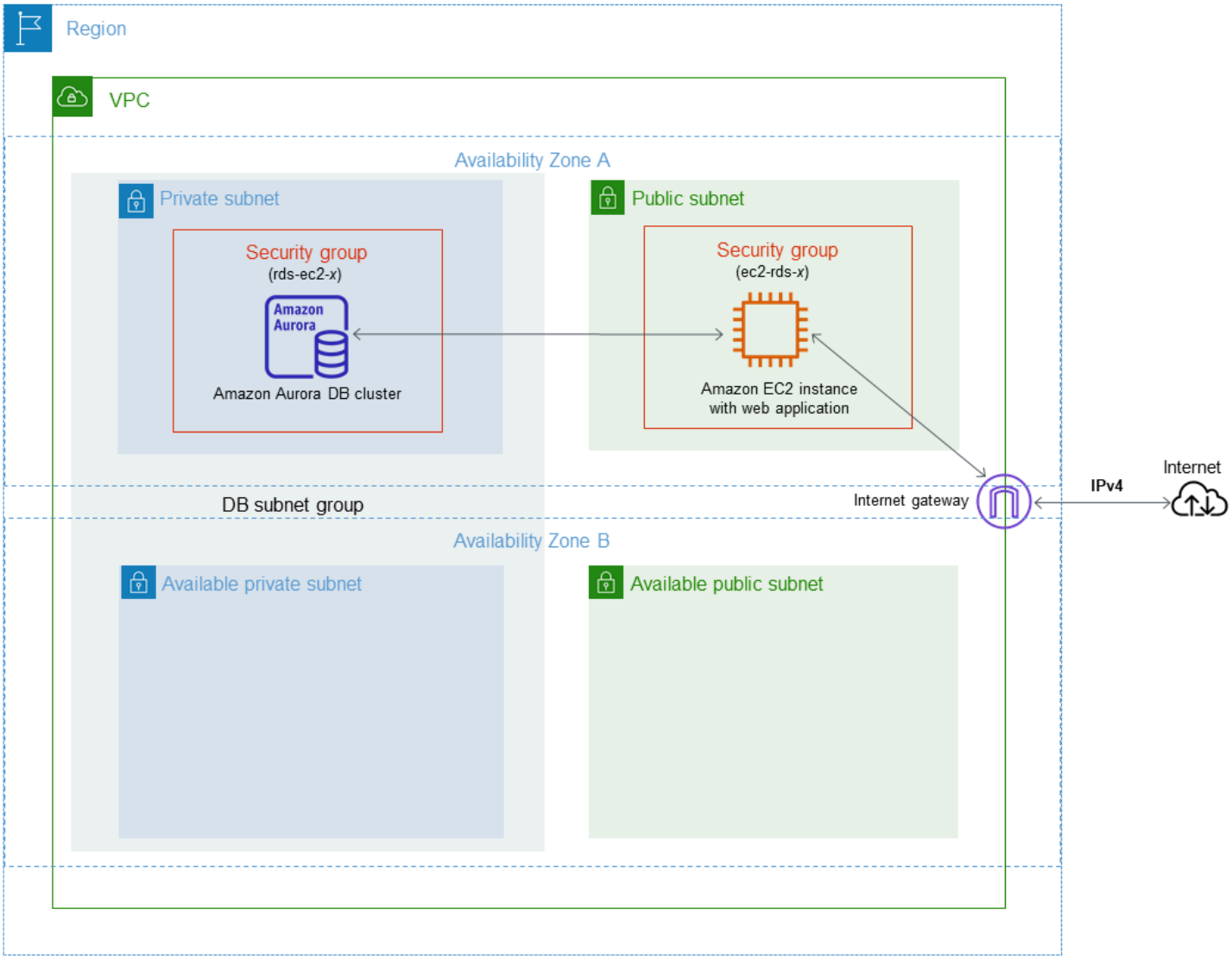
> **Important**
>
> Il n'y a pas de frais pour la création d'un compte AWS. Toutefois, au cours de ce didacticiel, des coûts peuvent être générés par l'utilisation des ressources AWS. Vous pouvez supprimer ces ressources après avoir terminé le didacticiel si elles ne sont plus nécessaires.

> **Note**
>
> Ce didacticiel s'applique à Amazon Linux 2023 et peut ne pas fonctionner pour d'autres versions de Linux.

Dans le tutoriel qui suit, vous créez une instance EC2 qui utilise le VPC, les sous-réseaux et le groupe de sécurité par défaut pour votre Compte AWS. Ce tutoriel vous montre comment créer le cluster de base de données et configurer automatiquement la connectivité avec l'instance EC2 que vous avez créée. Le tutoriel vous montre ensuite comment installer le serveur web sur l'instance EC2. Vous connectez votre serveur Web à votre cluster de base de données dans le VPC en utilisant le point de terminaison du cluster en écriture de la base de données.

1. Lancement d’une instance EC2 pour vous connecter à votre cluster de bases de données
2. Créer un cluster de bases de données Amazon Aurora
3. Installer un serveur web sur votre instance EC2

Le diagramme suivant affiche la configuration obtenue au terme de ce didacticiel.



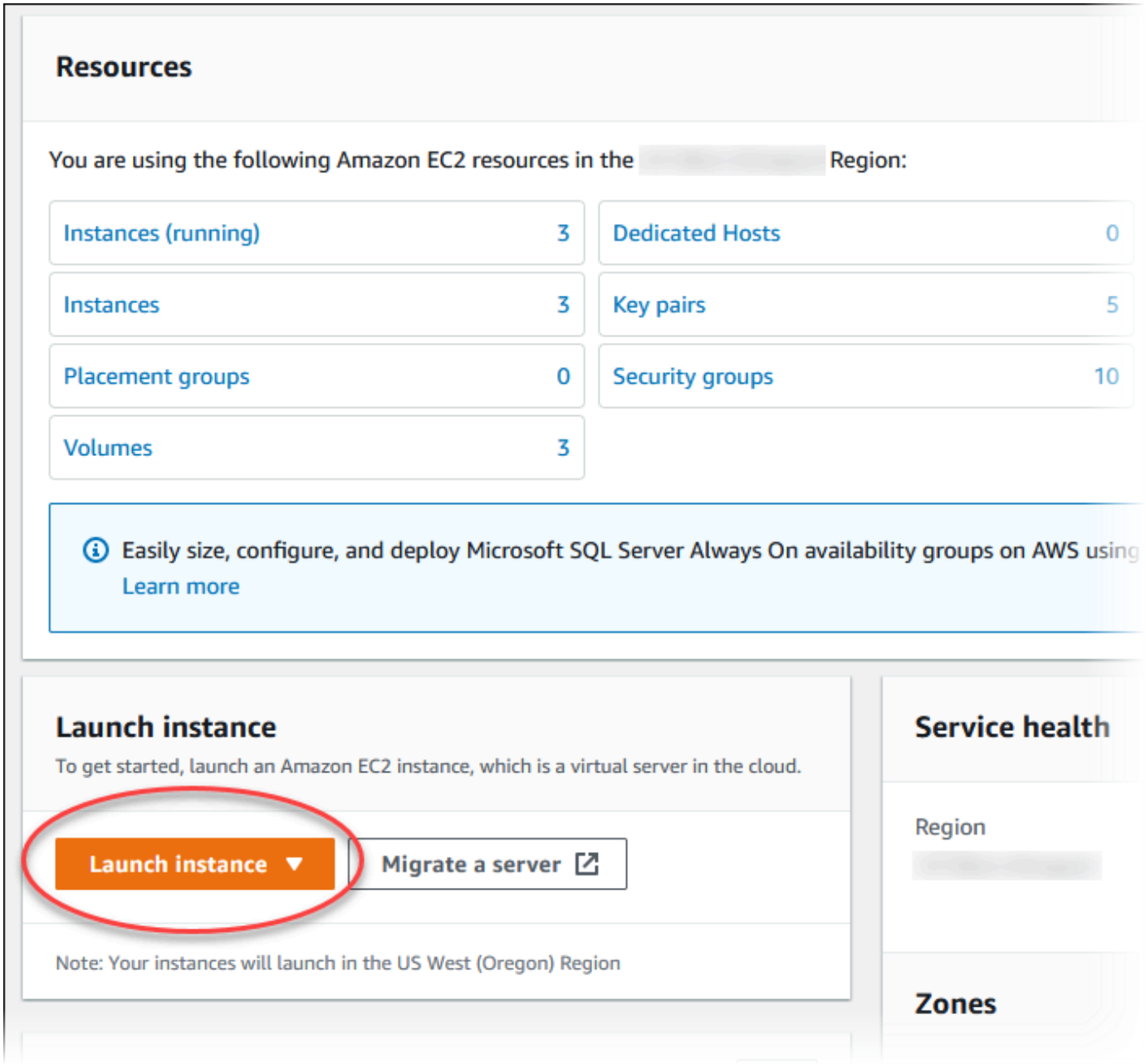
> **Note**
>
> Une fois le tutoriel terminé, chaque zone de disponibilité de votre VPC comporte un sous-réseau public et un sous-réseau privé. Ce tutoriel utilise le VPC par défaut pour votre Compte AWS et configure automatiquement la connectivité entre votre instance EC2 et le cluster de base de données. Si vous préférez plutôt configurer un nouveau VPC pour ce scénario, suivez les étapes décrites dans Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement).

Lancement d’une instance EC2 pour vous connecter à votre cluster de bases de données

Créez une instance Amazon EC2 dans le sous-réseau public de votre VPC.

Pour lancer une instance EC2

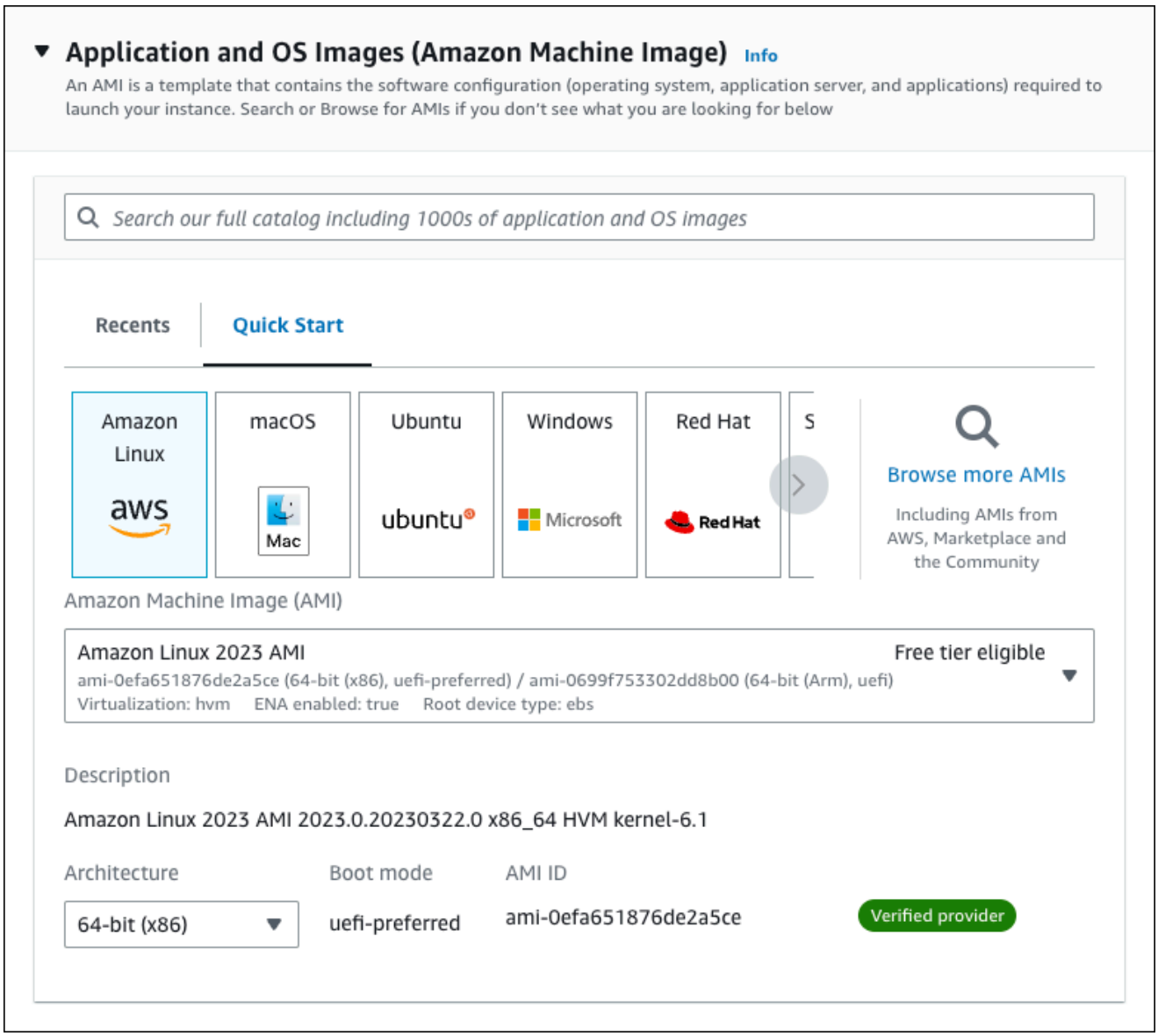
1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon EC2 à l’adresse https://console.aws.amazon.com/ec2/.
2. Dans le coin supérieur droit d’AWS Management Console, choisissez la Région AWS dans laquelle vous voulez créer l’instance EC2.
3. Choisissez Tableau de bord EC2, puis Lancer une instance, comme illustré ci-dessous.



4. Choisissez les paramètres suivants sur la page Lancer une instance.

   a. Sous Name and tags (Nom et identifications), pour Name (Nom), saisissez **`tutorial-ec2-instance-web-server`**.

   b. Sous Application et images OS (Amazon Machine Image), choisissez Amazon Linux, puis Amazon Linux 2023 AMI. Conservez les valeurs par défaut pour les autres choix.



c. Sous Instance type (Type d'instance), choisissez t2.micro.

d. Sous Key pair (login) [Paire de clés (connexion)], choisissez une valeur Key pair name (Nom de paire de clés) pour utiliser une paire de clés existante. Pour créer une paire de clés pour l'instance Amazon EC2, choisissez Create new key pair (Créer une paire de clés), puis utilisez la fenêtre Create key pair (Créer une paire de clés) pour la créer.

   Pour plus d'informations sur la création d'une paire de clés, consultez Création d'une paire de clés dans le Guide de l'utilisateur Amazon EC2.

e. Sous Network settings (Paramètres réseau), définissez ces valeurs et conservez les autres valeurs par défaut :

- Pour Allow SSH traffic from (Autoriser le trafic SSH depuis), choisissez la source des connexions SSH vers l'instance EC2.

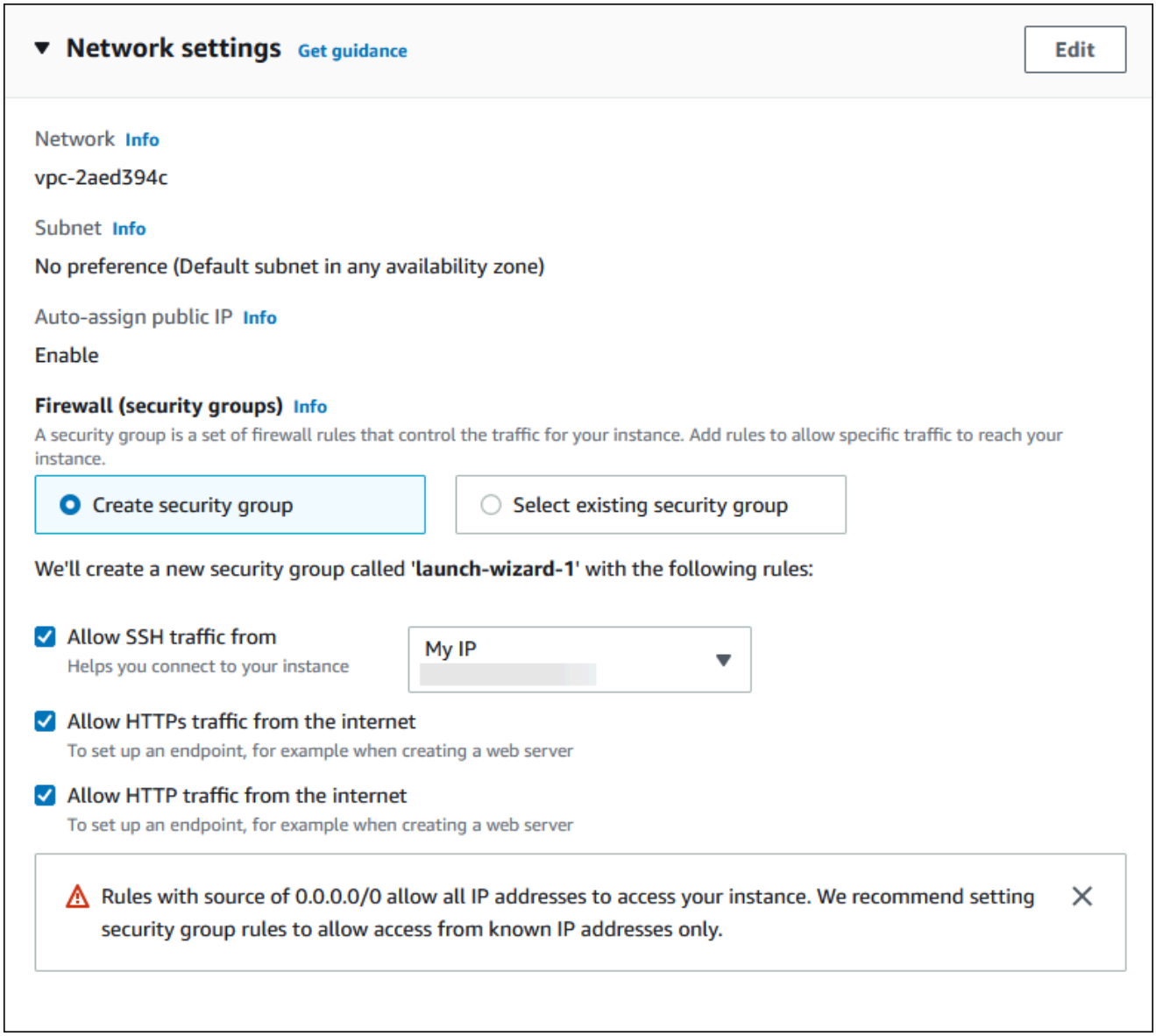
  Vous pouvez choisir My IP (Mon IP) si l'adresse IP affichée est correcte pour les connexions SSH.

  Sinon, vous pouvez déterminer l'adresse IP à utiliser pour vous connecter aux instances EC2 dans votre VPC en utilisant Secure Shell (SSH). Pour déterminer votre adresse IP publique, dans une fenêtre ou un onglet de navigateur différent, vous pouvez utiliser le service à l'adresse https://checkip.amazonaws.com. Exemple d'adresse IP : `203.0.113.25/32`.

  Dans de nombreux cas, votre connexion s'effectue via un fournisseur de services Internet (FSI) ou derrière votre pare-feu sans adresse IP statique. Si tel est le cas, assurez-vous de déterminer la plage d'adresses IP utilisées par les ordinateurs clients.

  > **Warning**
  >
  > Si vous utilisez `0.0.0.0/0` pour l'accès SSH, vous permettez à toutes les adresses IP d'accéder à vos instances publiques par SSH. Cette approche est acceptable pour une brève durée dans un environnement de test, mais n'est pas sécurisée pour les environnements de production. Dans un environnement de production, autorisez uniquement l'accès à vos instances à l'aide de SSH pour une adresse IP ou une plage d'adresses spécifique.

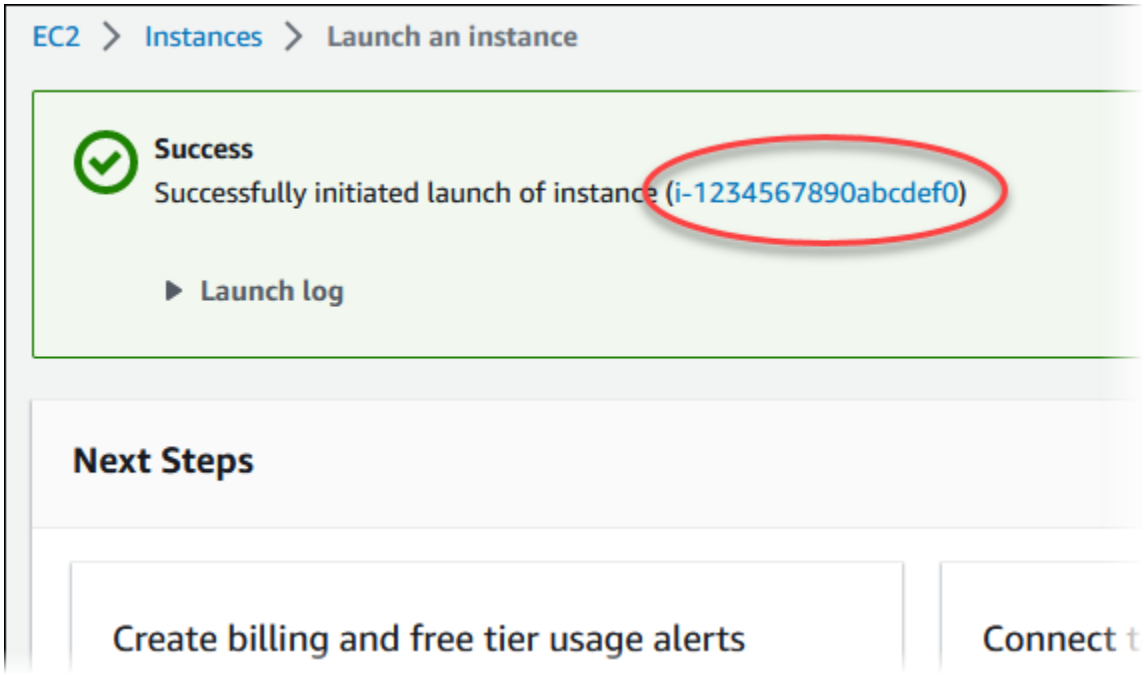
- Activez l'option Allow HTTPs traffic from the internet (Autoriser le trafic HTTPs depuis Internet).
- Activez l'option Allow HTTP traffic from the internet (Autoriser le trafic HTTP depuis Internet).



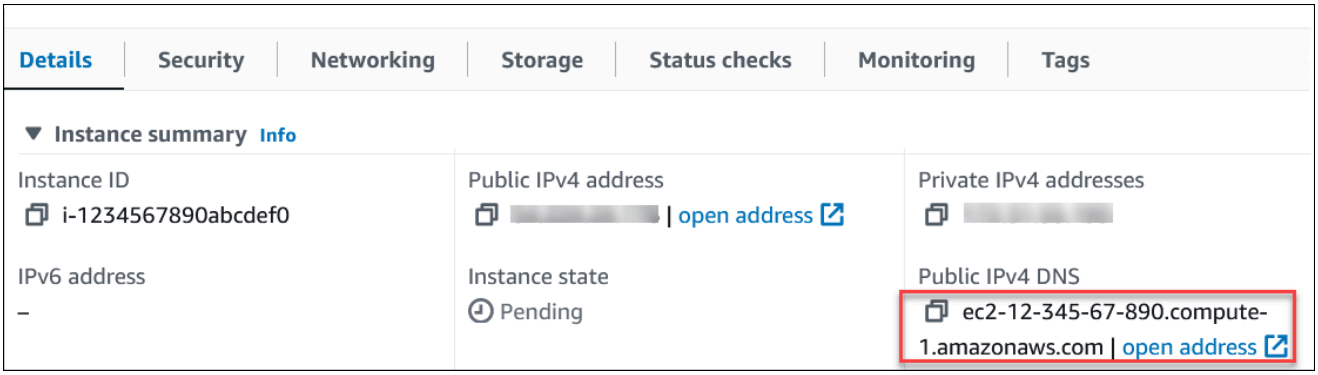
f. Laissez les valeurs par défaut pour les autres sections.

g. Consultez un résumé de la configuration de votre instance dans le panneau Summary (Récapitulatif) et, lorsque vous êtes prêt, choisissez Launch instance (Lancer l'instance).

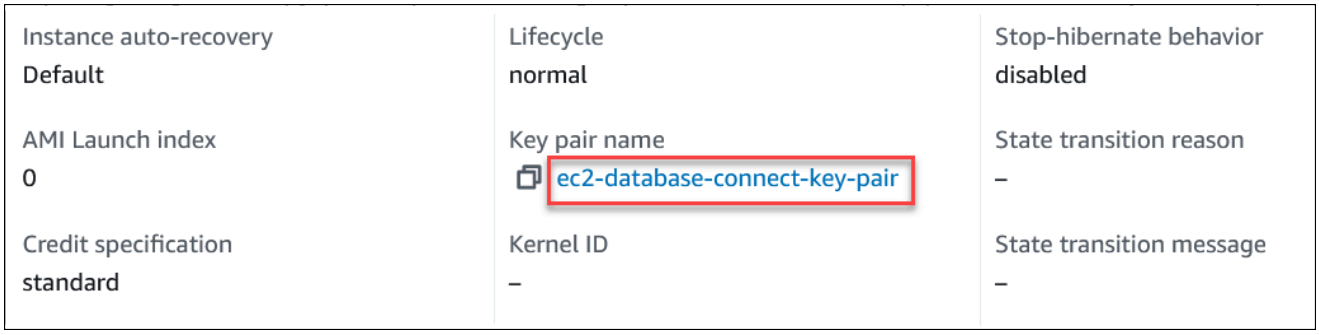
5. Sur la page Statut de lancement, notez l'identifiant de votre nouvelle instance EC2, tel que : `i-1234567890abcdef0`.



6. Choisissez l'identifiant de l'instance EC2 pour ouvrir la liste des instances EC2, puis sélectionnez votre instance EC2.
7. Dans l'onglet Détails, notez les valeurs suivantes. Vous en aurez besoin lorsque vous vous connecterez via SSH :
   a. Dans Résumé de l'instance, notez la valeur pour DNS IPv4 public.



   b. Dans Détails de l'instance, notez la valeur pour Nom de la paire de clés.



8. Attendez que la valeur Instance state (État de votre instance) soit Running (En cours d'exécution) avant de continuer.
9. Termin Créer un cluster de bases de données Amazon Aurora.

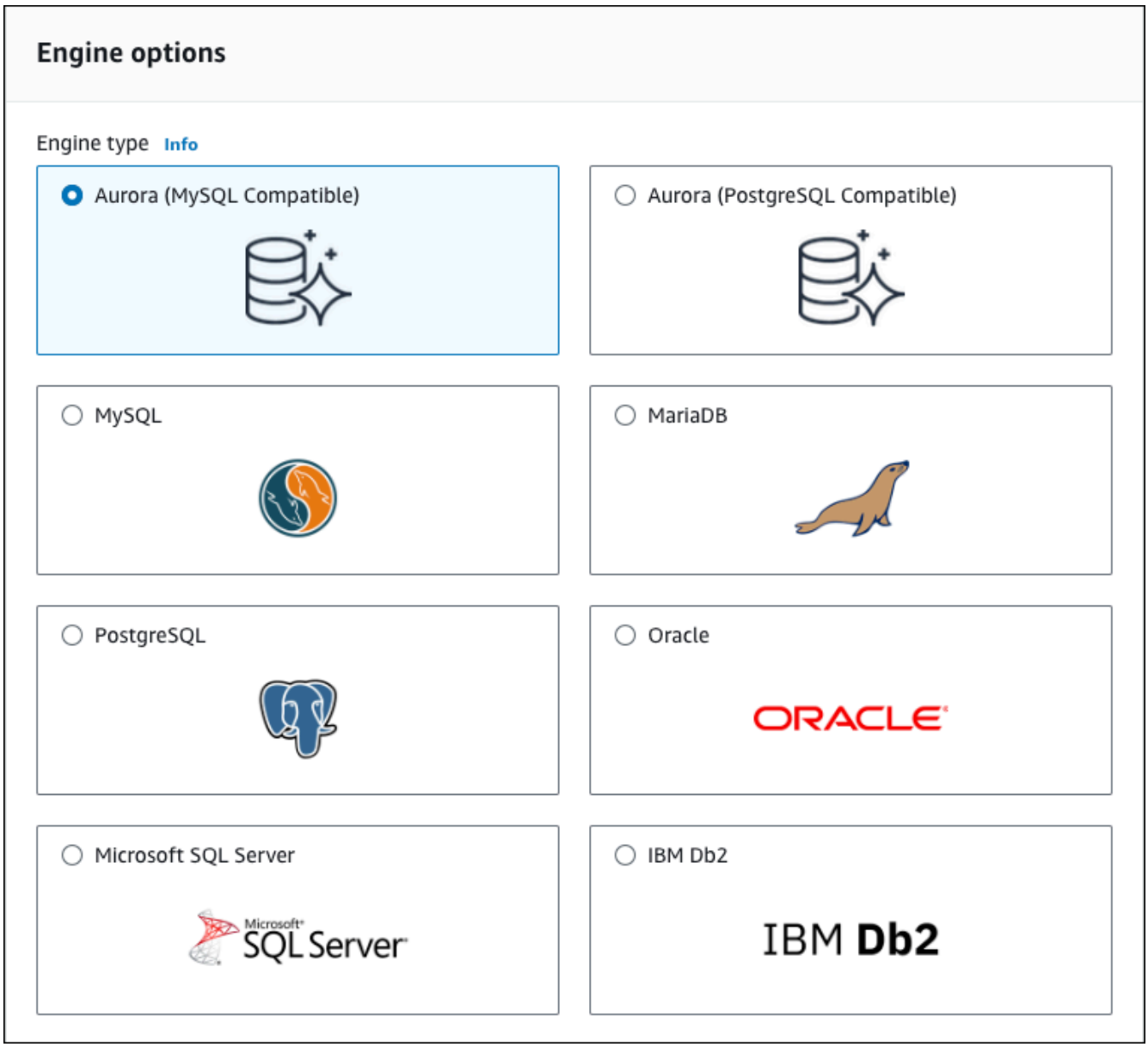
Créer un cluster de bases de données Amazon Aurora

Créez un cluster de bases de données Amazon Aurora MySQL ou Aurora PostgreSQL qui conserve les données utilisées par une application web.

Aurora MySQL

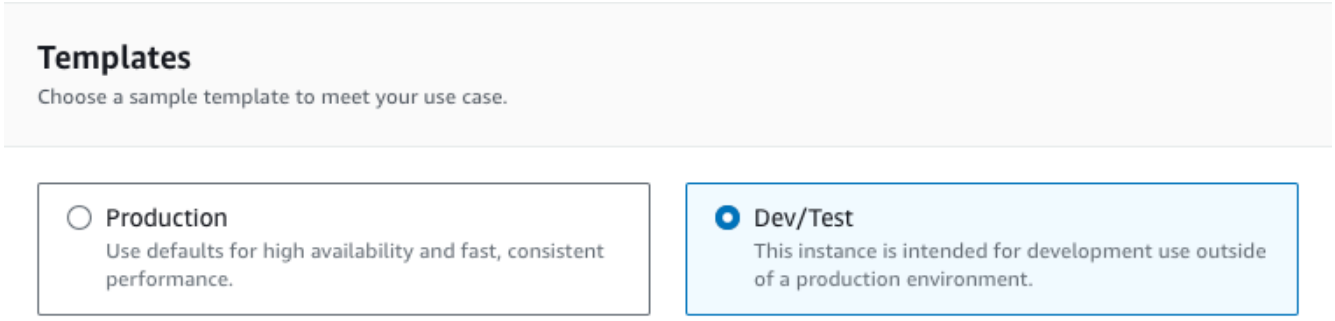
Pour créer un cluster de bases de données Aurora MySQL

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le coin supérieur droit de la AWS Management Console, assurez-vous que la Région AWS est la même que celle où vous avez créé votre instance EC2.
3. Dans la panneau de navigation, choisissez Bases de données.
4. Choisissez Create database (Créer une base de données).
5. Sur la page Créer une base de données, choisissez Création standard.
6. Pour Options de moteur, choisissez Aurora (compatible avec MySQL).



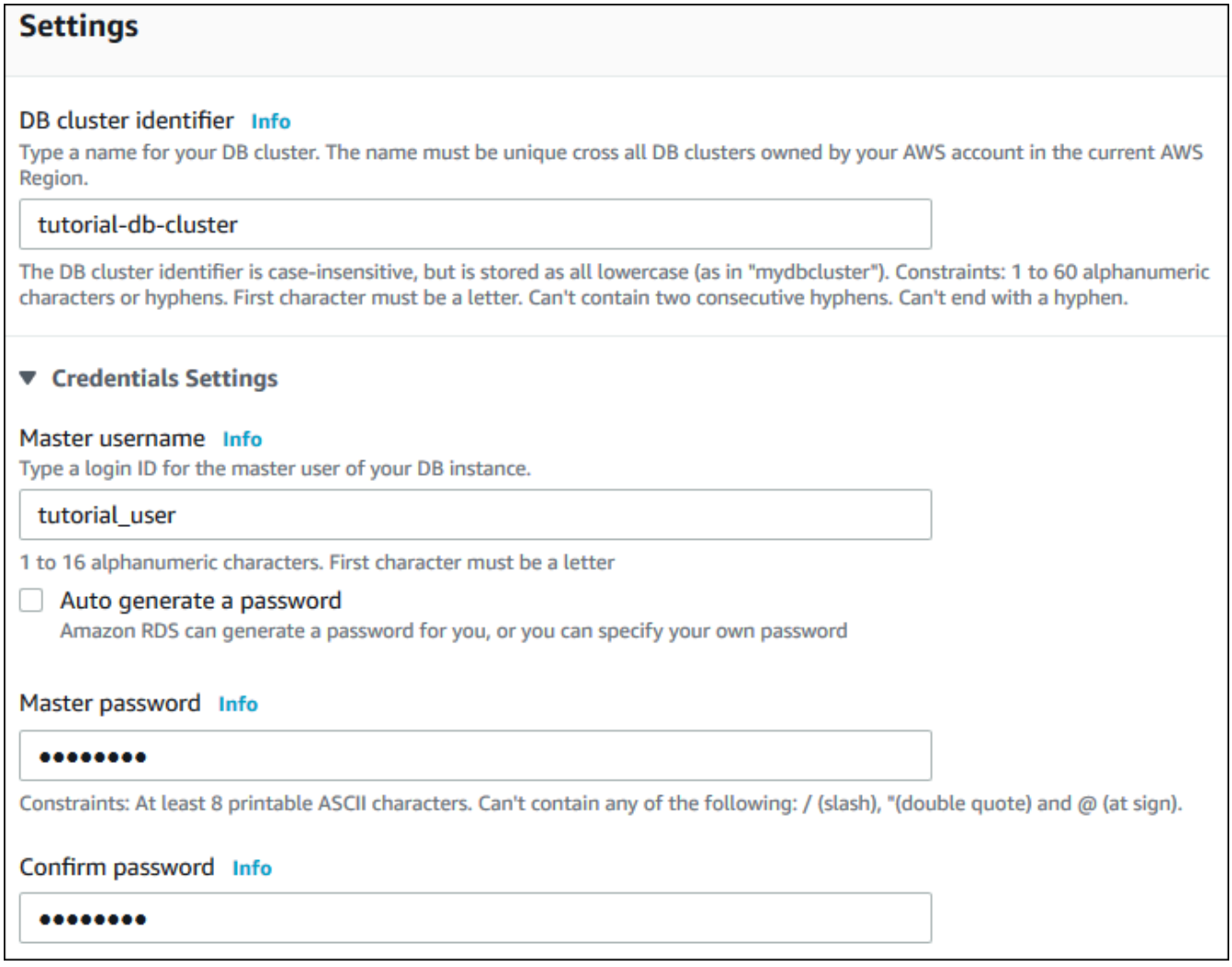
Conservez les valeurs par défaut pour Version et les autres options du moteur.

7. Dans la section Templates (Modèles), choisissez Dev/Test.



8. Dans la section Settings (Paramètres), définissez les valeurs suivantes :

   - Identifiant du cluster de bases de données : saisissez **`tutorial-db-cluster`**.
   - Master username (Identifiant principal) : saisissez **`tutorial_user`**.
   - Auto generate a password (Génération automatique d'un mot de passe) : laissez cette option désactivée.
   - Master password (Mot de passe principal) : saisissez un mot de passe.
   - Confirm password (Confirmer le mot de passe) – Saisissez à nouveau le mot de passe.

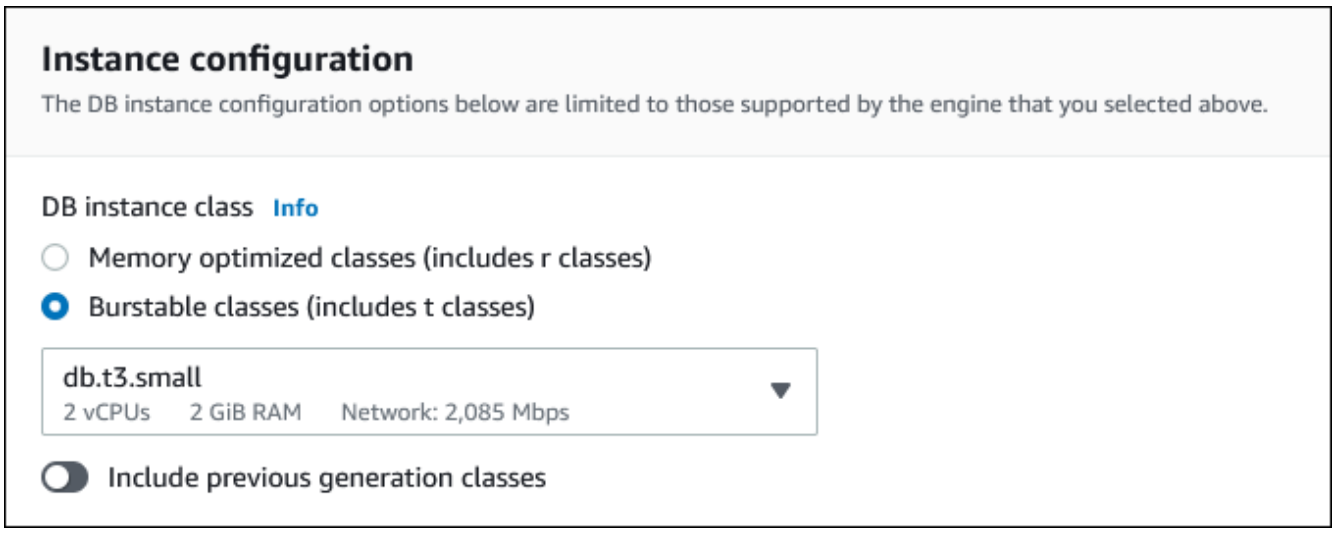


9. Dans la section Instance configuration (Configuration de l'instance), définissez les valeurs suivantes :

   - Classe à capacité extensible (inclut les classes t)

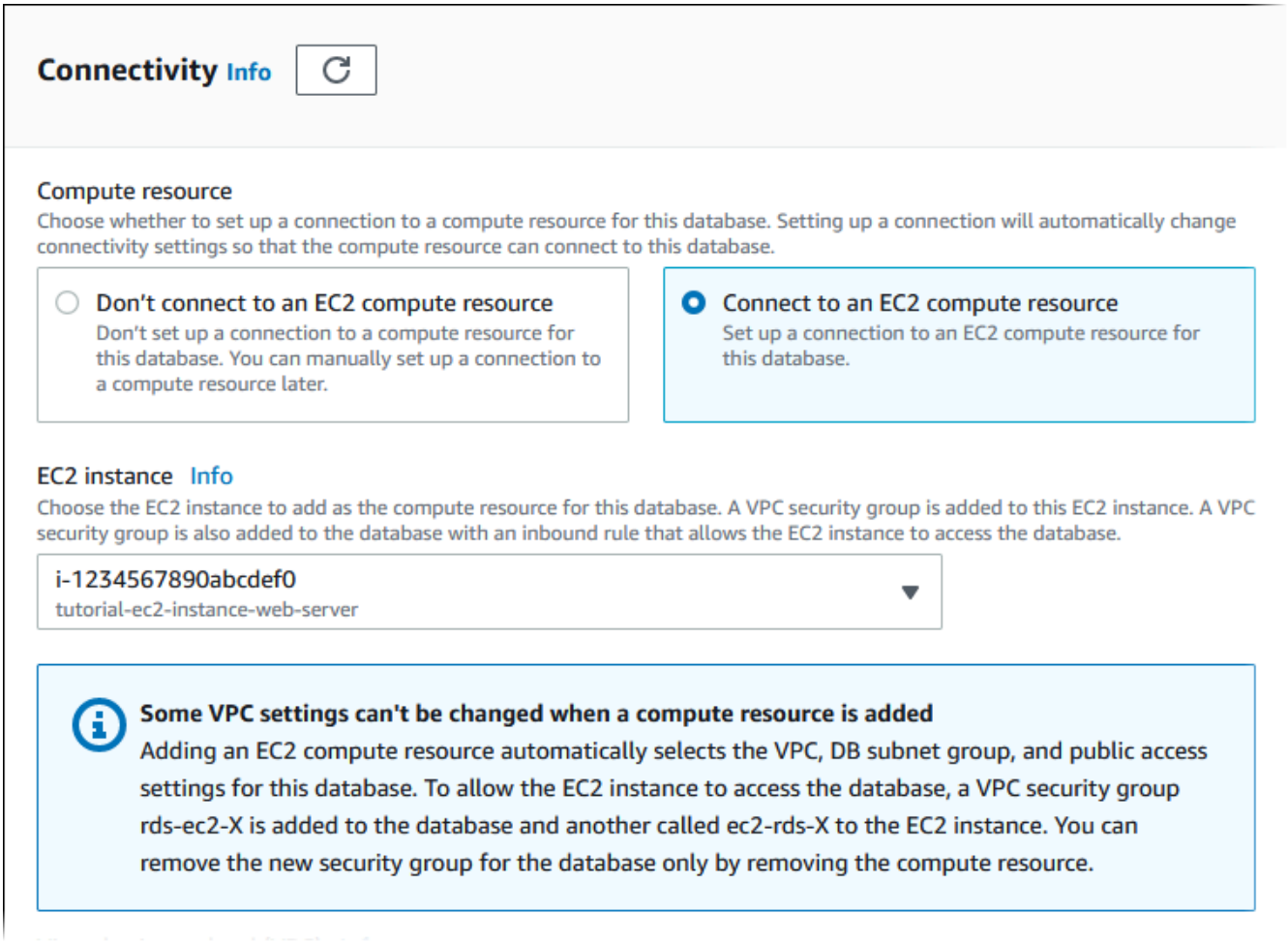
- db.t3.small ou db.t3.medium

> **Note**
>
> Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à la production. Pour plus de détails sur les classes d'instance T, consultez Types de classes d'instance de base de données.



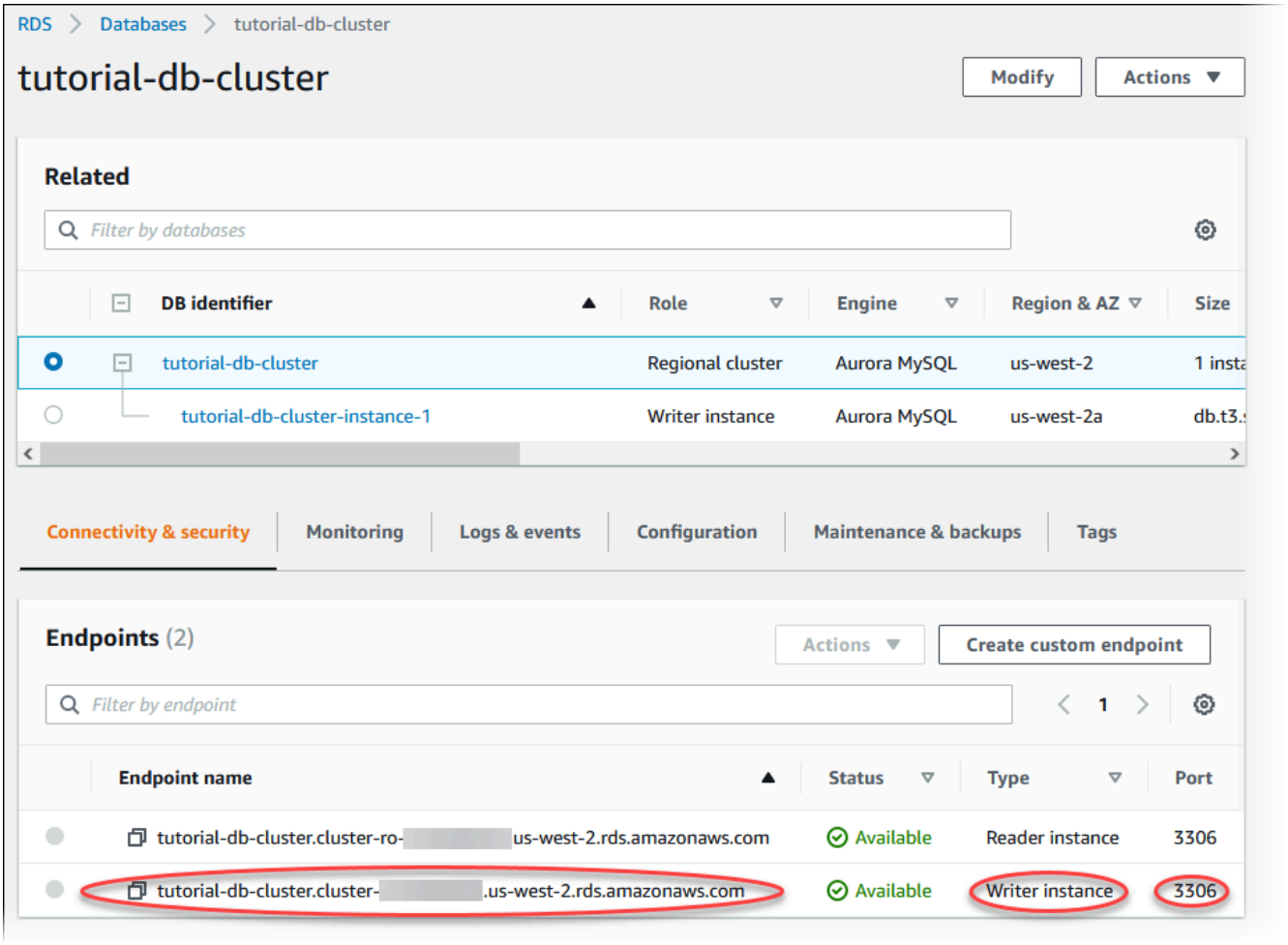
10. Dans la section Availability and durability (Disponibilité et durabilité), utilisez les valeurs par défaut.
11. Dans la section Connectivity (Connectivité), définissez ces valeurs et conservez les autres valeurs par défaut :

    - Pour Compute resource (Ressources de calcul), choisissez Connect to an EC2 compute resource (Se connecter à une ressource de calcul EC2).
    - Pour EC2 instance (Instance EC2), choisissez l'instance EC2 que vous avez créée précédemment, telle que tutorial-ec2-instance-web-server.



12. Ouvrez la section Additional configuration (Configuration supplémentaire), puis entrez **`sample`** pour Initial database name (Nom de la base de données initiale). Conservez les paramètres par défaut pour les autres options.
13. Pour créer votre cluster de bases de données Aurora MySQL, choisissez Créer une base de données.

    Votre nouveau cluster de bases de données apparaît dans la liste Bases de données avec l'état Création en cours.
14. Attendez que l'État de votre nouveau cluster de bases de données affiche Disponible. Sélectionnez ensuite le nom du cluster de bases de données pour afficher les détails.
15. Dans la section Connectivité et sécurité, affichez le Point de terminaison et le Port de l'instance de base de données de rédacteur.



Notez le point de terminaison et le port de votre instance de base de données de rédacteur. Vous utilisez ces informations pour connecter votre serveur web à votre cluster de bases de données.

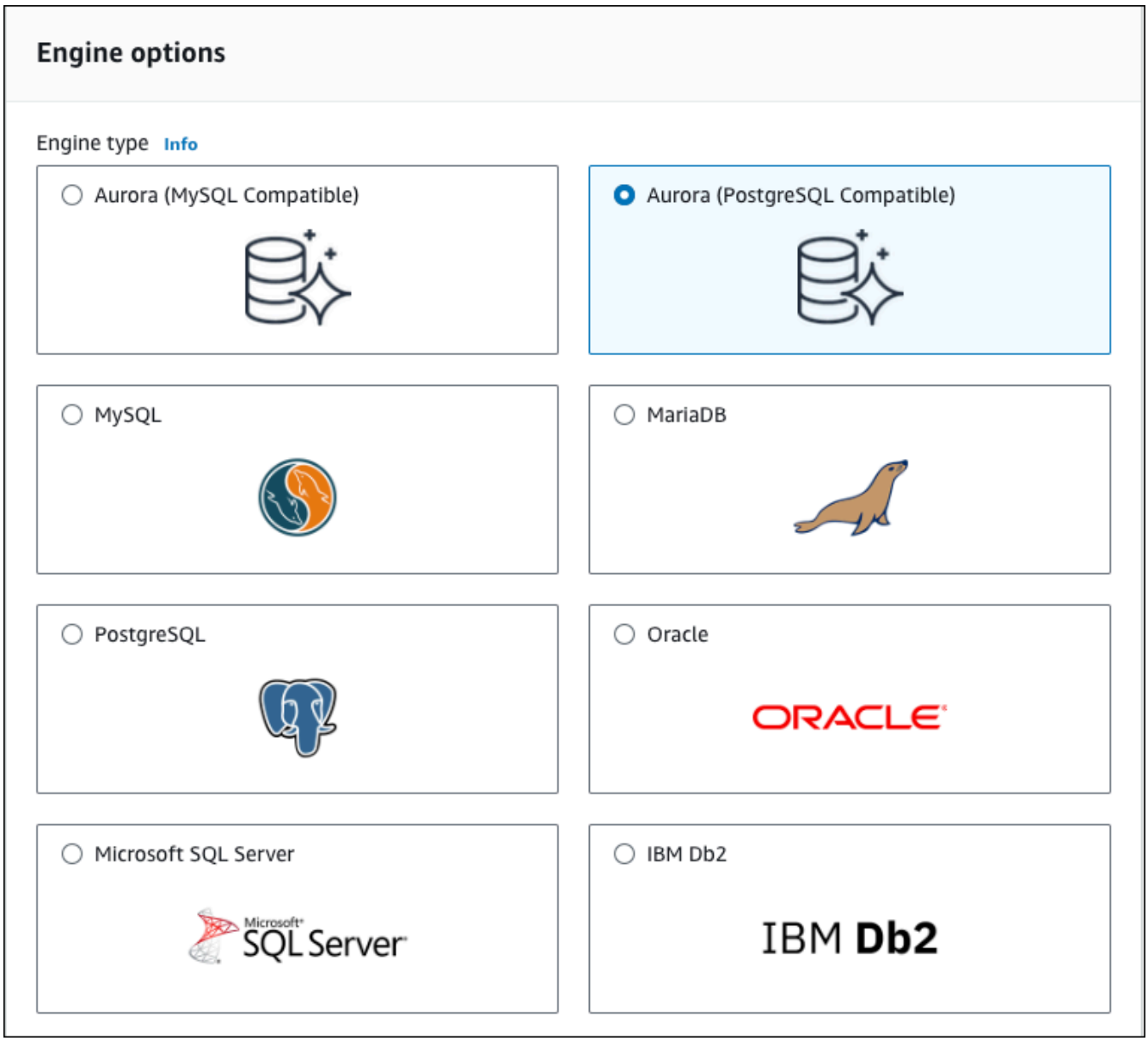
16. Termin Installer un serveur web sur votre instance EC2.

Aurora PostgreSQL

Pour créer un cluster de bases de données Aurora PostgreSQL

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le coin supérieur droit de la AWS Management Console, assurez-vous que la Région AWS est la même que celle où vous avez créé votre instance EC2.
3. Dans la panneau de navigation, choisissez Bases de données.
4. Choisissez Create database (Créer une base de données).

5. Sur la page Créer une base de données, choisissez Création standard.
6. Pour Options de moteur, choisissez Aurora (compatible avec PostgreSQL).

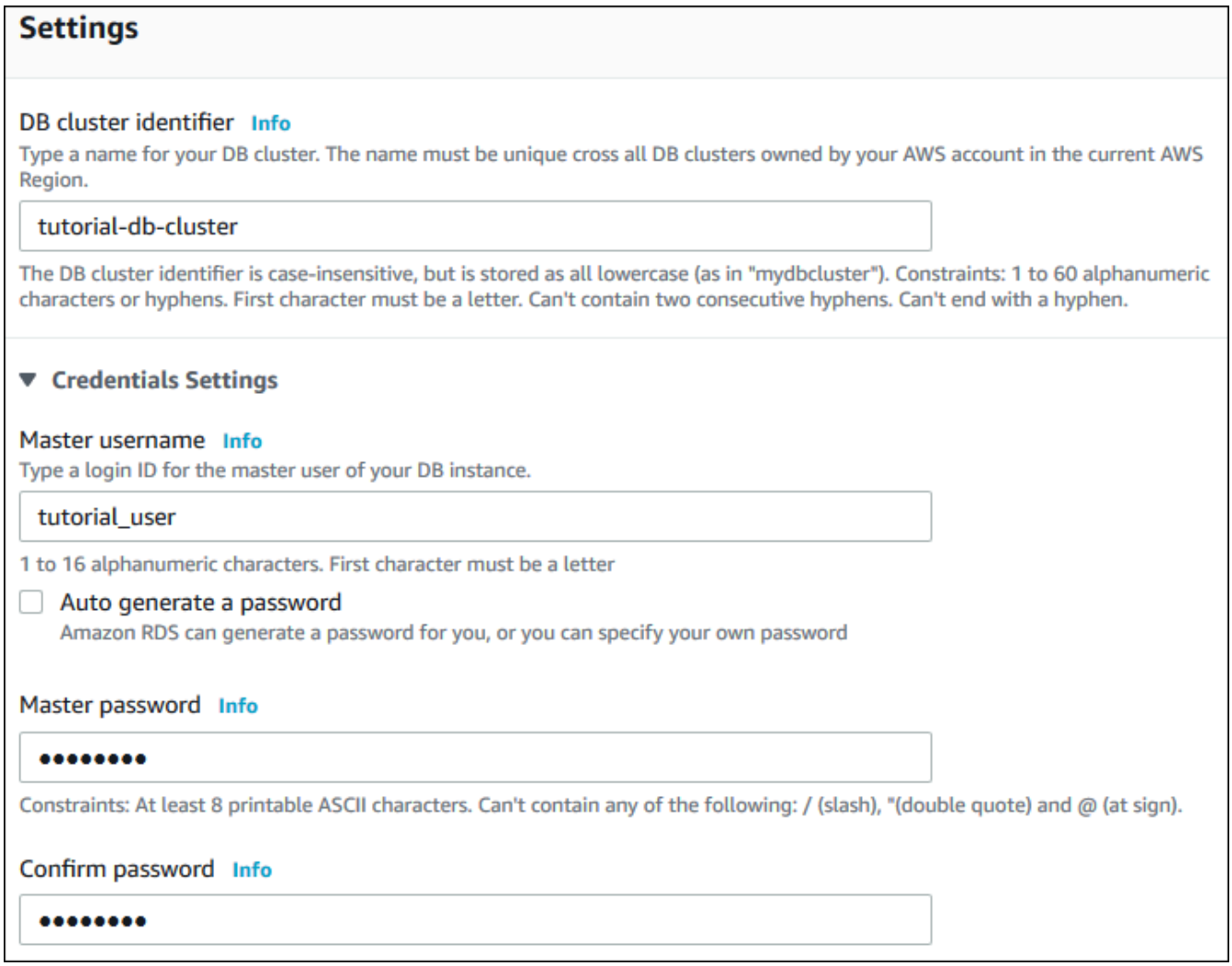


Conservez les valeurs par défaut pour Version et les autres options du moteur.

7. Dans la section Templates (Modèles), choisissez Dev/Test.

8. Dans la section Settings (Paramètres), définissez les valeurs suivantes :

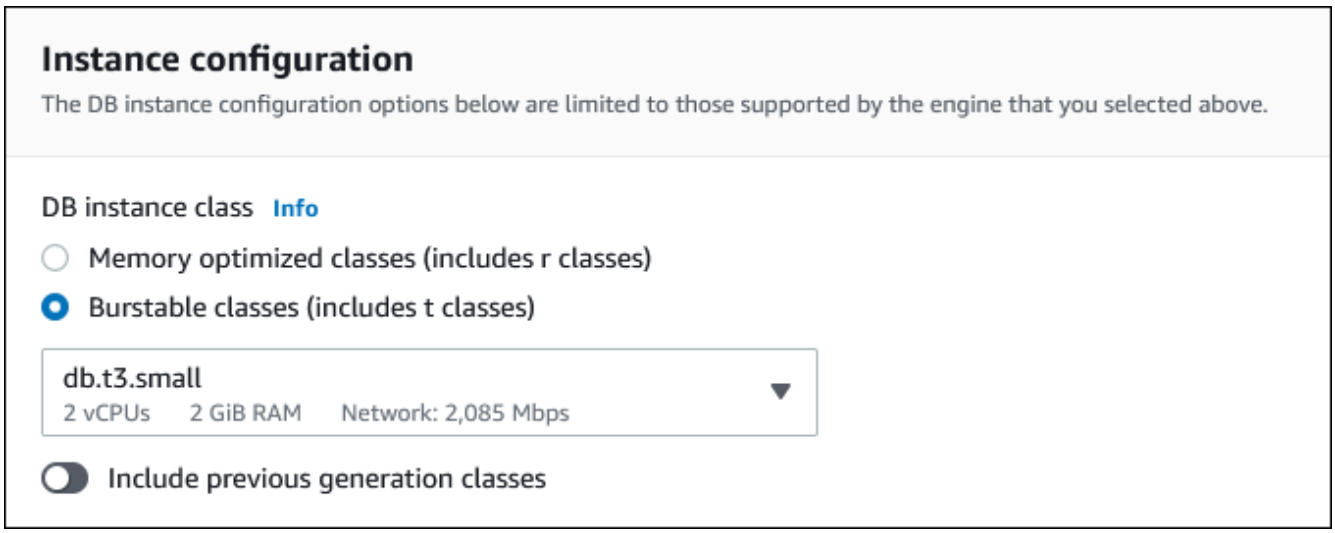
   - Identifiant du cluster de bases de données : saisissez **`tutorial-db-cluster`**.
   - Master username (Identifiant principal) : saisissez **`tutorial_user`**.
   - Auto generate a password (Génération automatique d'un mot de passe) : laissez cette option désactivée.
   - Master password (Mot de passe principal) : saisissez un mot de passe.
   - Confirm password (Confirmer le mot de passe) – Saisissez à nouveau le mot de passe.



9. Dans la section Instance configuration (Configuration de l'instance), définissez les valeurs suivantes :

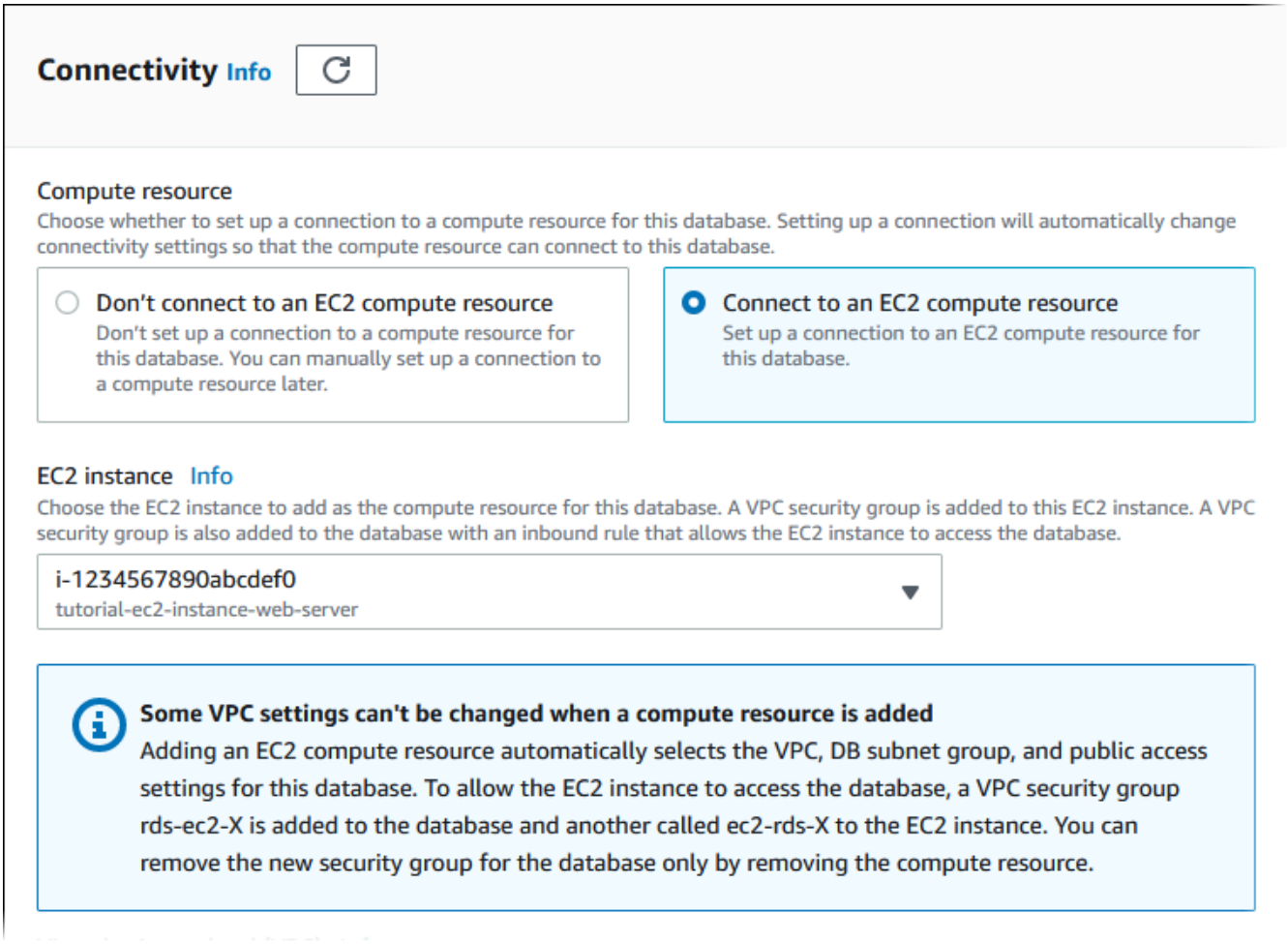
   - Classe à capacité extensible (inclut les classes t)
   - db.t3.small ou db.t3.medium

   > **Note**
   >
   > Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à la production. Pour plus de détails sur les classes d'instance T, consultez Types de classes d'instance de base de données.



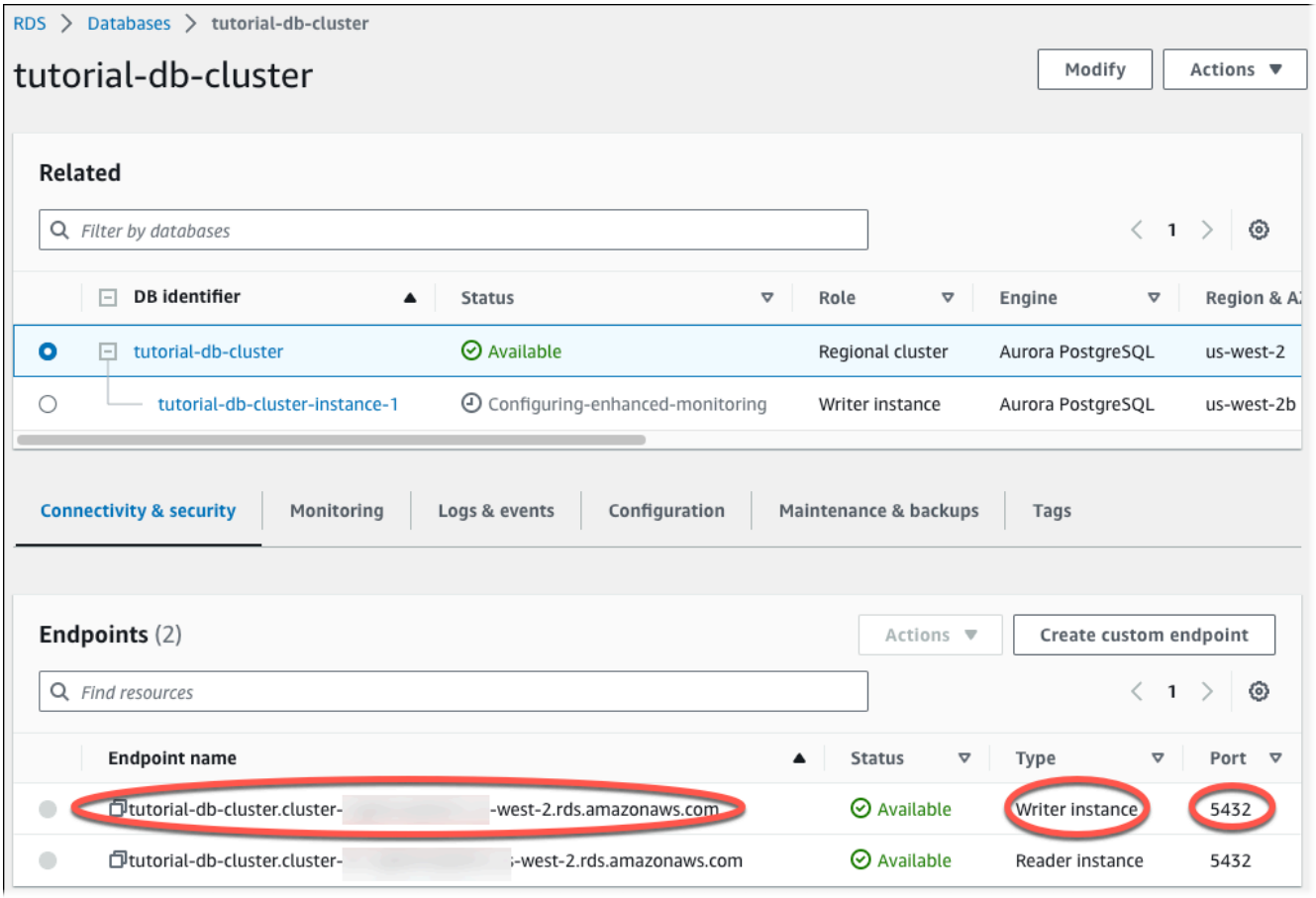
10. Dans la section Availability and durability (Disponibilité et durabilité), utilisez les valeurs par défaut.
11. Dans la section Connectivity (Connectivité), définissez ces valeurs et conservez les autres valeurs par défaut :

    - Pour Compute resource (Ressources de calcul), choisissez Connect to an EC2 compute resource (Se connecter à une ressource de calcul EC2).
    - Pour EC2 instance (Instance EC2), choisissez l'instance EC2 que vous avez créée précédemment, telle que tutorial-ec2-instance-web-server.



12. Ouvrez la section Additional configuration (Configuration supplémentaire), puis entrez **`sample`** pour Initial database name (Nom de la base de données initiale). Conservez les paramètres par défaut pour les autres options.
13. Pour créer votre cluster de bases de données Aurora PostgreSQL, choisissez Créer une base de données.

    Votre nouveau cluster de bases de données apparaît dans la liste Bases de données avec l'état Création en cours.
14. Attendez que l'État de votre nouveau cluster de bases de données affiche Disponible. Sélectionnez ensuite le nom du cluster de bases de données pour afficher les détails.
15. Dans la section Connectivité et sécurité, affichez le Point de terminaison et le Port de l'instance de base de données de rédacteur.



Notez le point de terminaison et le port de votre instance de base de données de rédacteur. Vous utilisez ces informations pour connecter votre serveur web à votre cluster de bases de données.

16. Termin Installer un serveur web sur votre instance EC2.

Installer un serveur web sur votre instance EC2

Installez un serveur Web sur l'instance EC2 que vous avez créée dans Lancement d'une instance EC2 pour vous connecter à votre cluster de bases de données. Le serveur Web se connecte au cluster de bases de données Amazon Aurora que vous avez créé dans Créer un cluster de bases de données Amazon Aurora.

Installer un serveur Web Apache avec PHP et MariaDB

Connectez-vous à votre instance EC2 et installez le serveur web.

Pour vous connecter à votre instance EC2 et installer le serveur Web Apache avec PHP

1. Connectez-vous à l'instance EC2 que vous avez précédemment créée en suivant la procédure spécifiée dans Connectez-vous à votre instance Linux dans le Guide de l'utilisateur Amazon EC2.

   Nous vous recommandons de vous connecter à votre instance EC2 en utilisant SSH. Si l'utilitaire client SSH est installé sur Windows, Linux ou Mac, vous pouvez vous connecter à l'instance à l'aide du format de commande suivant :

   ```
   ssh -i location_of_pem_file ec2-user@ec2-instance-public-dns-name
   ```

   Supposons, par exemple, que `ec2-database-connect-key-pair.pem` soit stocké dans `/dir1` sur Linux et que le DNS IPv4 public de votre instance EC2 soit `ec2-12-345-678-90.compute-1.amazonaws.com`. Votre commande SSH se présenterait comme suit :

   ```
   ssh -i /dir1/ec2-database-connect-key-pair.pem ec2-
   user@ec2-12-345-678-90.compute-1.amazonaws.com
   ```

2. Obtenez les dernières corrections de bogues et mises à jour de sécurité en mettant à jour le logiciel sur votre instance EC2. Pour ce faire, exécutez la commande suivante.

   > **Note**
   >
   > L'option `-y` installe les mises à jour sans demander de confirmation. Pour examiner les mises à jour avant de les installer, omettez cette option.

   ```
   sudo dnf update -y
   ```

3. Une fois les mises à jour terminées, installez le serveur web Apache, PHP et le logiciel MariaDB ou PostgreSQL à l'aide des commandes suivantes. Cette commande installe plusieurs packages logiciels et les dépendances connexes au même moment.

   MariaDB & MySQL

   ```
   sudo dnf install -y httpd php php-mysqli mariadb105
   ```

PostgreSQL

```
sudo dnf install -y httpd php php-pgsql postgresql15
```

Si vous recevez une erreur, votre instance n'a probablement pas été lancée avec une AMI Amazon Linux 2023. Vous utilisez peut-être une AMI Amazon Linux 2 à la place. Vous pouvez afficher votre version d'Amazon Linux avec la commande suivante

```
cat /etc/system-release
```

Pour plus d'informations, consultez Mise à jour du logiciel de l'instance.

4. Démarrez le serveur web avec la commande illustrée ci-dessous.

```
sudo systemctl start httpd
```

Vous pouvez vérifier que votre serveur web est correctement installé et démarré. Pour ce faire, saisissez le nom DNS (Domain Name System) public de votre instance EC2 dans la barre d'adresse d'un navigateur web, par exemple : `http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com`. Si votre serveur Web est en cours d'exécution, vous voyez la page de test Apache.

Si la page de test Apache ne s'affiche pas, vérifiez vos règles entrantes pour le groupe de sécurité du VPC que vous avez créé dans Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement). Assurez-vous que vos règles entrantes incluent un accès HTTP (port 80) à l'adresse IP pour vous connecter au serveur Web.

> **Note**
>
> La page de test Apache apparaît uniquement en l'absence de contenu dans le répertoire racine des documents, `/var/www/html`. Après l'ajout de contenu dans le répertoire racine des documents, votre contenu apparaît à l'adresse DNS publique de votre instance EC2. Avant cela, il apparaît sur la page de test d'Apache.

5. Configurez le serveur web pour qu'il démarre à chaque redémarrage du système à l'aide de la commande `systemctl`.

```
sudo systemctl enable httpd
```

Pour autoriser `ec2-user` à gérer les fichiers dans le répertoire racine par défaut pour votre serveur Web Apache, modifiez l'appartenance et les autorisations du répertoire `/var/www`. Il existe plusieurs façons d'accomplir cette tâche. Dans ce didacticiel, vous ajoutez l'utilisateur `ec2-user` au groupe `apache` pour donner au groupe `apache` la propriété du répertoire `/var/www` et attribuer les autorisations d'écriture au groupe.

Pour définir les autorisations sur les fichiers pour le serveur Web Apache

1. Ajoutez l'utilisateur `ec2-user` au groupe `apache`.

   ```
   sudo usermod -a -G apache ec2-user
   ```

2. Pour actualiser vos autorisations et inclure le nouveau groupe `apache`, déconnectez-vous.

   ```
   exit
   ```

3. Reconnectez-vous et vérifiez que le groupe `apache` existe à l'aide de la commande `groups`.

   ```
   groups
   ```

   Votre sortie se présente comme suit :

   ```
   ec2-user adm wheel apache systemd-journal
   ```

4. Remplacez le groupe propriétaire du répertoire `/var/www` et de son contenu par le groupe `apache`.

   ```
   sudo chown -R ec2-user:apache /var/www
   ```

5. Modifiez les autorisations des répertoires `/var/www` et de ses sous-répertoires pour ajouter des autorisations d'écriture de groupe et définir l'ID de groupe sur les sous-répertoires créés à l'avenir.

   ```
   sudo chmod 2775 /var/www
   find /var/www -type d -exec sudo chmod 2775 {} \;
   ```

6. Modifiez de façon récursive les autorisations pour les fichiers figurant dans le répertoire `/var/www` et ses sous-répertoires pour ajouter des autorisations d'écriture de groupe.

```
find /var/www -type f -exec sudo chmod 0664 {} \;
```

Maintenant, `ec2-user` (et tous les futurs membres du groupe `apache`) peut ajouter, supprimer et modifier les fichiers à la racine du document Apache. Cela vous permet d'ajouter du contenu, tel qu'un site Web statique ou une application PHP.

> **Note**
>
> Un serveur web exécutant le protocole HTTP ne fournit aucune sécurité de transport pour les données qu'il envoie ou reçoit. Lorsque vous vous connectez à un serveur HTTP via un navigateur Web, de nombreuses informations peuvent être vues par des personnes malveillantes sur le chemin d'accès réseau. Ces informations incluent les URL que vous visitez, le contenu des pages Web que vous recevez et le contenu (y compris les mots de passe) de tous les formulaires HTML.
> Les bonnes pratiques en matière de sécurisation de votre serveur Web consistent à installer la prise en charge HTTPS (HTTP Secure). Ce protocole protège vos données avec le chiffrement SSL/TLS. Pour plus d'informations, consultez Didacticiel : Configurer SSL/TLS avec l'AMI Amazon Linux dans le Guide de l'utilisateur Amazon EC2.

Connecter le serveur web Apache au cluster de bases de données

Ensuite, vous allez ajouter du contenu à votre serveur web Apache qui se connecte à votre cluster de bases de données Amazon Aurora.

Pour ajouter du contenu au serveur web Apache qui se connecte à votre cluster de bases de données

1. Alors que vous êtes encore connecté à votre instance EC2, remplacez le répertoire par `/var/www` et créez un sous-répertoire nommé `inc`.

```
cd /var/www
mkdir inc
cd inc
```

2. Créez un fichier dans le répertoire `inc` nommé `dbinfo.inc`, puis modifiez le fichier en appelant nano (ou l'éditeur de votre choix).

```
>dbinfo.inc
nano dbinfo.inc
```

3. Ajoutez le contenu suivant au fichier `dbinfo.inc`. Ici, *`db_instance_endpoint`* est le point de terminaison de l'enregistreur du cluster de bases de données, sans le port, pour votre cluster de bases de données.

> **Note**
>
> Nous vous recommandons de placer les informations de nom d'utilisateur et de mot de passe dans un dossier ne faisant pas partie de la racine du document de votre serveur web. Vous réduisez ainsi la possibilité d'exposer vos informations de sécurité. Veillez à remplacer `master password` par un mot de passe approprié dans votre application.

```
<?php

define('DB_SERVER', 'db_cluster_writer_endpoint');
define('DB_USERNAME', 'tutorial_user');
define('DB_PASSWORD', 'master password');
define('DB_DATABASE', 'sample');
?>
```

4. Enregistrez et fermez le fichier `dbinfo.inc`. Si vous utilisez nano, enregistrez et fermez le fichier à l'aide des touches Ctrl+S et Ctrl+X.
5. Remplacez le répertoire par `/var/www/html`.

```
cd /var/www/html
```

6. Créez un fichier dans le répertoire `html` nommé `SamplePage.php`, puis modifiez le fichier en appelant nano (ou l'éditeur de votre choix).

```
>SamplePage.php
nano SamplePage.php
```

7. Ajoutez le contenu suivant au fichier `SamplePage.php` :

   MariaDB & MySQL

```
<?php include "../inc/dbinfo.inc"; ?>
<html>
<body>
<h1>Sample page</h1>
<?php

  /* Connect to MySQL and select the database. */
  $connection = mysqli_connect(DB_SERVER, DB_USERNAME, DB_PASSWORD);

  if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " .
 mysqli_connect_error();

  $database = mysqli_select_db($connection, DB_DATABASE);

  /* Ensure that the EMPLOYEES table exists. */
  VerifyEmployeesTable($connection, DB_DATABASE);

  /* If input fields are populated, add a row to the EMPLOYEES table. */
  $employee_name = htmlentities($_POST['NAME']);
  $employee_address = htmlentities($_POST['ADDRESS']);

  if (strlen($employee_name) || strlen($employee_address)) {
    AddEmployee($connection, $employee_name, $employee_address);
  }
?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
        <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
      <td>
        <input type="text" name="ADDRESS" maxlength="90" size="60" />
```

```
      </td>
      <td>
        <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>

<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = mysqli_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = mysqli_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>

</table>

<!-- Clean up. -->
<?php

  mysqli_free_result($result);
  mysqli_close($connection);

?>

</body>
</html>


<?php
```

```
/* Add an employee to the table. */
function AddEmployee($connection, $name, $address) {
   $n = mysqli_real_escape_string($connection, $name);
   $a = mysqli_real_escape_string($connection, $address);

   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!mysqli_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID int(11) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!mysqli_query($connection, $query)) echo("<p>Error creating table.</
p>");
  }
}

/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = mysqli_real_escape_string($connection, $tableName);
  $d = mysqli_real_escape_string($connection, $dbName);

  $checktable = mysqli_query($connection,
      "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME = '$t'
 AND TABLE_SCHEMA = '$d'");

  if(mysqli_num_rows($checktable) > 0) return true;

  return false;
}
?>
```

PostgreSQL

```
<?php include "../inc/dbinfo.inc"; ?>

<html>
<body>
<h1>Sample page</h1>
<?php

/* Connect to PostgreSQL and select the database. */
$constring = "host=" . DB_SERVER . " dbname=" . DB_DATABASE . " user=" .
 DB_USERNAME . " password=" . DB_PASSWORD ;
$connection = pg_connect($constring);

if (!$connection){
 echo "Failed to connect to PostgreSQL";
 exit;
}

/* Ensure that the EMPLOYEES table exists. */
VerifyEmployeesTable($connection, DB_DATABASE);

/* If input fields are populated, add a row to the EMPLOYEES table. */
$employee_name = htmlentities($_POST['NAME']);
$employee_address = htmlentities($_POST['ADDRESS']);

if (strlen($employee_name) || strlen($employee_address)) {
  AddEmployee($connection, $employee_name, $employee_address);
}

?>

<!-- Input form -->
<form action="<?PHP echo $_SERVER['SCRIPT_NAME'] ?>" method="POST">
  <table border="0">
    <tr>
      <td>NAME</td>
      <td>ADDRESS</td>
    </tr>
    <tr>
      <td>
    <input type="text" name="NAME" maxlength="45" size="30" />
      </td>
```

```
      <td>
    <input type="text" name="ADDRESS" maxlength="90" size="60" />
      </td>
      <td>
    <input type="submit" value="Add Data" />
      </td>
    </tr>
  </table>
</form>
<!-- Display table data. -->
<table border="1" cellpadding="2" cellspacing="2">
  <tr>
    <td>ID</td>
    <td>NAME</td>
    <td>ADDRESS</td>
  </tr>

<?php

$result = pg_query($connection, "SELECT * FROM EMPLOYEES");

while($query_data = pg_fetch_row($result)) {
  echo "<tr>";
  echo "<td>",$query_data[0], "</td>",
       "<td>",$query_data[1], "</td>",
       "<td>",$query_data[2], "</td>";
  echo "</tr>";
}
?>
</table>

<!-- Clean up. -->
<?php

  pg_free_result($result);
  pg_close($connection);
?>
</body>
</html>


<?php

/* Add an employee to the table. */
```

```
function AddEmployee($connection, $name, $address) {
   $n = pg_escape_string($name);
   $a = pg_escape_string($address);
   echo "Forming Query";
   $query = "INSERT INTO EMPLOYEES (NAME, ADDRESS) VALUES ('$n', '$a');";

   if(!pg_query($connection, $query)) echo("<p>Error adding employee data.</
p>");
}

/* Check whether the table exists and, if not, create it. */
function VerifyEmployeesTable($connection, $dbName) {
  if(!TableExists("EMPLOYEES", $connection, $dbName))
  {
     $query = "CREATE TABLE EMPLOYEES (
         ID serial PRIMARY KEY,
         NAME VARCHAR(45),
         ADDRESS VARCHAR(90)
       )";

     if(!pg_query($connection, $query)) echo("<p>Error creating table.</p>");
  }
}
/* Check for the existence of a table. */
function TableExists($tableName, $connection, $dbName) {
  $t = strtolower(pg_escape_string($tableName)); //table name is case sensitive
  $d = pg_escape_string($dbName); //schema is 'public' instead of 'sample' db
 name so not using that

  $query = "SELECT TABLE_NAME FROM information_schema.TABLES WHERE TABLE_NAME =
 '$t';";
  $checktable = pg_query($connection, $query);

  if (pg_num_rows($checktable) >0) return true;
  return false;

}
?>
```

8. Enregistrez et fermez le fichier `SamplePage.php`.
9. Vérifiez que votre serveur web se connecte avec succès à votre cluster de bases de données en ouvrant un navigateur web et en accédant à une page `http://`*`EC2 instance`*

endpoint/SamplePage.php, par exemple : `http://ec2-12-345-67-890.us-west-2.compute.amazonaws.com/SamplePage.php`.

Vous pouvez utiliser `SamplePage.php` pour ajouter des données à votrecluster de bases de données. Les données que vous ajoutez sont ensuite affichées sur la page. Pour vérifier que les données ont été insérées dans la table, installez le client MySQL sur l'instance Amazon EC2. Connectez-vous ensuite au cluster de bases de données et interrogez la table.

Pour plus d'informations sur la connexion au cluster de bases de données, consultez Connexion à un cluster de bases de données Amazon Aurora.

Pour vérifier que la sécurité de votre cluster de bases de données est assurée, contrôlez que les sources extérieures du VPC ne peuvent pas se connecter à votre cluster de bases de données.

Après avoir testé votre serveur Web et votre base de données, vous devez supprimer votre cluster de bases de données et votre instance Amazon EC2.

- Pour supprimer un cluster de bases de données, suivez les instructions de la section Suppression de clusters de bases de données Aurora et d'instances de bases de données. Vous n'avez pas besoin de créer un instantané final.
- Pour résilier une instance Amazon EC2, suivez les instructions de la page Résilier votre instance dans le Guide de l'utilisateur Amazon EC2.

Tutoriels Amazon Aurora et exemple de code

La AWS documentation inclut plusieurs didacticiels qui vous guident à travers les cas d'utilisation courants d' et d'Amazon Aurora. La plupart de ces didacticiels vous montrent comment utiliser (Amazon Aurora) avec d'autres AWS services. En outre, vous pouvez accéder à un exemple de code dans GitHub.

> **Note**
>
> Vous pouvez trouver d’autres tutoriels sur le Blog AWS de base de données. Pour plus d’informations sur la formation, consultez AWS Training and Certification.

Rubriques

Tutoriels dans ce guide

Les tutoriels suivants dans ce guide montrent comment exécuter les tâches courantes à l’aide d’Amazon Aurora :

- Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement)

  Découvrez comment inclure un cluster de bases de données dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC. Dans ce cas, le VPC partage des données avec un serveur Web qui s'exécute sur une EC2 instance Amazon du même VPC.
- Tutoriel : Créer un VPC à utiliser avec un cluster de bases de données (mode double-pile)

Découvrez comment inclure un cluster de bases de données dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC. Dans ce cas, le VPC partage des données avec une EC2 instance Amazon du même VPC. Dans ce tutoriel, vous créez le VPC pour ce scénario qui fonctionne avec une base de données en mode double pile.

- Didacticiel : Créer un serveur web et une cluster de base de données Amazon Aurora

  Apprenez à installer un serveur web Apache avec PHP et à créer une base de données MySQL. Le serveur Web s'exécute sur une EC2 instance Amazon utilisant Amazon Linux, et la base de données MySQL est un cluster de base de données Aurora MySQL. L' EC2 instance Amazon et le cluster d' de base de données s'exécutent dans un Amazon VPC.

- Tutoriel : restaurez un cluster de bases de données Amazon Aurora à partir d'un instantané de cluster de bases de données

  Découvrez comment effectuer une restauration à partir d'un instantané de cluster de bases de données.

- Didacticiel : Utilisation de balises pour spécifier les clusters de bases de données Aurora à arrêter

  Apprenez à utiliser des balises pour spécifier les clusters de bases de données Aurora à arrêter.

- Tutoriel : journaliser les changements d'état de l'instance de base de données à l'aide d'Amazon EventBridge

  Découvrez comment enregistrer un changement d'état d'une instance de base de données à l'aide d'Amazon EventBridge et AWS Lambda.

Tutoriels dans d'autres AWS guides

Les didacticiels suivants, présentés dans d'autres AWS guides, vous montrent comment effectuer des tâches courantes avec (Amazon Aurora) :

> **Note**
>
> Certains des tutoriels utilisent des instances de base de données Amazon RDS, mais elles peuvent être adaptées pour utiliser des clusters de bases de données Aurora.

- Tutoriel : Aurora sans serveur dans le Guide du développeur AWS AppSync

Découvrez comment fournir une source AWS AppSync de données permettant d'exécuter des commandes SQL sur des clusters de Aurora Serverless bases de données avec l'API de données activée. Vous pouvez utiliser les résolveurs AWS AppSync pour exécuter des instructions SQL sur l'API de données avec des requêtes, des mutations et des abonnements GraphQL.

- Tutoriel : Rotation d'un secret pour une AWS base de données dans le guide de AWS Secrets Manager l'utilisateur

  Apprenez à créer un secret pour une AWS base de données et à configurer le secret pour qu'il alterne selon un calendrier. Vous déclenchez une rotation manuellement, puis vous vérifiez que la nouvelle version du secret continue de fournir l'accès.

- Tutoriels et exemples dans le Manuel du développeur AWS Elastic Beanstalk

  Découvrez comment déployer des applications qui utilisent des bases de données Amazon RDS avec AWS Elastic Beanstalk.

- Utilisation des données d'une base de données Amazon RDS pour créer une source de données Amazon ML dans le Amazon Machine Learning Developer Guide

  Apprenez à créer un objet de source de données Amazon Machine Learning (Amazon ML) à partir de données stockées dans une instance de base de données MySQL.

- Activation manuelle de l'accès à une instance Amazon RDS dans un VPC dans le guide de l'utilisateur Amazon Quick Suite

  Découvrez comment activer l'accès de Quick Suite à une instance de base de données Amazon RDS dans un VPC.

Tutoriels et exemples de code dans GitHub

Les didacticiels et les exemples de code suivants vous GitHub montrent comment effectuer des tâches courantes avec Amazon Aurora :

- Création d'une bibliothèque de prêt Aurora Serverless v2

  Apprenez à créer une application de bibliothèque de prêt où les clients peuvent emprunter et rendre des livres. L'exemple utilise Aurora Serverless v2 et AWS SDK pour Python (Boto3).

- Création d'une application de suivi des articles Amazon Aurora avec une API REST Spring qui interroge les données Aurora Serverless v2 à l'aide du kit SDK pour Java 2.x.

Découvrez comment créer une API REST Spring qui interroge des données Aurora Serverless v2. Elle doit être utilisée par une application React qui utilise le kit SDK pour Java 2.x.

- Création d'une application de suivi des articles Amazon Aurora qui interroge les Aurora Serverless v2 données à l'aide de AWS SDK pour PHP

  Découvrez comment créer une application qui utilise le `RdsDataClient` de l'API de données et Aurora Serverless v2 pour assurer le suivi et la création de rapports sur les éléments de travail. L'exemple utilise AWS SDK pour PHP.

- Création d'une application de suivi d'articles Amazon Aurora qui interroge les données Aurora Serverless v2 à l'aide de AWS SDK pour Python (Boto3)

  Découvrez comment créer une application qui utilise le `RdsDataClient` de l'API de données et Aurora Serverless v2 pour assurer le suivi et la création de rapports sur les éléments de travail. L'exemple utilise AWS SDK pour Python (Boto3).

AWS Livre de recettes de base de données

Le livre de recettes AWS DB est un guide de base de données complet qui vous explique comment créer, déployer et gérer des solutions de base de données performantes et rentables sur. AWS Step-by-stepdes didacticiels vous guident dans la création d'applications prêtes pour la production et dans le déploiement des applications à l'aide CloudFormation de modèles. Vous découvrirez les AWS services essentiels en construisant une infrastructure, en mettant en œuvre des réseaux, en développant des architectures sans serveur, en gérant des bases de données et en intégrant l'IA générative. Découvrez les AWS meilleures pratiques qui vous aident à créer des solutions sécurisées et évolutives tout en optimisant les coûts. Que vous soyez un débutant AWS ou un professionnel expérimenté, le livre de recettes AWS DB vous aide à développer les compétences nécessaires pour relever les défis courants liés aux bases de données et mettre en œuvre des solutions adaptées aux entreprises. Le guide pratique comprend les sections suivantes :

- Mise en route AWS pour les applications de base de données : découvrez AWS les principes de base tels que la configuration de votre compte et de l'environnement Jupyter Notebook.
- Principes fondamentaux des bases de données — Explorez les concepts essentiels des bases de données et comparez les services de AWS base de données afin de choisir la solution adaptée à vos charges de travail.
- Application Web sans serveur avec Amazon Aurora : créez une application de end-to-end vente au détail avec Amazon Aurora PostgreSQL qui gère les stocks, les commandes et les données clients.

- Surveillance et observabilité : configurez le suivi des performances, ainsi que des alertes pour identifier les problèmes potentiels liés aux bases de données avant qu'ils n'affectent vos applications.
- Mise à l'échelle avec Amazon Aurora : apprenez à créer des déploiements multirégionaux résilients avec Aurora DSQL, et à mettre à l'échelle vos bases de données pour augmenter la puissance de traitement ou à les répartir sur plusieurs instances pour augmenter la capacité.
- Optimisation des performances et des coûts : optimisez les performances de votre base de données et réduisez les coûts grâce à des stratégies de réglage éprouvées.
- Passez à des bases de données AWS spécialement conçues — Construisez une infrastructure sécurisée et fiable qui fait évoluer vos solutions d'IA générative et vos applications pilotées par les données, du prototype au déploiement en entreprise.
- Applications d'IA générative avec RAG : créez un système de recherche intelligente pour les documents d'assurance et de santé, qui utilise la génération à enrichissement contextuel (RAG) pour générer des résultats précis et contextuels.

AWS portail de contenu d'atelier et de laboratoire pour Amazon Aurora PostgreSQL

La collection suivante d'ateliers et d'autres contenus pratiques vous permet de mieux comprendre les fonctionnalités et capacités d'Amazon Aurora PostgreSQL :

- Création manuelle d'un cluster Aurora

  Découvrez comment créer manuellement un cluster Amazon Aurora PostgreSQL.
- Configuration de Cloud9 et initialisation de la base de données

  Découvrez comment configurer Cloud9 et initialiser la base de données PostgreSQL.
- Clonage rapide

  Découvrez comment créer un clone rapide Aurora.
- Gestion de plans de requêtes

  Découvrez comment contrôler les plans d'exécution pour un ensemble d'instructions à l'aide de la gestion des plans de requêtes.
- Gestion du cache du cluster

Découvrez la fonctionnalité de gestion du cache de cluster dans Aurora PostgreSQL.

- Diffusion d’activité de la base de données

  Découvrez comment surveiller et auditer l’activité de votre activité de base de données avec cette fonction.

- Utilisation de Performance Insights

  Découvrez comment surveiller et régler votre instance de base de données à l’aide de Performance Insights.

- Surveillance des performances avec les outils RDS

  Découvrez comment utiliser AWS les outils Postgres (Cloudwatch, Enhanced Monitoring, Slow Query Logs, Performance Insights, PostgreSQL Catalog Views) pour comprendre les problèmes de performances et identifier les moyens d'améliorer les performances de votre base de données.

- Réplicas en lecture d’autoscaling

  Découvrez le fonctionnement de l’autoscaling des réplicas en lecture Aurora dans la pratique à l’aide d’un script générateur de charge.

- Test de la tolérance aux pannes

  Découvrez comment un cluster de bases de données peut tolérer une panne.

- Aurora Global Database

  En savoir plus sur Aurora Global Database.

- Utilisation du machine learning

  En savoir plus sur le machine learning Aurora.

- Aurora sans serveur v2

  En savoir plus sur Aurora sans serveur v2.

- Trusted Language Extensions pour Aurora PostgreSQL

  Découvrez comment créer des extensions hautes performances qui s’exécutent en toute sécurité sur Aurora PostgreSQL.

AWS portail de contenu d'atelier et de laboratoire pour Amazon Aurora MySQL

La collection suivante d’ateliers et d’autres contenus pratiques vous permet de mieux comprendre les fonctionnalités et capacités d’Amazon Aurora MySQL :

- Création d’un cluster Aurora

  Découvrez comment créer manuellement un cluster Amazon Aurora MySQL.
- Création d’un environnement IDE basé sur le cloud Cloud9 pour vous connecter à votre base de données

  Découvrez comment configurer Cloud9 et initialiser la base de données MySQL.
- Clonage rapide

  Découvrez comment créer un clone rapide Aurora.
- Retour en arrière d’un cluster

  Découvrez comment effectuer un retour en arrière d’un cluster de bases de données.
- Utilisation de Performance Insights

  Découvrez comment surveiller et régler votre instance de base de données à l’aide de Performance Insights.
- Surveillance des performances avec les outils RDS

  Apprenez à utiliser AWS les outils SQL pour comprendre les problèmes de performances et identifier les moyens d'améliorer les performances de votre base de données.
- Analyser les performances des requêtes

  Découvrez comment résoudre les problèmes liés aux performances SQL à l’aide de différents outils.
- Réplicas en lecture d’autoscaling

  Découvrez le fonctionnement des réplicas en lecture d’autoscaling.
- Test de la tolérance aux pannes

  Découvrez les fonctions de haute disponibilité et de tolérance aux pannes d’Aurora MySQL.

- Aurora Global Database

  En savoir plus sur Aurora Global Database.
- Aurora sans serveur v2

  En savoir plus sur Aurora sans serveur v2.
- Utilisation du machine learning

  En savoir plus sur le machine learning Aurora.

Utilisation de ce service avec un AWS SDK

AWS des kits de développement logiciel (SDKs) sont disponibles pour de nombreux langages de programmation courants. Chaque SDK fournit une API, des exemples de code et de la documentation qui facilitent la création d'applications par les développeurs dans leur langage préféré.

Documentation SDK	Exemples de code
AWS SDK pour C++	AWS SDK pour C++ exemples de code
AWS CLI	AWS CLI exemples de code
AWS SDK pour Go	AWS SDK pour Go exemples de code
AWS SDK pour Java	AWS SDK pour Java exemples de code
AWS SDK pour JavaScript	AWS SDK pour JavaScript exemples de code
AWS SDK pour Kotlin	AWS SDK pour Kotlin exemples de code
AWS SDK pour .NET	AWS SDK pour .NET exemples de code
AWS SDK pour PHP	AWS SDK pour PHP exemples de code
Outils AWS pour PowerShell	Outils AWS pour PowerShell exemples de code
AWS SDK pour Python (Boto3)	AWS SDK pour Python (Boto3) exemples de code
AWS SDK pour Ruby	AWS SDK pour Ruby exemples de code

Documentation SDK	Exemples de code
AWS SDK pour Rust	AWS SDK pour Rust exemples de code
AWS SDK pour SAP ABAP	AWS SDK pour SAP ABAP exemples de code
AWS SDK pour Swift	AWS SDK pour Swift exemples de code

Pour voir des exemples spécifiques à ce service, consultez Exemples de code pour Aurora à l'aide d'Aurora AWS SDKs.

> **Exemple de disponibilité**
>
> Vous n’avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code en utilisant le lien Provide feedback (Fournir un commentaire) en bas de cette page.

Accès programmatique à Amazon Aurora

Aurora RDS met à votre disposition les outils suivants pour gérer vos ressources Amazon Aurora par programmation.

AWS Command Line Interface (AWS CLI)

Vous pouvez créer et gérer vos ressources Amazon Aurora à l'aide de l'AWS CLI dans un shell de ligne de commande. L'AWS CLI fournit un accès direct aux API des services AWS, tels qu'Amazon RDS. Pour obtenir la syntaxe et des exemples de commandes pour Amazon RDS, consultez rds dans la Référence de commandes de l'AWS CLI.

AWS CloudFormation

Avec cet outil d'infrastructure en tant que code (IaC) AWS, vous pouvez créer des modèles décrivant toutes les ressources Amazon Aurora souhaitées, puis provisionner AWS CloudFormation et configurer ces ressources pour vous. Pour plus d'informations, consultez the section called "Création de ressources avec AWS CloudFormation".

Kits de développement logiciel (SDK) AWS

AWS fournit des kits SDK pour de nombreuses technologies et de nombreux langages de programmation populaires. Ils vous permettent d'appeler plus facilement des services AWS depuis vos applications dans ce langage ou cette technologie. Pour plus d'informations sur ces kits SDK, consultez Outils de développement et de gestion d'applications sur AWS.

API Amazon RDS

Cette API est l'interface au niveau du protocole pour Amazon RDS. Lorsque vous utilisez cette API, vous devez formater correctement chaque demande HTTPS et ajouter une signature numérique valide à chaque demande. Pour plus d'informations, consultez Référence d'API Amazon RDS.

Console-to-Code

Cet outil vous permet de produire du code correspondant aux opérations réalisées dans la console Amazon RDS, afin de l'utiliser ensuite avec d'autres outils tels qu'AWS CloudFormation. Pour plus d'informations, consultez the section called "Console-to-Code".

API de données RDS

L'API de données RDS (API de données) fournit un accès programmatique aux données de vos bases de données Amazon Aurora sans serveur. Avec l'API de données, vous pouvez

utiliser un point de terminaison HTTP sécurisé pour exécuter des instructions SQL sans gérer de connexions. Pour plus d'informations, consultez Utilisation de l'API de données Amazon RDS.

Utilisez Console-to-Code pour générer du code pour les actions de votre console Amazon Aurora.

La console fournit un parcours guidé pour créer des ressources et tester des prototypes. Si vous souhaitez créer les mêmes ressources à grande échelle, vous aurez besoin d'un code d'automatisation. Console-to-Code est une fonctionnalité du service Amazon Q Developer qui peut vous aider à démarrer avec votre code d'automatisation. Console-to-Code enregistre les actions de votre console, y compris les valeurs par défaut et les paramètres que vous fournissez. Il utilise ensuite l'IA générative pour suggérer du code dans la langue et le format de votre choix pour les actions que vous choisissez. Étant donné que le flux de travail de la console permet de garantir que les valeurs de paramètres que vous spécifiez sont valides ensemble, le code que vous générez à l'aide de Console-to-Code possède des valeurs de paramètres compatibles. Vous pouvez utiliser le code comme point de départ, puis le personnaliser pour qu'il soit prêt pour la production en fonction de votre cas d'utilisation spécifique.

Par exemple, avec Console-to-Code, vous pouvez enregistrer la création d'un cluster de bases de données Aurora et choisir de générer du code au AWS CloudFormation format JSON. Vous pouvez ensuite copier ce code et le personnaliser pour l'utiliser dans votre modèle AWS CloudFormation.

Console-to-Code peut actuellement générer une infrastructure en tant que code (IaC) dans les langues et formats qui suivent :

- CDK Java
- CDK Python
- CDK TypeScript
- CloudFormation JSON
- CloudFormation YAML

Pour plus d'informations et obtenir des instructions sur l'utilisation de Console-to-Code, consultez Automatisation des services AWS avec Amazon Q Developer Console-to-Code dans le Guide de l'utilisateur Amazon Q Developer.

Configuration d’un cluster de bases de données Amazon Aurora

Cette section explique la manière de configurer votre cluster de bases de données Aurora. Avant de créer un cluster de bases de données Aurora, vous devez décider quelle instance de base de données exécutera le cluster de bases de données. Vous devez également déterminer l’emplacement de l’exécution du cluster de bases de données en choisissant une région AWS. Créez ensuite le cluster de bases de données. Si vous disposez d’un cluster à l’extérieur d’Aurora, vous pouvez migrer les données dans un cluster de bases de données Aurora.

Rubriques

Création d’un cluster de bases de données Amazon Aurora

Un cluster de bases de données Amazon Aurora se compose d’une instance de base de données, compatible avec MySQL ou PostgreSQL, et d’un volume de cluster qui contient les données du cluster de bases de données, copiées à travers trois zones de disponibilité comme un seul volume virtuel. Par défaut, un cluster de bases de données Aurora contient une instance de base de données primaire qui effectue les lectures et les écritures, et, en option, jusqu’à 15 réplicas Aurora (instances de base de données de lecteur). Pour plus d’informations sur les clusters de bases de données Aurora, consultez Clusters de bases de données Amazon Aurora.

Aurora possède deux principaux types de clusters de bases de données :

- Aurora approvisionné – vous choisissez la classe d’instance de base de données pour les instances d’écriture et de lecture en fonction de votre charge de travail attendue. Pour plus d’informations, consultez Classes d’instance de base de données Amazon Aurora. Aurora approvisionné dispose de plusieurs options, notamment des bases de données globales Aurora. Pour plus d’informations, consultez Utilisation d’Amazon Aurora Global Database.
- Aurora Serverless : Aurora Serverless v2 est une configuration de mise à l’échelle automatique à la demande pour Aurora. La capacité est ajustée automatiquement en fonction de la demande des applications. Seules les ressources consommées par votre cluster de bases de données vous sont facturées. Cette automatisation est particulièrement utile pour les environnements où les charges de travail sont très variables et imprévisibles. Pour plus d’informations, consultez Utiliser Aurora Serverless v2.

Dans la rubrique suivante, vous découvrirez comment créer un cluster de bases de données Aurora. Pour démarrer, consultez Prérequis des clusters de bases de données.

Pour obtenir des instructions sur la connexion à un cluster de bases de données Aurora, consultez Connexion à un cluster de bases de données Amazon Aurora.

Table des matières

Prérequis des clusters de bases de données

> **Important**
>
> Avant de pouvoir créer un cluster de bases de données Aurora, vous devez effectuer les tâches de la section Configuration de votre environnement pour Amazon Aurora.

Voici les conditions préalables à remplir avant de créer un cluster de bases de données.

Rubriques

Configurer le réseau pour la base de données

Vous pouvez créer un cluster de bases de données Amazon Aurora uniquement dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC, au sein d’une Région AWS qui compte au moins deux zones de disponibilité. Le groupe de sous-réseaux de base de données que vous choisissez pour le cluster de bases de données doit couvrir au moins deux zones de disponibilité. Cette configuration est l’assurance que votre cluster de bases de données dispose toujours d’au moins une instance de base de données pour le basculement, dans le cas improbable d’une défaillance d’une zone de disponibilité.

Si vous prévoyez de configurer la connectivité entre votre nouveau cluster de bases de données et une instance EC2 dans le même VPC, vous pouvez le faire pendant la création du cluster de bases de données. Si vous prévoyez de vous connecter à votre cluster de bases de données à partir de ressources autres que des instances EC2 dans le même VPC, vous pouvez configurer les connexions réseau manuellement.

Rubriques

Configurer la connectivité réseau automatique avec une instance EC2

Lorsque vous créez un cluster de bases de données Aurora, vous pouvez utiliser la AWS Management Console pour configurer la connectivité entre une instance Amazon EC2 et le nouveau cluster DB. Dans ce cas, RDS configure automatiquement votre VPC et vos paramètres réseau. Le cluster de bases de données est créé dans le même VPC que l'instance EC2 afin que cette dernière puisse accéder au cluster de bases de données.

Voici les conditions requises pour connecter une instance EC2 au cluster de bases de données :

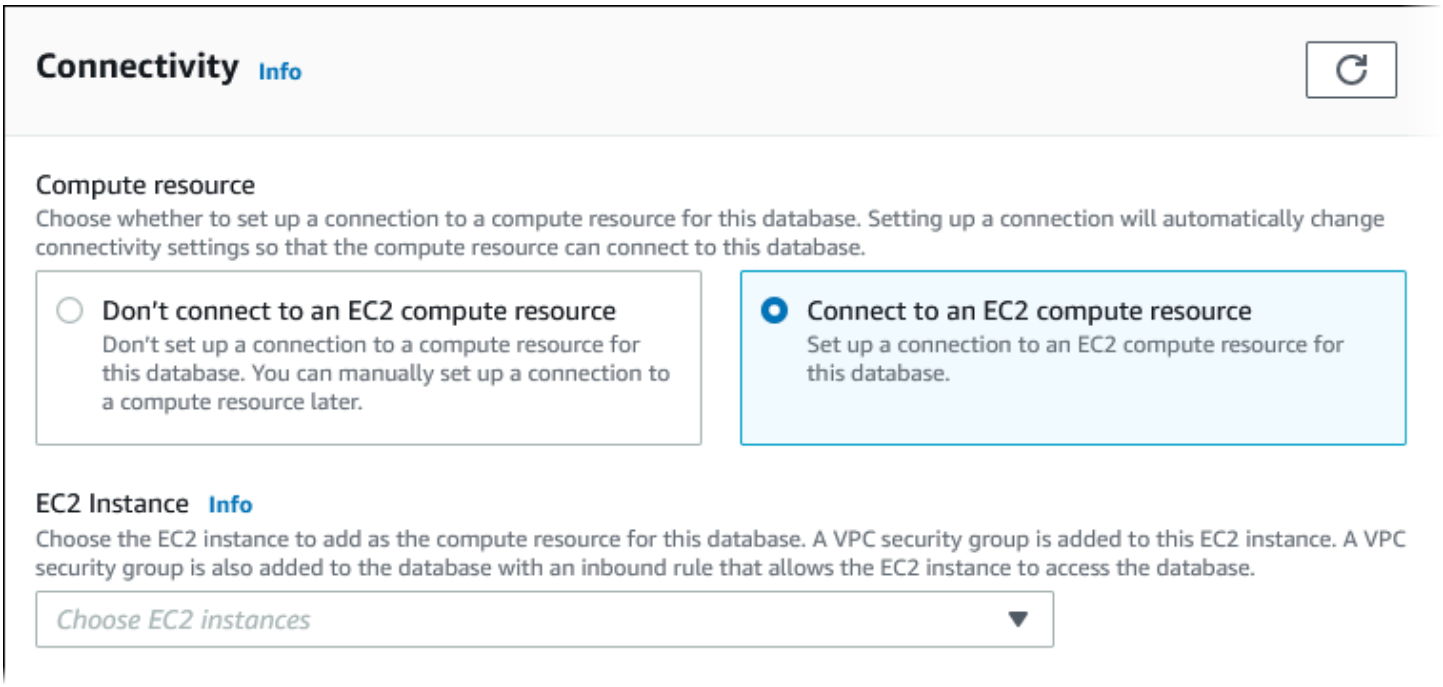
- L'instance EC2 doit exister dans la Région AWS avant que vous ne créiez le cluster de bases de données.

  S'il n'y a pas d'instances EC2 dans la Région AWS, la console fournit un lien pour en créer une.
- Actuellement, le cluster de bases de données ne peut pas être un cluster de bases de données Aurora Serverless ou une partie d'une base de données globale Aurora.
- L'utilisateur qui crée l'instance de base de données doit avoir les autorisations nécessaires pour effectuer les opérations suivantes :
  - `ec2:AssociateRouteTable`
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateRouteTable`
  - `ec2:CreateSubnet`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeRouteTables`
  - `ec2:DescribeSecurityGroups`
  - `ec2:DescribeSubnets`
  - `ec2:ModifyNetworkInterfaceAttribute`

- `ec2:RevokeSecurityGroupEgress`

Cette option permet de créer un cluster de bases de données privé. Le cluster de bases de données utilise un groupe de sous-réseaux de base de données avec uniquement des sous-réseaux privés pour restreindre l'accès aux ressources au sein du VPC.

Pour connecter une instance EC2 au cluster de bases de données, choisissez Connect to an EC2 compute resource (Se connecter à une ressource de calcul EC2) dans la section Connectivity (Connectivité) de la page Create database (Créer une base de données).



Lorsque vous choisissez Connect to an EC2 compute resource (Se connecter à une ressource de calcul EC2), RDS définit automatiquement les options suivantes. Vous ne pouvez pas modifier ces paramètres à moins de choisir de ne pas établir de connectivité avec une instance EC2 en sélectionnant Don't connect to an EC2 compute resource (Ne pas se connecter à une ressource de calcul EC2).

Option console	Réglage automatique
Network type (Type de réseau)	RDS définit le type de réseau comme IPv4. Actuellement, le mode double pile n'est pas pris en charge lorsque vous établissez une connexion entre une instance EC2 et le cluster de bases de données.

Option console	Réglage automatique
Cloud privé virtuel (VPC)	RDS définit le VPC comme celui qui est employé pour l’instance EC2.
Groupe de sous-réseaux de base de données	RDS nécessite un groupe de sous-réseaux de base de données avec un sous-réseau privé dans la même zone de disponibilité que l’instance EC2. Si un groupe de sous-réseau de base de données répondant à cette exigence existe, RDS utilise alors le groupe de sous-réseau de base de données existant. Par défaut, cette option est définie sur Automatic setup (Configur ation automatique). Lorsque vous choisissez Automatic setup (Configuration automatique) et qu’aucun groupe de sous-réseaux de base de données ne répond à cette exigence, l’action suivante se produit. RDS utilise trois sous-réseaux privés disponibles dans trois zones de disponibilité, l’une des zones de disponibilité étant la même que pour l’instance EC2. Si un sous-réseau privé n’est pas disponible dans une zone de disponibilité, RDS crée un sous-réseau privé dans la zone de disponibilité. RDS crée ensuite le groupe de sous-réseau de base de données. Lorsqu’un sous-réseau privé est disponible, RDS utilise la table de routage qui lui est associée avec le sous-réseau et ajoute les sous-réseaux qu’il crée à cette table de routage. Lorsqu’au cun sous-réseau privé n’est disponible, RDS crée une table de routage sans accès à la passerelle Internet et ajoute les sous-réseaux qu’il crée à la table de routage. RDS vous permet également d’utiliser des groupes de sous-réseaux de base de données existants. Sélectionnez Choose existing (Choisir existants) si vous souhaitez utiliser un groupe de sous-réseaux de base de données existant de votre choix.

Option console	Réglage automatique
Accès public	RDS choisit No (Non) pour que le cluster de bases de données ne soit pas publiquement accessible. Pour des raisons de sécurité, il est préférable de garder la base de données privée et de s'assurer qu'elle n'est pas accessible depuis Internet.
VPC security group (firewall) [Groupe de sécurité VPC (pare-feu)]	RDS crée un nouveau groupe de sécurité qui est employé avec le cluster de bases de données. Le groupe de sécurité est nommé `rds-ec2-`*n*, où *n* est un nombre. Ce groupe de sécurité comprend une règle d'entrée avec le groupe de sécurité EC2 VPC (pare-feu) comme source. Ce groupe de sécurité qui est employé avec le cluster de bases de données permet à l'instanc e EC2 d'accéder au cluster de bases de données. RDS crée également un groupe de sécurité qui est employé avec l'instance EC2. Le groupe de sécurité est nommé `ec2-rds-`*n*, où *n* est un nombre. Ce groupe de sécurité comprend une règle de sortie avec le groupe de sécurité VPC du cluster de bases de données comme source. Ce groupe de sécurité permet à l'instance EC2 d'envoyer du trafic au cluster de bases de données. Vous pouvez ajouter un autre groupe de sécurité en sélection nant Create new (Créer nouveau) et en saisissant le nom du nouveau groupe de sécurité. Vous pouvez ajouter des groupes de sécurité existants en choisissant Choose existing (Choisir existant) et en sélection nant les groupes de sécurité à ajouter.

Option console	Réglage automatique
Zone de disponibilité	Lorsque vous ne créez pas de réplica Aurora en Availability & durability (Disponibilité et durabilité) lors de la création du cluster de bases de données (déploiement Mono-AZ), RDS choisit la zone de disponibilité de l'instance EC2. Lorsque vous créez un réplica Aurora pendant la création d'un cluster de bases de données (déploiement Multi-AZ), RDS choisit la zone de disponibilité de l'instance EC2 pour une instance de base de données dans le cluster de bases de données. RDS choisit de manière aléatoire une zone de disponibilité différente pour l'autre instance de base de données dans le cluster de bases de données. L'instance de base de données principale ou le réplica Aurora est créé(e) dans la même zone de disponibilité que l'instance EC2. Vous pourriez être confronté à des coûts croisés entre zones de disponibilité si l'instance de base de données principale et l'instance EC2 se trouvent dans des zones de disponibilité différentes.

Pour plus d'informations sur ces paramètres, consultez la page Paramètres pour les clusters de bases de données Aurora.

Si vous modifiez ces paramètres après la création du cluster de bases de données, ces modifications peuvent affecter la connexion entre l'instance EC2 et le cluster de bases de données.

Configuration manuelle du réseau

Si vous prévoyez de vous connecter à votre cluster de bases de données à partir de ressources autres que des instances EC2 dans le même VPC, vous pouvez configurer les connexions réseau manuellement. Si vous utilisez la AWS Management Console pour créer votre cluster de bases de données, Amazon RDS peut alors créer automatiquement un VPC à votre place. Une autre solution consiste à utiliser un VPC existant ou à créer un VPC pour votre cluster de bases de données Aurora. Quelle que soit l'approche adoptée, votre VPC doit comporter au moins un sous-réseau dans chacune d'au moins deux zones de disponibilité pour que vous puissiez l'utiliser avec un cluster de bases de données Amazon Aurora.

Par défaut, Amazon RDS crée automatiquement pour vous l'instance de base de données principale et le réplica Aurora dans les zones de disponibilité. Pour choisir une zone de disponibilité spécifique, vous devez définir le paramètre de déploiement Multi-AZ Availability & durability (Disponibilité et durabilité) sur Don't create an Aurora Replica (Ne pas créer de réplica Aurora). Ce faisant, vous exposez un paramètre de zone de disponibilité qui vous permet de choisir parmi les zones de disponibilité de votre VPC. Toutefois, nous vous recommandons fortement de conserver le paramètre par défaut et de laisser Amazon RDS créer un déploiement Multi-AZ et choisir les zones de disponibilité pour vous. Ce faisant, votre cluster de bases de données Aurora est créé avec les fonctions de basculement rapide et de haute disponibilité qui sont deux des avantages clé d'Aurora.

Si vous n'avez pas de VPC par défaut ou que vous n'avez créé aucun VPC, Amazon RDS peut en créer un automatiquement lorsque vous créez un cluster de bases de données à partir de la console. Sinon, vous devez exécuter les actions suivantes :

- Créez un VPC doté d'au moins un sous-réseau dans au moins deux zones de disponibilité dans la Région AWS où vous voulez déployer votre cluster de bases de données. Pour plus d'informations, consultez Utilisation d'un cluster de bases de données dans un VPC et Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement).
- Spécifiez un groupe de sécurité VPC qui autorise les connexions à votre cluster de bases de données. Pour plus d'informations, consultez Créer un groupe de sécurité qui autorise l'accès au cluster de bases de données dans le VPC et Contrôle d'accès par groupe de sécurité.
- Spécifiez un groupe de sous-réseaux DB RDS définissant au moins deux sous-réseaux du VPC pouvant être utilisés par le cluster de bases de données . Pour plus d'informations, consultez Utilisation de groupes de sous-réseaux DB.

Pour plus d'informations sur les VPC, consultez Amazon VPC et Amazon Aurora. Pour accéder à un tutoriel qui configure le réseau pour un cluster de bases de données privé, consultez Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement).

Si vous souhaitez vous connecter à une ressource qui ne se trouve pas dans le même VPC que le cluster de bases de données Aurora, consultez les scénarios appropriés dans Scénarios d'accès à un cluster de bases de données d'un VPC.

Prérequis supplémentaires

Avant de créer votre cluster de bases de données, tenez compte des conditions préalables supplémentaires suivantes :

- Si vous vous connectez à AWS à l'aide des informations d'identification Gestion des identités et des accès AWS (IAM), votre compte AWS doit disposer des politiques IAM qui accordent les autorisations requises pour exécuter les opérations Amazon RDS. Pour plus d'informations, consultez Identity and Access Management pour Amazon Aurora.

  Si vous utilisez IAM pour accéder à la console Amazon RDS, vous devez d'abord vous connecter à la AWS Management Console avec vos informations d'identification. Connectez-vous ensuite à la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
- Si vous voulez adapter les paramètres de configuration de votre cluster de bases de données, vous devez spécifier un groupe de paramètres de cluster de bases de données et un groupe de paramètres de base de données avec les valeurs de paramètre requises. Pour plus d'informations sur la création ou la modification d'un groupe de paramètres de cluster de bases de données ou d'un groupe de paramètres de base de données, consultez Groupes de paramètres pour Amazon Aurora.
- Déterminez le numéro de port TCP/IP à spécifier pour le cluster de bases de données. Dans certaines entreprises, les pare-feux bloquent les connexions aux ports par défaut (3306 pour MySQL, 5432 pour PostgreSQL) pour Aurora. Si le pare-feu de votre entreprise bloque le port par défaut, choisissez un autre port pour le cluster de bases de données. Toutes les instances d'un cluster de bases de données utilisent le même port.
- Si la version majeure du moteur de votre base de données a atteint la date de fin de support standard RDS, vous devez utiliser l'option d'interface de ligne de commande Support étendu ou le paramètre API RDS. Pour plus d'informations, consultez Support étendu RDS dans Paramètres pour les clusters de bases de données Aurora.

Création d'un cluster de bases de données

Vous pouvez créer un cluster de bases de données Aurora à partir de l'AWS Management Console, de l'AWS CLI ou de l'API RDS.

Console

Vous pouvez créer une instance de base de données à l'aide de la AWS Management Console, que l'option Création facile soit activée ou désactivée. Lorsque l'option Easy create (Création facile) est activée, vous spécifiez uniquement le type de moteur, la taille de l'instance, ainsi que l'identifiant d'instance de base de données. Easy create (Création facile) utilise les paramètres par défaut pour les autres options de configuration. Lorsque Easy create (Création facile) est désactivé, vous

spécifiez davantage d'options de configuration lors de la création d'une base de données, notamment en matière de disponibilité, de sécurité, de sauvegardes et de maintenance.

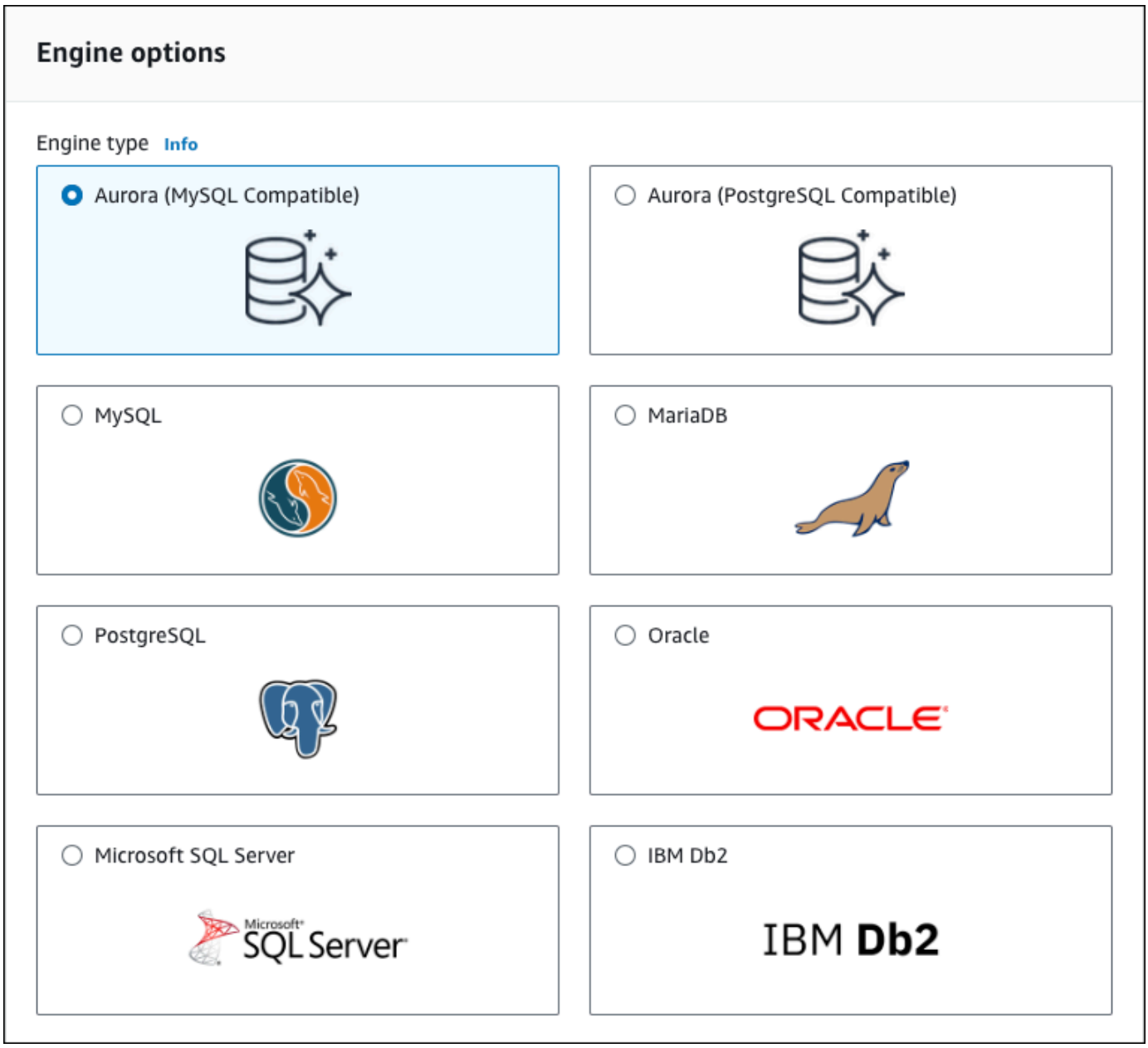
> **Note**
>
> Pour cet exemple, l'option Standard create (Création standard) est activée et Easy create (Création facile) est désactivée. Pour en savoir plus sur la création d'un cluster de bases de données avec l'option Création facile activée, consultez Mise en route avec Amazon Aurora.

Pour créer un cluster de bases de données Aurora à partir de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le coin supérieur droit de la AWS Management Console, sélectionnez la région AWS dans laquelle vous voulez créer le cluster de bases de données.

   Aurora n'est pas disponible dans toutes les régions AWS. Pour obtenir la liste des régions AWS où Aurora est disponible, consultez Disponibilité dans les Régions.
3. Dans la panneau de navigation, choisissez Databases (Bases de données).
4. Choisissez Create database (Créer une base de données).
5. Pour Choisir une méthode de création de base de données, choisissez Création standard.
6. Pour Type de moteur, choisissez l'une des valeurs suivantes :

   - Aurora (compatible avec MySQL)
   - Aurora (compatible avec PostgreSQL)



7. Choisissez la Version du moteur.

   Pour plus d'informations, consultez Versions d'Amazon Aurora. Vous pouvez utiliser les filtres pour choisir les versions compatibles avec les fonctionnalités que vous souhaitez, telles que Aurora Serverless v2. Pour plus d'informations, consultez Utiliser Aurora Serverless v2.

8. Dans Templates (Modèles), sélectionnez le modèle qui correspond à votre cas d'utilisation.

9. Pour entrer votre mot de passe principal, procédez comme suit :

   a. Dans la section Paramètres, développez Paramètres des informations d'identification.

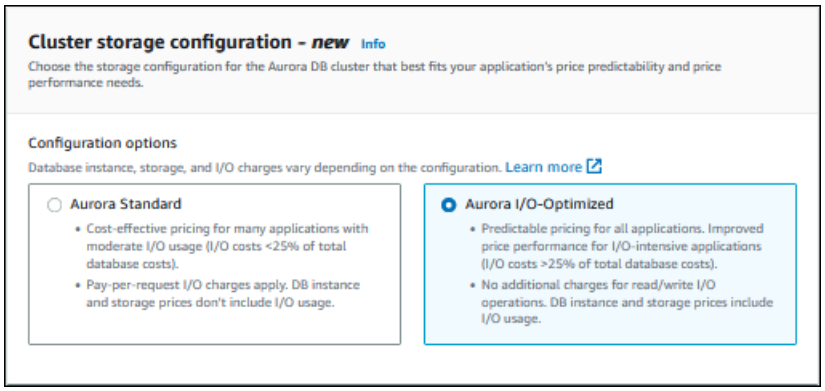
b. Décochez la case Auto generate a password (Générer un mot de passe automatiquement).

c. (Facultatif) Modifiez la valeur du champ Master username (Identifiant principal) et saisissez le même mot de passe dans les champs Master password (Mot de passe principal) et Confirm password (Confirmer le mot de passe).

Par défaut, la nouvelle instance de base de données utilise un mot de passe généré automatiquement pour l'utilisateur principal.

10. Dans la section Connectivité sous Groupe de sécurité VPC (pare-feu), si vous sélectionnez Créer, un groupe de sécurité VPC est créé avec une règle entrante qui autorise l'adresse IP de votre ordinateur local à accéder à la base de données.

11. Pour Configuration du stockage en cluster, choisissez Aurora I/O-Optimized ou Aurora Standard. Pour plus d'informations, consultez Configurations de stockage pour les clusters de bases de données Amazon Aurora.



12. (Facultatif) Configurez une connexion à une ressource de calcul pour ce cluster de bases de données.

Vous pouvez configurer la connectivité entre une instance Amazon EC2 et le nouveau cluster de bases de données pendant la création du cluster de bases de données. Pour plus d'informations, consultez Configurer la connectivité réseau automatique avec une instance EC2.

13. Pour les sections restantes, spécifiez vos paramètres de cluster de bases de données. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour les clusters de bases de données Aurora.

14. Choisissez Create database (Créer une base de données).

Si vous choisissez de générer un mot de passe automatiquement, le bouton Afficher les informations d'identification apparaît sur la page Bases de données.

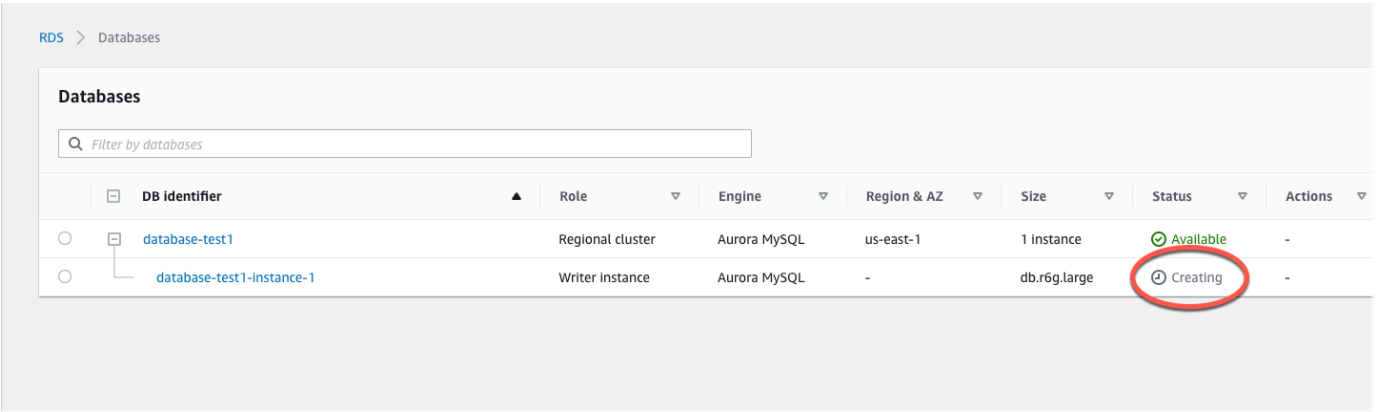
Pour afficher l'identifiant principal et le mot de passe pour le cluster de bases de données, choisissez View credential details (Afficher les informations d'identification).

Pour vous connecter à l'instance de base de données en tant qu'utilisateur principal, utilisez l'identifiant et le mot de passe affichés.

> **Important**
>
> Vous ne pourrez pas afficher le mot de passe de l'utilisateur principal de nouveau. Si vous ne l'enregistrez pas, il sera peut-être nécessaire de le modifier. Si vous devez changer le mot de passe de l'utilisateur principal une fois l'instance de base de données disponible, vous pouvez le faire en modifiant l'instance de base de données. Pour plus d'informations sur la modification d'une instance de base de données, consultez Modification d'un cluster de bases de données Amazon Aurora.

15. Pour Bases de données, choisissez le nom du nouveau cluster de bases de données Aurora.

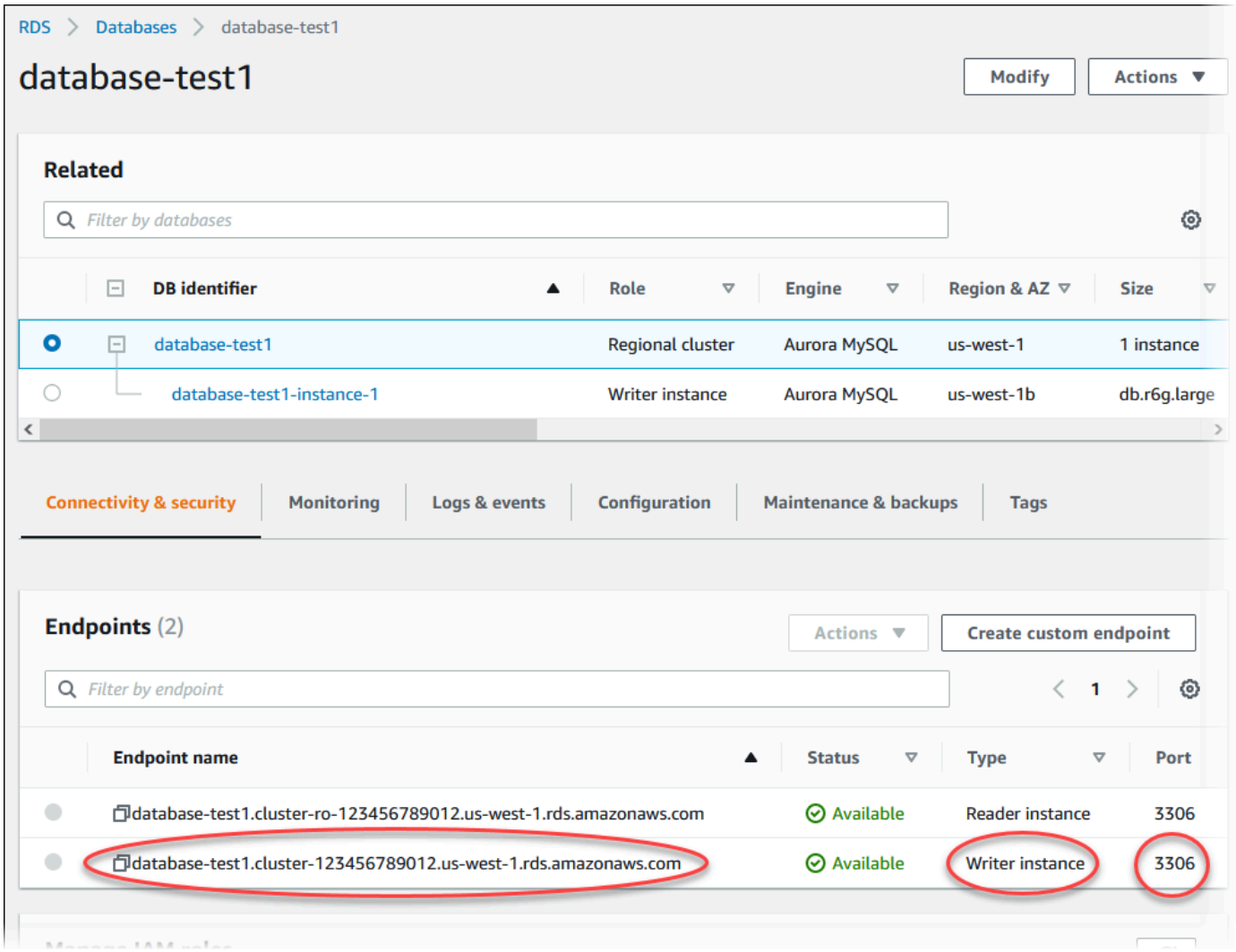
Sur la console RDS, les détails du nouveau cluster de bases de données s'affichent. Le cluster de bases de données et son instance de base de données auront un statut creating (création en cours) jusqu'à ce qu'il soit créé et prêt à l'emploi.



Lorsque l'état devient available (disponible) pour les deux éléments, vous pouvez vous connecter au cluster de bases de données. En fonction de la quantité de stockage et de la classe d'instance de base de données, la mise à disposition du nouveau cluster de bases de données peut prendre jusqu'à 20 minutes.

Pour afficher le cluster nouvellement créé, choisissez Bases de données depuis le panneau de navigation de la console Amazon RDS. Choisissez ensuite le cluster de bases de données pour

en afficher les détails. Pour plus d'informations, consultez Affichage d'un cluster de bases de données Amazon Aurora.



Dans l'onglet Connectivity & security (Connectivité et sécurité), notez le port et le point de terminaison du l'instance de base de données en écriture. Utilisez le point de terminaison et le port du cluster dans vos chaînes de connexion JDBC et ODBC pour toute application qui exécute des opérations de lecture et d'écriture.

AWS CLI

> **Note**
>
> Avant de pouvoir créer un cluster de bases de données Aurora à partir de l'AWS CLI, vous devez remplir des prérequis, comme la création d'un VPC et d'un groupe de sous-réseaux de

base de données RDS. Pour plus d'informations, consultez Prérequis des clusters de bases de données.

Vous pouvez utiliser l'AWS CLI pour créer un cluster de bases de données Aurora MySQL ou Aurora PostgreSQL.

Pour créer un cluster de bases de données Aurora MySQL à partir d'AWS CLI

Lorsque vous créez un cluster de bases de données ou une instance de base de données Aurora compatible MySQL 8.0 ou 5.7, vous spécifiez `aurora-mysql` pour l'option `--engine`.

Procédez comme suit :

1. Identifiez le groupe de sous-réseaux de base de données et l'ID de groupe de sécurité VPC de votre nouveau cluster de bases de données, puis appelez la commande de l'AWS CLI create-db-cluster pour créer le cluster de bases de données Aurora MySQL.

   Par exemple, la commande suivante crée un cluster de bases de données compatible avec MySQL 8.0 nommé `sample-cluster`. Le cluster utilise la version de moteur par défaut et le type de stockage Aurora I/O-Optimized.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster \
       --engine aurora-mysql --engine-version 8.0 \
       --storage-type aurora-iopt1 \
       --master-username user-name --manage-master-user-password \
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Pour Windows :

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
       --engine aurora-mysql --engine-version 8.0 ^
       --storage-type aurora-iopt1 ^
       --master-username user-name --manage-master-user-password ^
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

La commande suivante crée un cluster de bases de données compatible avec MySQL 5.7 nommé `sample-cluster`. Le cluster utilise la version de moteur par défaut et le type de stockage Aurora Standard.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster  \
    --engine aurora-mysql --engine-version 5.7 \
    --storage-type aurora \
    --master-username user-name --manage-master-user-password \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

Pour Windows :

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster sample-cluster  ^
    --engine aurora-mysql --engine-version 5.7 ^
    --storage-type aurora ^
    --master-username user-name --manage-master-user-password ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

2. Si vous utilisez la console pour créer un cluster de bases de données, Amazon RDS crée automatiquement l'instance principale (auteur) pour votre cluster de bases de données. Si vous utilisez l'AWS CLI pour créer un cluster de bases de données, vous devez créer explicitement l'instance principale pour votre cluster de bases de données. L'instance principale est la première instance créée dans un cluster de bases de données. Tant que vous n'avez pas créé l'instance de base de données principale, les points de terminaison du cluster de bases de données conservent le statut `Creating`.

   Appelez la commande create-db-instance de l'AWS CLI pour créer l'instance principale de votre cluster de bases de données. Incluez le nom du cluster de bases de données comme valeur de l'option `--db-cluster-identifier`.

   > **Note**
   >
   > Vous ne pouvez pas définir l'option `--storage-type` pour les instances de base de données. Vous pouvez la définir uniquement pour les clusters de bases de données.

Par exemple, la commande suivante crée une instance de base de données compatible avec MySQL 5.7 ou MySQL 8.0 nommée `sample-instance`.

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance --db-instance-identifier sample-instance \
     --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-
class db.r5.large
```

Pour Windows :

```
aws rds create-db-instance --db-instance-identifier sample-instance ^
     --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-
class db.r5.large
```

Pour créer un cluster de bases de données Aurora PostgreSQL à partir de l'AWS CLI

1. Identifiez le groupe de sous-réseaux de base de données et l'ID de groupe de sécurité VPC de votre nouveau cluster de bases de données, puis appelez la commande de l'AWS CLI create-db-cluster pour créer le cluster de bases de données Aurora PostgreSQL.

   Par exemple, la commande suivante crée un nouveau cluster de bases de données nommé `sample-cluster`. Le cluster utilise la version de moteur par défaut et le type de stockage Aurora I/O-Optimized.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster \
       --engine aurora-postgresql \
       --storage-type aurora-iopt1 \
       --master-username user-name --manage-master-user-password \
       --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
   ```

   Pour Windows :

   ```
   aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
       --engine aurora-postgresql ^
       --storage-type aurora-iopt1 ^
   ```

```
    --master-username user-name --manage-master-user-password ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2
```

2. Si vous utilisez la console pour créer un cluster de bases de données, Amazon RDS crée automatiquement l'instance principale (auteur) pour votre cluster de bases de données. Si vous utilisez l'AWS CLI pour créer un cluster de bases de données, vous devez créer explicitement l'instance principale pour votre cluster de bases de données. L'instance principale est la première instance créée dans un cluster de bases de données. Tant que vous n'avez pas créé l'instance de base de données principale, les points de terminaison du cluster de bases de données conservent le statut `Creating`.

   Appelez la commande create-db-instance de l'AWS CLI pour créer l'instance principale de votre cluster de bases de données. Incluez le nom du cluster de bases de données comme valeur de l'option `--db-cluster-identifier`.

   Pour Linux, macOS ou Unix :

```
aws rds create-db-instance --db-instance-identifier sample-instance \
     --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-
instance-class db.r5.large
```

   Pour Windows :

```
aws rds create-db-instance --db-instance-identifier sample-instance ^
     --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-
instance-class db.r5.large
```

Ces exemples spécifient l'option `--manage-master-user-password` permettant de générer le mot de passe de l'utilisateur principal et de le gérer dans Secrets Manager. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager. Vous pouvez également utiliser l'option `--master-password` pour spécifier et gérer vous-même le mot de passe.

API RDS

> **Note**
>
> Avant de pouvoir créer un cluster de bases de données Aurora à partir de l'AWS CLI, vous devez remplir des prérequis, comme la création d'un VPC et d'un groupe de sous-réseaux de

base de données RDS. Pour plus d'informations, consultez Prérequis des clusters de bases de données.

Identifiez le groupe de sous-réseaux de base de données et l'ID de groupe de sécurité VPC de votre nouveau cluster de bases de données, puis appelez l'opération CreateDBCluster pour créer le cluster de bases de données.

Lorsque vous créez un cluster de bases de données ou une instance de base de données Aurora MySQL version 2 ou 3, spécifiez `aurora-mysql` pour le paramètre `Engine`.

Lorsque vous créez un cluster de bases de données ou une instance de base de données Aurora PostgreSQL, spécifiez `aurora-postgresql` pour le paramètre `Engine`.

Si vous utilisez la console pour créer un cluster de bases de données, Amazon RDS crée automatiquement l'instance principale (auteur) pour votre cluster de bases de données. Si vous utilisez l'API RDS pour créer un cluster de bases de données, vous devez créer explicitement l'instance principale pour votre cluster de bases de données avec CreateDBInstance. L'instance principale est la première instance créée dans un cluster de bases de données. Tant que vous n'avez pas créé l'instance de base de données principale, les points de terminaison du cluster de bases de données conservent le statut `Creating`.

Création d'une instance de base de données principale (enregistreur)

Si vous utilisez la AWS Management Console pour créer un cluster de bases de données, Amazon RDS crée automatiquement l'instance principale (enregistreur) pour votre cluster de bases de données. Si vous utilisez l'AWS CLI ou l'API RDS pour créer un cluster de bases de données, vous devez créer explicitement l'instance principale pour votre cluster de bases de données. L'instance principale est la première instance créée dans un cluster de bases de données. Tant que vous n'avez pas créé l'instance de base de données principale, les points de terminaison du cluster de bases de données conservent le statut `Creating`.

Pour plus d'informations, consultez Création d'un cluster de bases de données.

> **Note**
>
> Si vous avez un cluster de bases de données sans instance de base de données d'enregistreur, également appelé cluster sans périphériques, vous ne pouvez pas utiliser la console pour créer une instance d'enregistreur. Vous devez utiliser l'AWS CLI ou l'API RDS.

L'exemple suivant utilise la commande create-db-instance de l'AWS CLI pour créer une instance d'enregistreur pour un cluster de bases de données Aurora PostgreSQL nommé `headless-test`.

```
aws rds create-db-instance \
    --db-instance-identifier no-longer-headless \
    --db-cluster-identifier headless-test \
    --engine aurora-postgresql \
    --db-instance-class db.t4g.medium
```

Paramètres pour les clusters de bases de données Aurora

Le tableau suivant contient des détails sur les paramètres que vous choisissez lors de la création d'un cluster de bases de données Aurora.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Mise à niveau automatique de versions mineures	Choisissez Enable auto minor version upgrade (Activer la mise à niveau automatique de versions mineures) si vous souhaitez que votre cluster de bases de données Aurora reçoive automatiquement les mises à niveau des versions mineures préférées du moteur de base de données dès qu'elles deviennent disponibles. Le paramètre Auto minor version upgrade (Mise à niveau automatique des versions mineures) s'applique aux clusters de bases de données Aurora PostgreSQL et Aurora MySQL. Pour plus d'informations sur les mises à jour de moteur pour Aurora PostgreSQL, consultez	Définissez cette valeur pour chaque instance de base de données de votre cluster Aurora. Si ce paramètre est désactivé dans une instance de base de données de votre cluster, le cluster n'est pas automatiquement mis à niveau. À partir de l'AWS CLI, exécutez `create-db-instance` et définissez l'option `--auto-minor-version-upgrade|--no-auto-minor-version-upgrade`. À partir de l'API RDS, appelez `CreateDBInstance` et définissez le paramètre `AutoMinorVersionUpgrade`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
	Mises à jour du moteur de base de données pour Amazon Aurora PostgreSQL. Pour plus d'informations sur les mises à jour de moteur pour Aurora MySQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL.	
AWS KMS key	Disponible uniquement si l'option Chiffrement est définie sur Activer le chiffrement. Choisissez la AWS KMS key à utiliser pour le chiffrement de ce cluster de bases de données. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--kms-key-id`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `KmsKeyId`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Retour sur trace	S'applique uniquement à Aurora MySQL. Choisissez Enable Backtrack (Activer le retour sur trace) pour activer le retour sur trace ou Disable Backtrack (Désactiver le retour sur trace) pour le désactiver. Le retour en arrière vous permet de ramener un cluster de bases de données à un point dans le temps spécifique sans créer de nouveau cluster de bases de données. Ce paramètre est désactivé par défaut. Si vous activez le retour en arrière, spécifiez également la période de temps pendant laquelle vous souhaitez pouvoir effectuer un retour en arrière de votre cluster de bases de données (fenêtre de retour en arrière cible). Pour plus d'informations, consultez Retour en arrière d'un cluster de bases de données Aurora.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--backtrack-window` . À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `BacktrackWindow` .
Autorité de certification	L'autorité de certification (CA) pour le certificat de serveur utilisé par les instances de base de données dans le cluster de bases de données. Pour plus d'informations, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.	À partir de l'AWS CLI, exécutez `create-db-instance` et définissez l'option `--ca-certificate-identifier` . À partir de l'API RDS, appelez `CreateDBInstance` et définissez le paramètre `CACertificateIdentifier` .

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Configuration du stockage du cluster	Type de stockage pour le cluster de bases de données : Aurora I/O-Optimized ou Aurora Standard. Pour plus d'informations, consultez Configurations de stockage pour les clusters de bases de données Amazon Aurora.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--storage-type`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `StorageType` .
Copier les balises aux instantanés	Choisissez cette option pour copier toutes les balises de l'instanc e de base de données dans un instantané de base de données lors de la création d'un instantané. Pour plus d'informations, consultez Marquage des ressources Amazon Aurora et Amazon RDS.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--copy-ta gs-to-snapshot | --no-copy-tags-to-snapshot` . À partir de l'API RDS, appelez `CreateDBCluster` et définisse z le paramètre `CopyTagsT oSnapshot` .

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Authentification de base de données	L'authentification de base de données que vous souhaitez utiliser. Pour MySQL : • Choisissez Authentification par mot de passe pour authentifier les utilisateurs de base de données avec des mots de passe de base de données uniquement. • Choisissez Mot de passe et authentification de base de données IAM pour authentifier les utilisateurs de bases de données avec des mots de passe de bases de données et des informations d'identification utilisateur via des utilisateurs et rôles IAM. Pour plus d'informations, consultez Authentification de base de données IAM. Pour PostgreSQL : • Choisissez IAM database authentication (Authentification de base de données IAM) pour authentifier les utilisateurs de bases de données avec des mots de passe de bases de données et des informations d'identification utilisateur via	Pour utiliser l'authentification de base de données IAM avec l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--enable-iam-database-authentication | --no-enable-iam-database-authentication` . Pour utiliser l'authentification de base de données IAM avec l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `EnableIAMDatabaseAuthentication` . Pour utiliser l'authentification Kerberos avec l'AWS CLI, exécutez `create-db-cluster` et définissez les options `--domain` et `--domain-iam-role-name` . Pour utiliser l'authentification Kerberos avec l'API RDS, appelez `CreateDBCluster` et définissez les paramètres `Domain` et `DomainIAMRoleName` .

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
	des utilisateurs et rôles. Pour plus d'informations, consultez Authentification de base de données IAM. • Choisissez Authentification Kerberos pour authentifier les mots de passe de bases de données et les informations d'identification utilisateur à l'aide de l'authentification Kerberos. Pour plus d'informations, consultez Utilisation de l'authentification Kerberos avec Aurora PostgreSQL.	
Port de la base de données	Spécifiez le port utilisé par les applications et les utilitaires pour accéder à la base de données. Les clusters de bases de données Aurora MySQL accèdent par défaut au port MySQL par défaut, 3306, et les clusters de bases de données Aurora PostgreSQL accèdent par défaut au port PostgreSQL par défaut, 5432. Dans certaines entreprises, les pare-feux bloquent les connexions à ces ports par défaut. Si le pare-feu de votre entreprise bloque le port par défaut, choisissez un autre port pour le nouveau cluster DB.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--port`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `Port`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Identifiant du cluster de bases de données	Entrez un nom pour votre cluster de bases de données qui est unique pour votre compte dans la région AWS que vous avez choisie. Cet identifiant est utilisé dans l'adresse de point de terminaison de votre cluster de bases de données. Pour plus d'informations sur le point de terminaison de cluster, consultez Connexions de point de terminaison Amazon Aurora. L'identifiant de cluster de bases de données obéit aux contraintes suivantes : • Il doit contenir entre 1 et 63 caractères alphanumériques ou traits d'union. • Son premier caractère doit être une lettre. • Il ne peut pas se terminer par un trait d'union ou contenir deux traits d'union consécutifs. • Il doit être unique pour tous les clusters de bases de données par compte AWS et par région AWS.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--db-cluster-identifier`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `DBClusterIdentifier`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Groupe de paramètres de cluster de bases de données	Choisissez un groupe de paramètres de cluster de bases de données. Aurora possède un groupe de paramètres de cluster de bases de données par défaut que vous pouvez utiliser, ou bien vous pouvez créer votre propre groupe de paramètres de cluster de bases de données. Pour plus d'informations sur les groupes de paramètres de cluster DB, consultez Groupes de paramètres pour Amazon Aurora.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--db-cluster-parameter-group-name`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `DBClusterParameterGroupName`.
Classe d'instance de base de données	Concerne uniquement le type de capacité allouée. Choisissez une classe d'instance de base de données qui définit les exigences de mémoire et de traitement pour chaque instance du cluster de bases de données. Pour plus d'informations sur les classes d'instance de base de données, consultez Classes d'instance de base de données Amazon Aurora.	Définissez cette valeur pour chaque instance de base de données de votre cluster Aurora. À partir de l'AWS CLI, exécutez `create-db-instance` et définissez l'option `--db-instance-class`. À partir de l'API RDS, appelez `CreateDBInstance` et définissez le paramètre `DBInstanceClass`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Groupe de paramètres de base de données	Choisissez un groupe de paramètres. Aurora possède un groupe de paramètres par défaut que vous pouvez utiliser, ou bien vous pouvez créer votre propre groupe de paramètres. Pour plus d'informations sur les groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora.	Définissez cette valeur pour chaque instance de base de données de votre cluster Aurora. À partir de l'AWS CLI, exécutez `create-db-instance` et définissez l'option `--db-parameter-group-name` . À partir de l'API RDS, appelez `CreateDBInstance` et définissez le paramètre `DBParameterGroupName` .

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Groupe de sous-réseaux de base de données	Le groupe de sous-réseaux de base de données à utiliser pour le cluster de bases de données. Sélectionnez Choose existing (Choisir existants) pour utiliser un groupe de sous-réseaux de base de données. Choisissez ensuite le groupe de sous-réseaux requis dans la liste déroulante Existing DB subnet groups (Groupes de sous-réseaux de base de données existants). Choisissez Automatic setup (Configuration automatique) pour permettre à RDS de sélectionner un groupe de sous-réseaux de base de données compatible. S'il n'en existe aucun, RDS crée un nouveau groupe de sous-réseaux pour votre cluster. Pour plus d'informations, consultez Prérequis des clusters de bases de données.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--db-subnet-group-name` . À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `DBSubnetGroupName` .

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Enable deletion protection (Activer la protection contre la suppression)	Sélectionnez Enable deletion protection (Activer la protection contre la suppression) pour empêcher la suppression de votre cluster de bases de données. Si vous créez un cluster de bases de données de production avec la console, la protection de la suppression est activée par défaut.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--deletion-protection | --no-deletion-protection`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `DeletionProtection`.
Activer le chiffrement	Choisissez `Enable encryption` si vous souhaitez activer le chiffrement au repos pour ce cluster de bases de données. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--storage-encrypted | --no-storage-encrypted`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `StorageEncrypted`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Activer la surveillance améliorée	Choisissez Activer la surveillance améliorée pour activer la collecte de métriques en temps réel pour le système d'exploitation sur lequel votre cluster de bases de données s'exécute. Pour plus d'informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.	Définissez ces valeurs pour chaque instance de base de données de votre cluster Aurora. À partir de l'AWS CLI, exécutez `create-db-instance` et définissez les options `--monitoring-interval` et `--monitoring-role-arn` . À partir de l'API RDS, appelez `CreateDBInstance` et définissez les paramètres `MonitoringInterval` et `MonitoringRoleArn` .
Activer l'API de données RDS	Choisissez Activer l'API de données RDS pour activer l'API de données RDS (API de données). L'API de données fournit un point de terminaison HTTP sécurisé pour exécuter des instructions SQL sans avoir à gérer de connexions. Pour plus d'informations, consultez Utilisation de l'API de données Amazon RDS.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--enable-http-endpoint | --no-enable-http-endpoint` . À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `EnableHttpEndpoint` .
Type de moteur	Choisissez le nom du moteur de base de données à utiliser pour ce cluster de bases de données.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--engine`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `Engine`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Version du moteur	Concerne uniquement le type de capacité allouée. Choisissez le numéro de version de votre moteur de base de données.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--engine-version`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `EngineVersion`.
Priorité de basculement	Choisissez une priorité de basculement pour l'instance. Si vous ne choisissez pas de valeur, la valeur par défaut est tier-1. Cette priorité détermine l'ordre dans lequel les réplicas Aurora sont promus lors de la reprise après une défaillance de l'instance principale. Pour plus d'informations, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.	Définissez cette valeur pour chaque instance de base de données de votre cluster Aurora. À partir de l'AWS CLI, exécutez `create-db-instance` et définissez l'option `--promotion-tier`. À partir de l'API RDS, appelez `CreateDBInstance` et définissez le paramètre `PromotionTier`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Nom de la base de données initiale	Entrez un nom pour votre base de données par défaut. Si vous ne fournissez pas de nom pour le cluster de bases de données Aurora MySQL que vous créez, Amazon RDS ne crée pas de base de données dans ce cluster. Si vous ne fournissez pas de nom pour un cluster de bases de données Aurora PostgreSQL, Amazon RDS crée une base de données nommée `postgres`. Pour Aurora MySQL, le nom de base de données par défaut doit respecter les contraintes suivantes : • Il doit contenir entre 1 et 64 caractères alphanumériques. • Il ne peut pas être un mot réservé par le moteur de base de données. Pour Aurora PostgreSQL, le nom de base de données par défaut doit respecter les contraintes suivantes : • Il doit contenir entre 1 et 63 caractères alphanumériques. • Il doit commencer par une lettre. Les caractères suivants peuvent être des lettres, des traits de	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--database-name`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `DatabaseName` .

Paramètre de la console	Description du paramètre	Option de l’interface CLI et paramètre de l’API RDS
	soulignement ou des chiffres (0 à 9). • Il ne peut pas être un mot réservé par le moteur de base de données. Pour créer des bases de données supplémentaires, connectez-vous au cluster de bases de données et utilisez la commande SQL CREATE DATABASE. Pour plus d’informations sur la connexion au cluster de bases de données, consultez Connexion à un cluster de bases de données Amazon Aurora.	

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Exportations des journaux	Dans la section Log exports (Exportations des journaux), choisissez les journaux que vous voulez commencer à publier dans Amazon CloudWatch Logs. Pour plus d'informations sur la publication des journaux Aurora MySQL vers CloudWatch Logs, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs. Pour plus d'informations sur la publication des journaux Aurora PostgreSQL vers CloudWatch Logs, consultez Publication de journaux Aurora PostgreSQL sur Amazon CloudWatch Logs.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--enable-cloudwatch-logs-exports`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `EnableCloudwatchLogsExports`.
Fenêtre de maintenance	Choisissez Sélectionner la fenêtre et spécifiez la plage de temps hebdomadaire au cours de laquelle la maintenance peut avoir lieu. Vous pouvez également choisir No preference (Aucune préférence) afin qu'Amazon RDS affecte une période de manière aléatoire.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--preferred-maintenance-window`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `PreferredMaintenanceWindow`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Gérer les informations d'identification principales dans AWS Secrets Manager	Sélectionnez Gérer les informations d'identification principales dans AWS Secrets Manager pour gérer le mot de passe d'utilisateur principal dans un secret, dans Secrets Manager. Vous pouvez éventuellement choisir une clé KMS à utiliser pour protéger le secret. Choisissez l'une des clés KMS de votre compte ou entrez la clé d'un autre compte. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez les options `--manage-master-user-password | --no-manage-master-user-password` et `--master-user-secret-kms-key-id`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez les paramètres `ManageMasterUserPassword` et `MasterUserSecretKmsKeyId`.
Mot de passe principal	Entrez un mot de passe pour vous connecter à votre cluster de bases de données : • Pour Aurora MySQL, le mot de passe doit contenir entre 8 et 41 caractères ASCII imprimables. • Pour Aurora PostgreSQL, il doit contenir entre 8 et 99 caractères ASCII imprimables. • Il ne peut pas contenir `/`, `"`, `@` ou un espace.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--master-user-password`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `MasterUserPassword`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Identifiant principal	Entrez un nom à utiliser en tant que nom d'utilisateur principal pour vous connecter à votre cluster de bases de données. • Pour Aurora MySQL, le nom doit contenir entre 1 et 16 caractères alphanumériques. • Pour Aurora PostgreSQL, il doit contenir entre 1 et 63 caractères alphanumériques. • Le premier caractère doit être une lettre. • Le nom ne peut pas être un mot réservé par le moteur de base de données. Vous ne pouvez pas modifier le nom de l'utilisateur principal après la création du cluster de bases de données.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--master-username` . À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `MasterUsername` .

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
déploiement multi-AZ	Concerne uniquement le type de capacité allouée. Détermine z si vous souhaitez créer des réplicas Aurora dans d'autres zones de disponibilité pour la prise en charge du basculement. Si vous choisissez Créer un réplica dans une autre zone, Amazon RDS crée automatiquement un réplica Aurora dans votre cluster de bases de données dans une zone de disponibilité différente de celle de l'instance principale de votre cluster de bases de données. Pour plus d'informations sur les zones de disponibilité multiples , consultez Régions et zones de disponibilité.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--availability-zones` . À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `AvailabilityZones` .

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Network type (Type de réseau)	Les protocoles d'adressage IP pris en charge par le cluster de la base de données. IPv4 pour spécifier que les ressources peuvent communiquer avec le cluster de bases de données uniquement via le protocole d'adressage IPv4. Dual-stack mode (Mode double pile) pour spécifier que les ressources peuvent communiquer avec le cluster de bases de données sur IPv4, IPv6, ou les deux. Utilisez le mode double pile si vous possédez des ressources qui doivent communiquer avec votre cluster de bases de données via le protocole d'adressage IPv6. Pour utiliser le mode double pile, assurez-vous qu'au moins deux sous-réseaux couvrant deux zones de disponibilité prennent en charge le protocole réseau IPv4 et IPv6. Veillez également à associer un bloc CIDR IPv6 aux sous-réseaux du groupe de sous-réseaux de base de données que vous spécifiez. Pour plus d'informations, consultez Adressage IP Amazon Aurora.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--network-type`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `NetworkType`.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Accès public	Choisissez Accessible publiquement pour donner au cluster de bases de données une adresse IP publique, ou choisissez Non accessible publiquement. Les instances de votre cluster de bases de données peuvent être un mélange d'instances de base de données publiques et privées. Pour plus d'informations sur le masquage des instances de l'accès public, consultez Masquer un cluster de bases de données dans un VPC depuis Internet. Pour se connecter à une instance de base de données hors de son Amazon VPC, l'instance de base de données doit être accessible au public, l'accès doit être accordé en utilisant les règles entrantes du groupe de sécurité de l'instance de base de données et d'autres exigences doivent être respectées. Pour plus d'informations, consultez Impossible de se connecter à l'instance de base de données Amazon RDS. Si votre instance de base de données n'est pas accessible publiquement, vous pouvez également utiliser une connexion AWS Site-to-Site VPN ou une	Définissez cette valeur pour chaque instance de base de données de votre cluster Aurora. À partir de l'AWS CLI, exécutez `create-db-instance` et définissez l'option `--publicly-accessible | --no-publicly-accessible` . À partir de l'API RDS, appelez `CreateDBInstance` et définissez le paramètre `PubliclyAccessible` .

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
	connexion Direct Connect pour y accéder à partir d'un réseau privé. Pour plus d'informations, consultez Confidentialité du trafic inter-réseau.	
Support étendu RDS	Sélectionnez Activer le support étendu RDS pour permettre aux versions majeures du moteur prises en charge de continuer à fonctionner après la date de fin du support standard Aurora. Lorsque vous créez un cluster de bases de données, Amazon Aurora utilise par défaut le support étendu RDS. Pour empêcher la création d'un nouveau cluster de bases de données après la date de fin de support standard Aurora et pour éviter les frais liés au support étendu RDS, désactivez ce paramètre. Vos clusters de bases de données existants ne seront pas facturés avant la date de début de la tarification du support étendu RDS. Pour plus d'informations, consultez Support étendu Amazon RDS avec Amazon Aurora.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--engine-lifecycle-support` . À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `EngineLifecycleSupport` .

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
RDS Proxy (Proxy RDS)	Sélectionnez Create an RDS Proxy (Créer un proxy RDS) pour créer un proxy pour votre cluster de bases de données. Amazon RDS crée automatiq uement un rôle IAM et un secret Secrets Manager pour le proxy. Pour plus d'informations, consultez Proxy Amazon RDS pour Aurora.	Non disponible lors de la création d'un cluster de bases de données.
Période de conservat ion	Sélectionnez la durée, comprise entre 1 et 35 jours, pendant laquelle Aurora conserve les copies de sauvegarde de la base de données. Les copies de sauvegarde peuvent être utilisées pour les restaurations à un instant dans le passé de votre base de données à la seconde.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--backup-retention-period` . À partir de l'API RDS, appelez `CreateDBCluster` et définisse z le paramètre `BackupRet entionPeriod` .
Turn on DevOps Guru (Activer DevOps Guru)	Choisissez Turn on DevOps Guru (Activer DevOps Guru) pour activer Amazon DevOps Guru pour votre base de données Aurora. Pour que DevOps Guru for RDS fournisse une analyse détaillée des anomalies de performan ces, Performance Insights doit être activé. Pour plus d'informa tions, consultez Configuration de DevOps Guru pour RDS.	Vous pouvez activer DevOps Guru for RDS à partir de la console RDS, mais vous ne pouvez pas le faire à l'aide de l'API RDS ou de l'interface de ligne de commande. Pour plus d'informations sur l'activation de DevOps Guru, consultez le Guide de l'utilisateur Amazon DevOps Guru.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Activer Performance Insights	Choisissez Turn on Performance Insights (Activer Performance Insights) pour activer Amazon RDS Performance Insights. Pour plus d'informations, consultez Surveillance de la charge de la base de données avec Performance Insights sur .	Définissez ces valeurs pour chaque instance de base de données de votre cluster Aurora. À l'aide de l'AWS CLI, exécutez `create-db-instance` et définissez les options `--enable-performance-insights | --no-enable-performance-insights`, `--performance-insights-kms-key-id` et `--performance-insights-retention-period` . À l'aide de l'API RDS, appelez `CreateDBInstance` et définissez les paramètres `EnablePerformanceInsights`, `PerformanceInsightsKMSKeyId` et `PerformanceInsightsRetentionPeriod` .
Cloud privé virtuel (VPC)	Choisissez le VPC pour héberger le cluster de bases de données. Choisissez Create a New VPC (Créer un nouveau VPC) pour qu'Amazon RDS crée un VPC pour vous. Pour plus d'informations, consultez Prérequis des clusters de bases de données.	Pour la AWS CLI et l'API, vous spécifiez les ID de groupe de sécurité VPC.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Groupe de sécurité VPC (pare-feu)	Choisissez Create new (Créer) pour qu'Amazon RDS crée un groupe de sécurité VPC automatiquement. Vous pouvez également choisir Choose existing (Choisir existant) et spécifier un ou plusieurs groupes de sécurité VPC pour sécuriser l'accès réseau au cluster de bases de données. Lorsque vous choisissez Create new (Créer) dans la console RDS, un groupe de sécurité est créé avec une règle de trafic entrant qui autorise l'accès à l'instance de base de données à partir de l'adresse IP détectée dans votre navigateur. Pour plus d'informations, consultez Prérequis des clusters de bases de données.	À partir de l'AWS CLI, exécutez `create-db-cluster` et définissez l'option `--vpc-security-group-ids`. À partir de l'API RDS, appelez `CreateDBCluster` et définissez le paramètre `VpcSecurityGroupIds`.

Paramètres non applicables aux clusters de bases de données Amazon Aurora

Les paramètres suivants dans la commande `create-db-cluster` de l'AWS CLI et l'opération de l'API RDS `CreateDBCluster` ne s'appliquent pas aux clusters de bases de données Amazon Aurora.

> **Note**
>
> L'AWS Management Console n'affiche pas ces paramètres pour les clusters de bases de données Aurora.

Paramètre AWS CLI	Paramètre de l'API RDS
`--allocated-storage`	`AllocatedStorage`
`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`	`AutoMinorVersionUpgrade`
`--db-cluster-instance-class`	`DBClusterInstanceClass`
`--enable-performance-insights | --no-enable-performance-insights`	`EnablePerformanceInsights`
`--iops`	`Iops`
`--monitoring-interval`	`MonitoringInterval`
`--monitoring-role-arn`	`MonitoringRoleArn`
`--option-group-name`	`OptionGroupName`
`--performance-insights-kms-key-id`	`PerformanceInsightsKMSKeyId`
`--performance-insights-retention-period`	`PerformanceInsightsRetentionPeriod`
`--publicly-accessible | --no-publicly-accessible`	`PubliclyAccessible`

Paramètres non applicables aux instances de base de données Amazon Aurora

Les paramètres suivants dans la commande `create-db-instance` de l'AWS CLI et l'opération de l'API RDS `CreateDBInstance` ne s'appliquent pas aux instances de base de données du cluster de bases de données Amazon Aurora.

> **Note**
>
> L'AWS Management Console n'affiche pas ces paramètres pour les instances de base de données Aurora.

Paramètre AWS CLI	Paramètre de l'API RDS
`--allocated-storage`	`AllocatedStorage`
`--availability-zone`	`AvailabilityZone`
`--backup-retention-period`	`BackupRetentionPeriod`
`--backup-target`	`BackupTarget`
`--character-set-name`	`CharacterSetName`
`--character-set-name`	`CharacterSetName`
`--custom-iam-instance-profile`	`CustomIamInstanceProfile`
`--db-security-groups`	`DBSecurityGroups`
`--deletion-protection | --no-deletion-protection`	`DeletionProtection`
`--domain`	`Domain`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--enable-cloudwatch-logs-exports`	`EnableCloudwatchLogsExports`

Paramètre AWS CLI	Paramètre de l'API RDS
`--enable-customer-owned-ip | --no-enable-customer-owned-ip`	`EnableCustomerOwnedIp`
`--enable-iam-database-authentication | --no-enable-iam-database-authentication`	`EnableIAMDatabaseAuthentication`
`--engine-version`	`EngineVersion`
`--iops`	`Iops`
`--kms-key-id`	`KmsKeyId`
`--master-username`	`MasterUsername`
`--master-user-password`	`MasterUserPassword`
`--max-allocated-storage`	`MaxAllocatedStorage`
`--multi-az | --no-multi-az`	`MultiAZ`
`--nchar-character-set-name`	`NcharCharacterSetName`
`--network-type`	`NetworkType`
`--option-group-name`	`OptionGroupName`
`--preferred-backup-window`	`PreferredBackupWindow`
`--processor-features`	`ProcessorFeatures`
`--storage-encrypted | --no-storage-encrypted`	`StorageEncrypted`
`--storage-type`	`StorageType`
`--tde-credential-arn`	`TdeCredentialArn`
`--tde-credential-password`	`TdeCredentialPassword`

Paramètre AWS CLI	Paramètre de l’API RDS
`--timezone`	`Timezone`
`--vpc-security-group-ids`	`VpcSecurityGroupIds`

Création de ressources Amazon Aurora avec AWS CloudFormation

Amazon Aurora est intégré avec AWS CloudFormation, un service qui vous aide à modéliser et à configurer vos ressources AWS pour vous permettre de consacrer moins de temps à la création et à la gestion de vos ressources et de votre infrastructure. Vous créez un modèle qui décrit toutes les ressources AWS souhaitées (comme les clusters de bases de données et les groupes de paramètres de cluster de bases de données), et CloudFormation met en service et configure ces ressources pour vous.

Lorsque vous utilisez CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos ressources Aurora de manière cohérente et répétée. Décrivez vos ressources une seule fois, puis mettez-le en service autant de fois que vous le souhaitez dans plusieurs comptes et régions AWS.

Aurora et modèles CloudFormation

Les modèles CloudFormation sont des fichiers texte au format JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez allouer dans vos piles CloudFormation. Si JSON ou YAML ne vous est pas familier, vous pouvez utiliser CloudFormation Designer pour vous aider à démarrer avec des modèles CloudFormation. Pour plus d'informations, consultez Qu'est-ce que CloudFormation Designer dans le Guide de l'utilisateur AWS CloudFormation.

Aurora prend en charge la création de ressources dans CloudFormation. Pour plus d'informations, y compris des exemples de modèles JSON et YAML pour ces ressources, consultez la Référence de type de ressource RDS dans le Guide de l'utilisateur AWS CloudFormation.

En savoir plus sur CloudFormation

Pour en savoir plus sur CloudFormation, consultez les ressources suivantes :

- AWS CloudFormation
- Guide de l'utilisateur AWS CloudFormation
- Référence d'API CloudFormation
- Guide de l'utilisateur de l'interface de ligne de commande AWS CloudFormation

Connexion à un cluster de bases de données Amazon Aurora

Vous pouvez vous connecter à un cluster de bases de données Aurora à l'aide des mêmes outils que ceux que vous utilisez pour vous connecter à une base de données MySQL ou PostgreSQL. Vous spécifiez une chaîne de connexion avec n'importe quel script, utilitaire ou application qui se connecte à une instance de base de données MySQL ou PostgreSQL. Vous utilisez la même clé publique que celle utilisée pour les connexions SSL (Secure Sockets Layer).

Dans la chaîne de connexion, vous utilisez généralement les informations sur le port et l'hôte depuis des points de terminaison spéciaux associés au cluster de bases de données. Avec ces points de terminaison, vous pouvez utiliser les mêmes paramètres de connexion, peu importe le nombre d'instances de base de données dans le cluster. Pour les tâches spécialisées telles que le dépannage, vous pouvez utiliser les informations sur l'hôte et le port depuis une instance de base de données spécifique dans votre cluster de bases de données Aurora.

> **Note**
>
> Pour les clusters de bases de données Aurora Serverless, vous vous connectez au point de terminaison de base de données plutôt qu'à l'instance de base de données. Vous pouvez trouver le point de terminaison de base de données pour un cluster Aurora Serverless dans l'onglet Connectivité et sécurité de la AWS Management Console. Pour de plus amples informations, veuillez consulter Utilisation d'Amazon Aurora Serverless v1.

Quel que soit le moteur de base de données Aurora et les outils spécifiques que vous utilisez pour travailler avec le cluster ou l'instance, le point de terminaison doit être accessible. Un cluster de bases de données Aurora ne peut être créé que dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC. Cela signifie que vous accédez au point de terminaison depuis l'intérieur ou depuis l'extérieur du VPC en utilisant l'une des approches suivantes.

- Accéder au cluster de bases de données Aurora à l'intérieur du VPC : activez l'accès au cluster de bases de données Aurora via le VPC. Pour ce faire, modifiez les règles entrantes sur le groupe Sécurité du VPC afin d'autoriser l'accès à votre cluster de bases de données Aurora spécifique. Pour en savoir plus, notamment comment configurer votre VPC pour différents scénarios de cluster de base de données Aurora, consultez Amazon Virtual Private Cloud VPCs et Amazon Aurora.
- Accéder au cluster de bases de données Aurora en dehors du VPC : pour accéder à un cluster de bases de données Aurora depuis l'extérieur du VPC, utilisez l'adresse de point de terminaison publique du cluster de bases de données.

Pour plus d'informations, consultez Dépannage des problèmes de connexion d'Aurora.

Table des matières

Connexion aux clusters de bases de données Aurora avec les pilotes AWS

La AWS suite de pilotes a été conçue pour accélérer les temps de basculement et de basculement, ainsi que pour l'authentification avecAWS Secrets Manager, Gestion des identités et des accès AWS (IAM) et l'identité fédérée. Les AWS pilotes s'appuient sur la surveillance de l'état du cluster de bases de données et sur la connaissance de la topologie du cluster pour déterminer le nouveau rédacteur. Cette approche réduit les temps de bascule et de basculement à moins de 10 secondes, contre des dizaines de secondes pour les pilotes open source.

Le tableau suivant répertorie les fonctionnalités prises en charge pour tous les pilotes. À mesure que de nouvelles fonctionnalités de service sont introduites, l'objectif de la AWS suite de pilotes est de fournir un support intégré pour ces fonctionnalités de service.

Fonctionnalité	AWSPilote JDBC	AWSPilote Python	AWSPilote ODBC pour MySQL	AWSWrapper NodeJS avancé
Prise en charge du basculement	Oui	Oui	Oui	Oui
Surveillance améliorée du basculement	Oui	Oui	Oui	Oui
Répartition lecture/écriture	Oui	Oui	Non	Oui
Suivi des connexions Aurora	Oui	Oui	Non	Oui
Connexion aux métadonnées du pilote	Oui	N/A	N/A	N/A
Télémétrie	Oui	Oui	Non	Oui
Secrets Manager	Oui	Oui	Oui	Oui
Authentification IAM	Oui	Oui	Oui	Oui
Identité fédérée (AD FS)	Oui	Oui	Non	Oui
Identité fédérée (Okta)	Oui	Oui	Oui	Oui
Base de données Aurora PostgreSQL Limitless	Oui (Aurora PostgreSQL uniquement)	Non	Non	Oui (Aurora PostgreSQL uniquement)

Pour plus d'informations sur les AWS pilotes, consultez le pilote de langue correspondant à votre cluster de base de données Aurora MySQL ou Aurora PostgreSQL.

Connexion à un cluster de bases de données Amazon Aurora MySQL

Pour vous authentifier auprès de votre cluster de base de données Aurora MySQL, vous pouvez utiliser l'authentification par nom d'utilisateur et mot de passe MySQL ou l'authentification de base de données Gestion des identités et des accès AWS (IAM). Pour plus d'informations sur l'utilisation de l'authentification par nom d'utilisateur et mot de passe MySQL, consultez User Account Management dans la documentation MySQL. Pour plus d'informations sur l'utilisation de l'authentification de base de données IAM, consultez Authentification de base de données IAM.

Une fois que vous êtes connecté à votre cluster de bases de données Amazon Aurora compatible avec MySQL 8.0, vous pouvez exécuter les commandes SQL compatibles avec MySQL version 8.0. La version minimale compatible est MySQL 8.0.23. Pour en savoir plus sur la syntaxe SQL de MySQL 8.0, consultez le manuel de référence MySQL 8.0. Pour en savoir plus sur les limitations qui s'appliquent à Aurora MySQL 3, consultez Comparaison d'Aurora MySQL version 3 et de MySQL 8.0 Community Edition.

Une fois que vous êtes connecté à votre cluster de bases de données Amazon Aurora compatible avec MySQL 5.7, vous pouvez exécuter les commandes SQL compatibles avec MySQL version 5.7. Pour plus d'informations sur la syntaxe SQL de MySQL 5.7, consultez le manuel de référence MySQL 5.7. Pour plus d'informations sur les limitations qui s'appliquent à Aurora MySQL 5.7, consultez Aurora MySQL version 2 compatible avec MySQL 5.7.

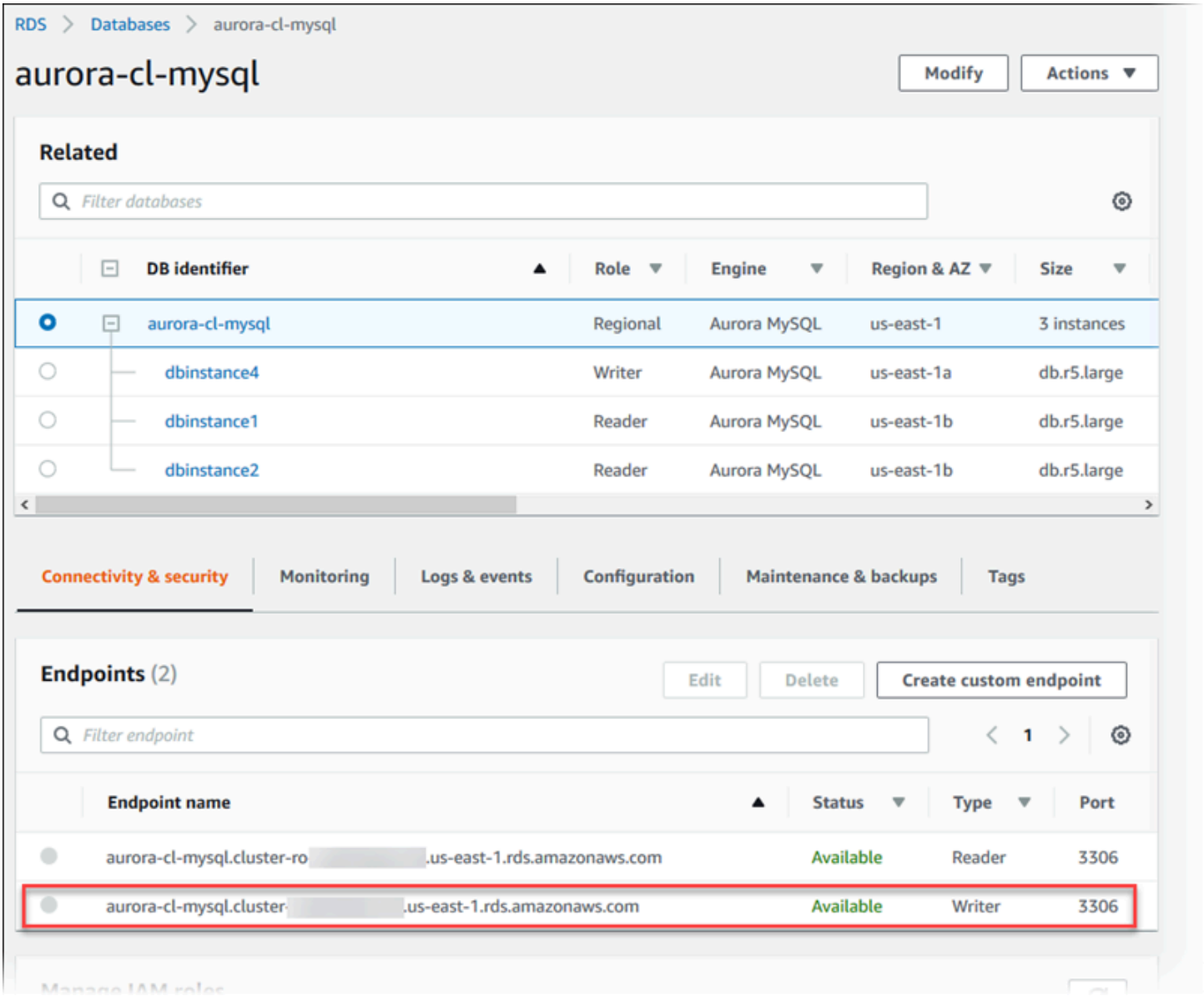
> **Note**
>
> Pour obtenir un guide pratique et détaillé sur la connexion à un cluster de bases de données Amazon Aurora MySQL, consultez le manuel de gestion des connexions Aurora.

Dans la vue détaillée de votre cluster de bases de données, vous pouvez trouver le point de terminaison du cluster, que vous pouvez utiliser dans votre chaîne de connexion MySQL. Le point de terminaison se compose du nom de domaine et du port de votre cluster de bases de données. Par exemple, si la valeur d'un point de terminaison est `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com:3306`, spécifiez les valeurs suivantes dans une chaîne de connexion MySQL :

- Pour un hôte ou nom d'hôte, spécifiez `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- Pour le port, spécifiez `3306` ou la valeur de port que vous avez utilisée lors de la création du cluster de bases de données

Le point de terminaison du cluster vous connecte à l'instance principale du cluster de bases de données. Vous pouvez effectuer des opérations de lecture et d'écriture à l'aide du point de terminaison du cluster. Votre cluster de bases de données peut aussi avoir jusqu'à 15 réplicas Aurora qui prennent en charge l'accès en lecture seule aux données de votre cluster de bases de données. L'instance principale et chaque réplica Aurora possèdent un point de terminaison unique, qui est indépendant du point de terminaison du cluster et vous permet de vous connecter directement à une instance de base de données spécifique du cluster. Le point de terminaison du cluster pointe toujours vers l'instance principale. Si l'instance principale échoue et est remplacée, le point de terminaison du cluster pointe vers la nouvelle instance principale.

Pour afficher le point de terminaison du cluster (point de terminaison d'enregistreur), choisissez Bases de données dans la console Amazon RDS et choisissez le nom du cluster de bases de données dont vous souhaitez afficher les détails.



Rubriques

- Utilitaires de connexion pour Aurora MySQL
- Connexion à Aurora MySQL à l'aide de l'utilitaire MySQL
- Connexion à Aurora MySQL avec le pilote JDBC Amazon Web Services (AWS)
- Connexion à Aurora MySQL avec le pilote Python Amazon Web Services (AWS)
- Connexion à Aurora MySQL avec le pilote ODBC Amazon Web Services (AWS) pour MySQL
- Connexion à Aurora MySQL avec le wrapper Amazon Web Services (AWS) Advanced NodeJS
- Connexion à Aurora MySQL avec SSL

Utilitaires de connexion pour Aurora MySQL

Vous trouverez ci-après certains des utilitaires de connexion que vous pouvez utiliser :

- Ligne de commande : vous pouvez vous connecter à un cluster de bases de données Amazon Aurora en utilisant des outils comme l'utilitaire de ligne de commande MySQL. Pour plus d'informations sur l'utilisation de l'utilitaire MySQL, consultez mysql : client en ligne de commande MySQL dans la documentation sur MySQL.
- Interface utilisateur graphique : vous pouvez utiliser l'utilitaire MySQL Workbench pour vous connecter à l'aide d'une interface utilisateur graphique. Pour plus d'informations, consultez la page Download MySQL Workbench.
- AWSpilotes :
  - Connexion à Aurora MySQL avec le pilote JDBC Amazon Web Services (AWS)
  - Connexion à Aurora MySQL avec le pilote Python Amazon Web Services (AWS)
  - Connexion à Aurora MySQL avec le pilote ODBC Amazon Web Services (AWS) pour MySQL
  - Connexion à Aurora MySQL avec le wrapper Amazon Web Services (AWS) Advanced NodeJS

Connexion à Aurora MySQL à l'aide de l'utilitaire MySQL

Utilisez la procédure suivante. Elle suppose que vous avez configuré votre cluster de bases de données dans un sous-réseau privé de votre VPC. Vous vous connectez à l'aide d'une EC2 instance Amazon que vous avez configurée conformément aux didacticiels présentés dansDidacticiel : Créer un serveur web et une cluster de base de données Amazon Aurora.

> **Note**
>
> Cette procédure n'exige pas d'installer le serveur Web dans le didacticiel, mais elle exige d'installer MariaDB 10.5.

Pour vous connecter à un cluster de bases de données à l'aide de l'utilitaire MySQL

1. Connectez-vous à l' EC2 instance que vous utilisez pour vous connecter à votre cluster de base de données.

   Vous devez visualiser des résultats similaires à ce qui suit.

```
Last login: Thu Jun 23 13:32:52 2022 from xxx.xxx.xxx.xxx

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-xxx.xxx ~]$
```

2. Saisissez la commande suivante dans l'invite de commande pour vous connecter à l'instance de base de données principale de votre cluster de bases de données.

   Pour le paramètre `-h`, remplacez le nom DNS du point de terminaison de votre instance principale. Pour le paramètre `-u`, remplacez l'ID d'utilisateur d'un compte d'utilisateur de base de données.

   ```
   mysql -h primary-instance-endpoint.AWS_account.AWS_Region.rds.amazonaws.com -P 3306
    -u database_user -p
   ```

   Par exemple :

   ```
   mysql -h my-aurora-cluster-instance.c1xy5example.123456789012.eu-
   central-1.rds.amazonaws.com -P 3306 -u admin -p
   ```

3. Saisissez le mot de passe de l'utilisateur de la base de données.

   Vous devez visualiser des résultats similaires à ce qui suit.

   ```
   Welcome to the MariaDB monitor.  Commands end with ; or \g.
   Your MySQL connection id is 1770
   Server version: 8.0.23 Source distribution

   Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

   MySQL [(none)]>
   ```

4. Saisissez vos commandes SQL.

Connexion à Aurora MySQL avec le pilote JDBC Amazon Web Services (AWS)

Le pilote JDBC Amazon Web Services (AWS) est conçu comme un wrapper JDBC avancé. Ce wrapper complète et étend les fonctionnalités d'un pilote JDBC existant pour aider les applications à tirer parti des fonctionnalités des bases de données en cluster telles qu'Aurora MySQL. Le pilote est compatible directement avec le Connector/J driver and the community MariaDB Connector/J pilote MySQL communautaire.

Pour installer le pilote AWS JDBC, ajoutez le fichier .jar du pilote AWS JDBC (situé dans l'application`CLASSPATH`) et conservez les références au pilote communautaire correspondant. Mettez à jour le préfixe d'URL de connexion correspondant comme suit :

- `jdbc:mysql://` sur `jdbc:aws-wrapper:mysql://`
- `jdbc:mariadb://` sur `jdbc:aws-wrapper:mariadb://`

Pour plus d'informations sur le pilote AWS JDBC et des instructions complètes pour son utilisation, consultez le référentiel de pilotes JDBC Amazon Web Services (AWS). GitHub

> **Note**
>
> La version 3.0.3 de l'utilitaire Connector/J MariaDB ne prend plus en charge les clusters de base de données Aurora. Nous vous recommandons donc vivement de passer au pilote JDBC. AWS

Connexion à Aurora MySQL avec le pilote Python Amazon Web Services (AWS)

Le pilote Python Amazon Web Services (AWS) est conçu comme un wrapper Python avancé. Ce wrapper complète et étend les fonctionnalités du pilote open source Psycopg. Le pilote AWS Python prend en charge les versions 3.8 et supérieures de Python. Vous pouvez installer le package `aws-advanced-python-wrapper` à l'aide de la commande `pip`, en même temps que les packages `psycopg` open source.

Pour plus d'informations sur le pilote AWS Python et des instructions complètes pour son utilisation, consultez le GitHub référentiel de pilotes Python Amazon Web Services (AWS).

Connexion à Aurora MySQL avec le pilote ODBC Amazon Web Services (AWS) pour MySQL

Le pilote AWS ODBC pour MySQL est un pilote client conçu pour la haute disponibilité d'Aurora MySQL. Le pilote peut coexister avec le Connector/ODBC pilote MySQL et est compatible avec les mêmes flux de travail.

Pour plus d'informations sur le pilote AWS ODBC pour MySQL et des instructions complètes pour son installation et son utilisation, consultez le référentiel du pilote ODBC pour MySQL GitHub d'Amazon Web Services (AWS).

Connexion à Aurora MySQL avec le wrapper Amazon Web Services (AWS) Advanced NodeJS

L'AWSAdvanced NodeJS Wrapper complète et étend les fonctionnalités d'un pilote NodeJS existant. Il permet aux applications de tirer parti des fonctionnalités des bases de données en cluster telles qu'Aurora MySQL.

Pour plus d'informations sur l'AWSAdvanced NodeJS Wrapper et des instructions complètes pour son utilisation, consultez le référentiel Amazon Web Services (AWS) Advanced NodeJS Wrapper. GitHub

Connexion à Aurora MySQL avec SSL

Vous pouvez utiliser le chiffrement SSL sur les connexions à une instance de base de données Aurora MySQL. Pour plus d'informations, consultez Connexions TLS aux clusters de bases de données Aurora MySQL.

Pour vous connecter avec SSL, choisissez l'utilitaire MySQL comme décrit dans la procédure suivante. Si vous utilisez l'authentification de base de données IAM, vous devez utiliser une connexion SSL. Pour plus d'informations, consultez Authentification de base de données IAM.

> **Note**
>
> Pour se connecter au point de terminaison du cluster à l'aide de SSL, votre utilitaire de connexion client doit prendre en charge les SAN (Subject Alternative Names). Si votre utilitaire de connexion client ne prend pas en charge les SAN, vous pouvez vous connecter directement aux instances de votre cluster DB Aurora. Pour plus d'informations sur les points de terminaison Aurora, consultez Connexions de point de terminaison Amazon Aurora.

Pour vous connecter à un cluster de bases de données avec SSL en utilisant l'utilitaire MySQL

1. Téléchargez la clé publique du certificat de signature Amazon RDS.

   Pour plus d'informations sur le téléchargement de certificats, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.
2. Saisissez la commande suivante dans une invite de commande pour vous connecter à l'instance principale d'un cluster de bases de données avec SSL en utilisant l'utilitaire MySQL. Pour le paramètre `-h`, remplacez le nom DNS du point de terminaison de votre instance principale. Pour le paramètre `-u`, remplacez l'ID d'utilisateur d'un compte d'utilisateur de base de données. Pour le paramètre `--ssl-ca`, remplacez le nom de fichier du certificat SSL par le nom approprié. Entrez le mot de passe de l'utilisateur maître quand vous y êtes invité.

   ```
   mysql -h mycluster-primary.123456789012.us-east-1.rds.amazonaws.com -u admin_user -p --ssl-ca=[full path]global-bundle.pem --ssl-verify-server-cert
   ```

Vous devez visualiser des résultats similaires à ce qui suit.

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 350
Server version: 8.0.26-log MySQL Community Server (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Pour obtenir des instructions générales sur la construction de chaînes de connexion RDS for MySQL et sur la recherche de la clé publique des connexions SSL, consultez Connexion à une instance de base de données exécutant le moteur de base de données MySQL.

Connexion à un cluster de bases de données Amazon Aurora PostgreSQL

Vous pouvez vous connecter à une instance de base de données dans un cluster de bases de données Amazon Aurora PostgreSQL à l'aide des mêmes outils que ceux que vous utilisez pour vous connecter à une base de données PostgreSQL. Dans ce cadre, vous utilisez la même clé publique que celle utilisée pour les connexions SSL (Secure Sockets Layer). Vous pouvez utiliser les informations de point de terminaison et de port de l'instance principale ou des réplicas Aurora de votre cluster de bases de données Aurora PostgreSQL dans la chaîne de connexion d'un script, d'un

utilitaire ou d'une application qui se connecte à une instance de base de données PostgreSQL. Dans la chaîne de connexion, spécifiez l'adresse DNS du point de terminaison de l'instance principale ou du réplica Aurora comme paramètre d'hôte. Spécifiez le numéro de port du point de terminaison comme paramètre de port.

Lorsque vous êtes connecté à une instance de base de données dans votre cluster de bases de données Amazon Aurora PostgreSQL, vous pouvez exécuter toute commande SQL compatible avec PostgreSQL.

Vous pouvez trouver le nom, le statut, le type et le numéro de port du point de terminaison du cluster dans la vue des détails du cluster de bases de données Aurora PostgreSQL. Vous pouvez utiliser le point de terminaison et le numéro de port dans votre chaîne de connexion PostgreSQL. Par exemple, si la valeur d'un point de terminaison est `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`, spécifiez les valeurs suivantes dans une chaîne de connexion PostgreSQL :

- Pour un hôte ou nom d'hôte, spécifiez `mycluster.cluster-123456789012.us-east-1.rds.amazonaws.com`
- Pour le port, spécifiez `5432` ou la valeur de port que vous avez utilisée lors de la création du cluster de bases de données

Le point de terminaison du cluster vous connecte à l'instance principale du cluster de bases de données. Vous pouvez effectuer des opérations de lecture et d'écriture à l'aide du point de terminaison du cluster. Votre cluster de bases de données peut aussi avoir jusqu'à 15 réplicas Aurora qui prennent en charge l'accès en lecture seule aux données de votre cluster de bases de données. Chaque instance de base de données du cluster Aurora (c'est-à-dire, l'instance principale et chaque réplica Aurora) possède un point de terminaison unique qui est indépendant du point de terminaison du cluster. Ce point de terminaison unique vous permet de vous connecter directement à une instance de base de données spécifique dans le cluster. Le point de terminaison du cluster pointe toujours vers l'instance principale. Si l'instance principale échoue et est remplacée, le point de terminaison du cluster pointe vers la nouvelle instance principale.

Pour afficher le point de terminaison du cluster (point de terminaison d'enregistreur), choisissez Bases de données dans la console Amazon RDS et choisissez le nom du cluster de bases de données dont vous souhaitez afficher les détails.
Point de terminaison d'enregistreur Aurora PostgreSQL.

Utilitaires de connexion pour Aurora PostgreSQL

Vous trouverez ci-après certains des utilitaires de connexion que vous pouvez utiliser :

- Ligne de commande – Vous pouvez vous connecter aux clusters de bases de données Aurora PostgreSQL en utilisant des outils comme psql, le terminal interactif PostgreSQL. Pour plus d'informations sur l'utilisation du terminal interactif PostgreSQL, consultez psql dans la documentation PostgreSQL.
- Interface utilisateur graphique : vous pouvez utiliser l'utilitaire pgAdmin pour vous connecter aux clusters de bases de données Aurora PostgreSQL à partir d'une interface utilisateur. Pour plus d'informations, consultez la page de téléchargement sur le site web pgAdmin.
- AWSpilotes :
  - Connexion à Aurora PostgreSQL avec le pilote Amazon Web Services (AWS) JDBC
  - Connexion à Aurora PostgreSQL avec le pilote Amazon Web Services (AWS) Python
  - Connexion à Aurora PostgreSQL avec le wrapper Amazon Web Services (AWS) Advanced NodeJS

Connexion à Aurora PostgreSQL avec le pilote Amazon Web Services (AWS) JDBC

Le pilote JDBC Amazon Web Services (AWS) est conçu comme un wrapper JDBC avancé. Ce wrapper complète et étend les fonctionnalités d'un pilote JDBC existant pour aider les applications à tirer parti des fonctionnalités des bases de données en cluster telles qu'Aurora PostgreSQL. Le pilote est compatible avec le pilote communautaire pgJDBC.

Pour installer le pilote AWS JDBC, ajoutez le fichier .jar du pilote AWS JDBC (situé dans l'applicationCLASSPATH) et conservez les références au pilote communautaire pgJDBC. Mettez à jour le préfixe de l'URL de connexion en remplaçant `jdbc:postgresql://` par `jdbc:aws-wrapper:postgresql://`.

Pour plus d'informations sur le pilote AWS JDBC et des instructions complètes pour son utilisation, consultez le référentiel de pilotes JDBC Amazon Web Services (AWS). GitHub

Connexion à Aurora PostgreSQL avec le pilote Amazon Web Services (AWS) Python

Le pilote Python Amazon Web Services (AWS) est conçu comme un wrapper Python avancé. Ce wrapper complète et étend les fonctionnalités du pilote open source Psycopg. Le pilote AWS Python prend en charge les versions 3.8 et supérieures de Python. Vous pouvez installer le package `aws-`

`advanced-python-wrapper` à l'aide de la commande `pip`, en même temps que les packages `psycopg` open source.

Pour plus d'informations sur le pilote AWS Python et des instructions complètes pour son utilisation, consultez le GitHub référentiel de pilotes Python Amazon Web Services (AWS).

Connexion à Aurora PostgreSQL avec le wrapper Amazon Web Services (AWS) Advanced NodeJS

L'AWSAdvanced NodeJS Wrapper complète et étend les fonctionnalités d'un pilote NodeJS existant. Il permet aux applications de tirer parti des fonctionnalités des bases de données en cluster telles qu'Aurora PostgreSQL.

Pour plus d'informations sur l'AWSAdvanced NodeJS Wrapper et des instructions complètes pour son utilisation, consultez le référentiel Amazon Web Services (AWS) Advanced NodeJS Wrapper. GitHub

Dépannage des problèmes de connexion d'Aurora

Les causes les plus courantes d'échec de connexion à un nouveau cluster de bases de données Aurora sont les suivantes :

- Le groupe de sécurité du VPC n'autorise pas l'accès : votre VPC doit autoriser les connexions depuis votre appareil ou depuis une EC2 instance Amazon en configurant correctement le groupe de sécurité dans le VPC. Pour résoudre ce problème, modifiez les règles de trafic entrant du groupe de sécurité de votre VPC pour autoriser les connexions. Pour obtenir un exemple, consultez Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement).
- Port bloqué par les règles de pare-feu – Vérifiez la valeur du port configuré pour votre cluster de bases de données Aurora. Si une règle de pare-feu bloque ce port, vous pouvez recréer l'instance à l'aide d'un autre port.
- Configuration IAM incomplète ou incorrecte : si vous avez créé votre instance de base de données Aurora pour utiliser l'authentification basée sur IAM, assurez-vous qu'elle est correctement configurée. Pour plus d'informations, consultez Authentification de base de données IAM.

Pour plus d'informations sur la résolution des problèmes de connexion de base de données Aurora, consultez Impossible de se connecter à l'instance de base de données Amazon RDS.

Groupes de paramètres pour Amazon Aurora

Les paramètres de base de données spécifient comment la base de données est configurée. Par exemple, les paramètres de base de données peuvent spécifier la quantité de ressources, telles que la mémoire, à allouer à une base de données.

Vous gérez la configuration de votre base de données en associant vos instances de base de données et clusters de base de données Aurora à des groupes de paramètres. Aurora définit des groupes de paramètres avec des paramètres par défaut. Vous pouvez également définir vos propres groupes de paramètres à l'aide de paramètres personnalisés.

Rubriques

Présentation des groupes de paramètres

Un groupe de paramètres de cluster de bases de données sert de conteneur pour les valeurs de configuration du moteur qui sont appliquées à chaque instance de base de données dans un cluster de bases de données Aurora. Par exemple, le modèle de stockage partagé Aurora requiert que chaque instance de base de données d'un cluster Aurora utilise la même valeur pour les paramètres tels que `innodb_file_per_table`. Les paramètres qui affectent l'organisation du stockage physique font donc partie du groupe de paramètres du cluster. Le groupe de paramètres du cluster de bases de données contient également des valeurs par défaut pour tous les paramètres au niveau de l'instance.

Un groupe de paramètres de base de données sert de conteneur pour les valeurs de configuration du moteur qui sont appliquées à une ou plusieurs instances de base de données. Les groupes de paramètres de base de données s'appliquent aux instances de base de données à la fois dans Amazon RDS et dans Aurora. Ces paramètres de configuration s'appliquent à des propriétés qui

peuvent varier entre les instances de base de données d'un cluster Aurora comme, par exemple, les tailles des mémoires tampons.

Rubriques

Groupes de paramètres par défaut et personnalisés

Si vous créez une instance de base de données sans spécifier de groupe de paramètres de base de données, l'instance de base de données utilise un groupe de paramètres de base de données par défaut. De même, si vous créez un cluster de base de données Aurora sans spécifier de groupe de paramètres de cluster de base de données, le cluster de base de données utilise un groupe de paramètres de cluster de base de données par défaut. Chaque groupe de paramètres par défaut contient les valeurs par défaut du moteur de base de données, ainsi que celles du système Amazon RDS en fonction du moteur, de la classe de calcul et de l'espace de stockage alloué de l'instance.

Vous ne pouvez pas modifier les valeurs de paramètre d'un groupe de paramètres de base de données par défaut. Au lieu de cela, vous pouvez effectuer les actions suivantes :

1. Créez un groupe de paramètres.
2. Modifiez les paramètres souhaités. Il n'est pas possible de modifier tous les paramètres du moteur de base de données dans un groupe de paramètres.
3. Modifiez votre instance de base de données ou votre cluster de bases de données afin d'associer le nouveau groupe de paramètres.

   Pour plus d'informations sur la modification d'un cluster de bases de données ou d'une instance de base de données, consultez Modification d'un cluster de bases de données Amazon Aurora.

> **Note**
>
> Si vous avez modifié votre instance de base de données pour utiliser un groupe de paramètres personnalisés et que vous démarrez l'instance de base de données, RDS

redémarre automatiquement l'instance de base de données dans le cadre du processus de démarrage. Pour les instances RDS for SQL Server multi-AZ pour lesquelles l'option Toujours active ou Mise en miroir est activée, un basculement est attendu lorsque l'instance est redémarrée après l'opération de démarrage.

RDS applique les paramètres statiques et dynamiques modifiés dans un groupe de paramètres nouvellement associé uniquement après le redémarrage de l'instance de base de données. Toutefois, si vous modifiez des paramètres dynamiques dans le groupe de paramètres de base de données après l'avoir associé à l'instance de base de données, ces modifications sont appliquées immédiatement sans redémarrage. Pour plus d'informations sur la modification du groupe de paramètres de base de données, consultez Modification d'un cluster de bases de données Amazon Aurora.

Si vous mettez à jour les paramètres d'un groupe de paramètres de base de données, les modifications effectuées s'appliquent à toutes les instances de base de données qui sont associées à ce groupe de paramètres. De même, si vous mettez à jour les paramètres d'un groupe de paramètres de cluster de bases de données Aurora, les modifications effectuées s'appliquent à tous les clusters de bases de données Aurora qui sont associés à ce groupe de paramètres du cluster de bases de données.

Si vous ne souhaitez pas créer de groupe de paramètres à partir de zéro, vous pouvez copier un groupe de paramètres existant à l'aide de la commande AWS CLI copy-db-parameter-group ou de la commande copy-db-cluster-parameter-group. Vous trouverez peut-être utile de copier un groupe de paramètres dans certains cas. Par exemple, vous pouvez vouloir inclure la plupart des valeurs et paramètres personnalisés d'un groupe de paramètres de dans un nouveau groupe de paramètres de .

Paramètres de cluster de bases de données statiques et dynamiques

Les paramètres de cluster de bases de données sont statiques ou dynamiques. Ils diffèrent comme suit :

- Lorsque vous modifiez un paramètre statique et que vous enregistrez le groupe de paramètres de base de données d'un cluster, la modification du paramètre est appliquée après le redémarrage manuel des instances de base de données dans chaque cluster de bases de données associé. Lorsque vous utilisez l'AWS Management Console pour modifier les valeurs des paramètres du cluster de bases de données, elle utilise toujours `pending-reboot` pour `ApplyMethod`.

- Lorsque vous modifiez un paramètre dynamique, par défaut, la modification du paramètre s'applique immédiatement, sans nécessiter de redémarrage. Lorsque vous utilisez la console, elle utilise toujours `immediate` pour `ApplyMethod`. Pour reporter la modification du paramètre après le redémarrage des instances de base de données d'un cluster de bases de données associé, utilisez l'AWS CLI ou l'API RDS. Définissez `ApplyMethod` sur `pending-reboot` pour le changement de paramètre.

Pour plus d'informations sur l'utilisation de AWS CLI pour modifier la valeur d'un paramètre, consultez modify-db-cluster-parameter-group. Pour plus d'informations sur l'utilisation de l'API RDS pour modifier la valeur d'un paramètre, consultez ModifyDBClusterParameterGroup.

Si vous modifiez le groupe de paramètres du cluster de bases de données associé à un cluster de bases de données, redémarrez les instances de base de données dans le cluster de bases de données. Le redémarrage applique les modifications à toutes les instances de base de données du cluster de bases de données. Pour déterminer si les instances de base de données d'un cluster de bases de données doivent être redémarrées pour appliquer les modifications, exécutez la commande AWS CLI suivante.

```
aws rds describe-db-clusters --db-cluster-identifier db_cluster_identifier
```

Vérifiez la valeur `DBClusterParameterGroupStatus` de l'instance de base de données principale dans la sortie. Si la valeur est `pending-reboot`, alors redémarrez les instances de base de données du cluster de bases de données.

Paramètres d'instance de base de données statiques et dynamiques

Les paramètres d'instance de base de données sont statiques ou dynamiques. Ils diffèrent comme suit :

- Lorsque vous modifiez un paramètre statique et que vous enregistrez le groupe de paramètres de base de données, la modification du paramètre est appliquée après le redémarrage manuel des instances de base de données associées. Pour les paramètres statiques, la console utilise toujours `pending-reboot` pour `ApplyMethod`.
- Lorsque vous modifiez un paramètre dynamique, par défaut, la modification du paramètre s'applique immédiatement, sans nécessiter de redémarrage. Lorsque vous utilisez la AWS Management Console pour modifier les valeurs des paramètres de l'instance de base de données, elle utilise toujours `immediate` pour la valeur `ApplyMethod` des paramètres dynamiques. Pour reporter la modification du paramètre après le redémarrage d'une instance de base de données

associée, utilisez l'AWS CLI ou l'API RDS. Définissez `ApplyMethod` sur `pending-reboot` pour le changement de paramètre.

Pour plus d'informations sur l'utilisation de AWS CLI pour modifier la valeur d'un paramètre, consultez modify-db-parameter-group. Pour plus d'informations sur l'utilisation de l'API RDS pour modifier la valeur d'un paramètre, consultez ModifyDBParameterGroup.

Si une instance de base de données n'utilise pas les dernières modifications apportées à son groupe de paramètres de base de données associé, la console affiche le statut pending-reboot pour le groupe de paramètres de base de données. Le statut n'entraîne pas de redémarrage automatique lors de la fenêtre de maintenance suivante. Pour appliquer les modifications de paramètre les plus récentes apportées à cette instance de base de données, vous devez la redémarrer manuellement.

Paramètres de jeu de caractères

Avant de créer le cluster, définissez tous les paramètres relatifs au jeu de caractères ou au classement de votre base de données dans votre groupe de paramètres. Faites-le également avant d'y créer une base de données. Cela garantit que la base de données par défaut et les nouvelles bases de données utilisent les valeurs de jeu de caractères et de classement que vous spécifiez. Si vous modifiez les paramètres de jeu de caractères ou de classement, les modifications de paramètre ne sont pas appliquées aux bases de données existantes.

Pour certains moteurs de base de données, vous pouvez modifier les valeurs de jeu de caractères ou de classement pour une base de données existante à l'aide de la commande `ALTER DATABASE`, par exemple :

```
ALTER DATABASE database_name CHARACTER SET character_set_name COLLATE collation;
```

Pour plus d'informations sur le changement de jeu de caractères ou de valeurs de classement d'une base de données, consultez la documentation de votre moteur de base de données.

Paramètres et valeurs de paramètres pris en charge

Pour déterminer les paramètres pris en charge pour votre moteur de base de données, affichez les paramètres du groupe de paramètres de base de données et du groupe de paramètres de cluster de bases de données utilisés par l'instance de base de données ou le cluster de bases de données. Pour plus d'informations, consultez Affichage des valeurs de paramètres pour un groupe de paramètres de base de données dans Amazon Aurora et Affichage des valeurs de paramètres pour un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

Dans la plupart des cas, vous pouvez spécifier des valeurs de paramètres entiers et booléens au moyen d'expressions, de formules et de fonctions. Les fonctions peuvent inclure une expression de journal mathématique. Cependant, tous les paramètres ne prennent pas en charge les expressions, les formules et les fonctions des valeurs de paramètres. Pour plus d'informations, consultez Spécification des paramètres de base de données.

Dans le cas d'une base de données mondiale Aurora, vous pouvez spécifier différents paramètres de configuration pour les clusters Aurora individuels. Veillez à vérifier que les paramètres sont suffisamment similaires pour générer un comportement cohérent si vous transformez un cluster secondaire en cluster principal. Par exemple, utilisez les mêmes paramètres pour les fuseaux horaires et les jeux de caractères pour tous les clusters d'une base de données mondiale Aurora.

La configuration incorrecte de paramètres dans un groupe de paramètres peut avoir des effets contraires involontaires, dont une dégradation de la performance et une instabilité du système. Montrez-vous toujours prudent lorsque vous modifiez des paramètres de base de données et sauvegardez vos données avant de modifier un groupe de paramètres. Essayez de modifier les paramètres des groupes de paramètres sur une instance de base de données ou un cluster de bases de données de test avant d'appliquer ces modifications à une instance de base de données ou un cluster de bases de données de production.

Groupes de paramètres de cluster de bases de données pour les clusters de bases de données Amazon Aurora

Les clusters de bases de données Amazon Aurora utilisent les groupes de paramètres de cluster de bases de données. Les sections suivantes décrivent la configuration et la gestion des groupes de paramètres de cluster de bases de données.

Rubriques

- Paramètres de cluster de bases de données et d'instance de base de données Amazon Aurora
- Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora
- Association d'un groupe de paramètres de cluster de bases de données à un cluster de bases de données Amazon Aurora
- Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora
- Réinitialisation des paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora
- Copie d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora

- Répertorier les groupes de paramètres de cluster de bases de données dans Amazon Aurora
- Affichage des valeurs de paramètres pour un groupe de paramètres de cluster de bases de données dans Amazon Aurora
- Suppression d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora

Paramètres de cluster de bases de données et d'instance de base de données Amazon Aurora

Aurora utilise un système de paramètres de configuration à deux niveaux :

- Les paramètres d'un groupe de paramètres de cluster de base de données s'appliquent à toute instance de base de données d'un cluster de base de données. Vos données sont stockées dans le sous-système de stockage partagé Aurora. Par conséquent, tous les paramètres concernant la disposition physique des données de table doivent être les mêmes pour toutes les instances de base de données d'un cluster Aurora. De même, étant donné que les instances de base de données Aurora sont connectées par réplication, tous les paramètres de réplication doivent être identiques au sein d'un cluster Aurora.
- Les paramètres d'un groupe de paramètres de base de données s'appliquent à une seule instance de base de données dans un cluster de bases de données Aurora. Ces paramètres concernent des aspects tels que l'utilisation de la mémoire, que vous pouvez faire varier entre les instances de base de données d'un même cluster Aurora. Par exemple, un cluster contient souvent des instances de base de données ayant des classes d'instance AWS différentes.

Chaque cluster Aurora est associé à un groupe de paramètres de cluster de bases de données. Ce groupe de paramètres attribue des valeurs par défaut pour chaque valeur de configuration pour le moteur de base de données correspondant. Le groupe de paramètres de cluster comprend des valeurs par défaut pour les paramètres de niveau cluster et de niveau instance. Chaque instance de base de données au sein d'un cluster alloué ou Aurora Serverless v2 hérite des paramètres du groupe de paramètres de ce cluster de bases de données.

Chaque instance de base de données est également associée à un groupe de paramètres de base de données. Les valeurs du groupe de paramètres de base de données peuvent remplacer les valeurs par défaut du groupe de paramètres du cluster. Par exemple, si une instance d'un cluster rencontre des problèmes, vous pouvez lui attribuer un groupe de paramètres de base de données personnalisé. Le groupe de paramètres personnalisés peut contenir des réglages spécifiques pour les paramètres liés au débogage ou à l'optimisation des performances.

Aurora affecte des groupes de paramètres par défaut lorsque vous créez un cluster ou une instance de base de données, en fonction du moteur et de la version de base de données spécifiés. Vous pouvez spécifier des groupes de paramètres personnalisés à la place. Vous créez ces groupes de paramètres vous-même, et vous pouvez modifier les valeurs des paramètres. Vous pouvez spécifier ces groupes de paramètres personnalisés au moment de la création. Vous pouvez également modifier ultérieurement un cluster ou une instance de base de données pour utiliser un groupe de paramètres personnalisé.

Pour les instances allouées et Aurora Serverless v2, toute valeur de configuration que vous modifiez dans le groupe de paramètres du cluster de bases de données remplace les valeurs par défaut du groupe de paramètres de base de données. Si vous modifiez les valeurs correspondantes dans le groupe de paramètres de base de données, ces valeurs remplacent celles du groupe de paramètres de cluster de bases de données.

Les valeurs de paramètre de base de données que vous modifiez sont prioritaires par rapport aux valeurs du groupes de paramètres de cluster de bases de données, même si vous rétablissez la valeur par défaut des paramètres de configuration. Vous pouvez voir quels paramètres sont remplacés en utilisant la commande AWS CLI describe-db-parameters ou l'opération DescribeDBParameters de l'API RDS. Le champ `Source` contient la valeur `user` si vous avez modifié ce paramètre. Pour réinitialiser un ou plusieurs paramètres afin que la valeur du groupe de paramètres du cluster de bases de données soit prioritaire, utilisez la commande AWS CLI reset-db-parameter-group ou l'opération ResetDBParameterGroup de l'API RDS.

Les paramètres de clusters et d'instances de bases de données à votre disposition dans Aurora varient en fonction de la compatibilité du moteur de base de données.

Moteur de base de données	Paramètres
Aurora MySQL	Consultez Paramètres de configuration d'Aurora MySQL. Pour les clusters Aurora Serverless, vous trouverez des informations supplémentaires dans Utilisation des groupes de paramètres pour Aurora Serverless v2 et Groupes de paramètres pour Aurora Serverless v1.
Aurora PostgreSQL	Consultez Paramètres Amazon Aurora PostgreSQL. Pour les clusters Aurora Serverless, vous trouverez des informations supplémentaires dans Utilisation des groupes de

Moteur de base de données	Paramètres
	paramètres pour Aurora Serverless v2 et Groupes de paramètres pour Aurora Serverless v1.

> **Note**
>
> Les clusters Aurora Serverless v1 ne possèdent que des groupes de paramètres de cluster de bases de données, et non des groupes de paramètres de base de données. Pour les clusters Aurora Serverless v2, vous apportez toutes vos modifications aux paramètres personnalisés dans le groupe de paramètres du cluster de bases de données.
> Aurora Serverless v2 utilise à la fois les groupes de paramètres de cluster de bases de données et les groupes de paramètres de base de données. Avec Aurora Serverless v2, vous pouvez modifier presque tous les paramètres de configuration. Aurora Serverless v2 remplace les paramètres de configuration liés à la capacité afin que votre charge de travail ne soit pas interrompue lorsque les instances Aurora Serverless v2 sont réduites.
> Pour en savoir plus sur les paramètres de configuration Aurora Serverless et les paramètres que vous pouvez modifier, consultez Utilisation des groupes de paramètres pour Aurora Serverless v2 et Groupes de paramètres pour Aurora Serverless v1.

Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora

Vous pouvez créer un groupe de paramètres de cluster base de données à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API RDS.

Après avoir créé un groupe de paramètres de base de données, attendez au moins cinq minutes avant de créer un cluster de bases de données qui utilise ce groupe de paramètres de base de données. Cela permet à Amazon RDS de créer entièrement le groupe de paramètres avant qu'il ne soit utilisé par le nouveau cluster de bases de données. Vous pouvez utiliser la page Parameter groups (Groupe de paramètres) de la console Amazon RDS ou la commande describe-db-cluster-parameters pour vérifier que votre groupe de paramètres de cluster de bases de données a été créé.

Les limites suivantes s'appliquent aux noms de groupes de paramètres de cluster de bases de données :

- Ces noms doivent comporter entre 1 et 255 lettres, chiffres ou traits d'union.

Les noms des groupes de paramètres par défaut peuvent inclure un point, par exemple `default.aurora-mysql5.7`. Toutefois, les noms de groupes de paramètres personnalisés ne peuvent pas inclure de point.

- Le premier caractère doit être une lettre.
- Les noms ne peuvent pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs.

Console

Pour créer un groupe de paramètres de cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Choisissez Créer un groupe de paramètres.
4. Pour Nom de groupe de paramètres, entrez le nom du nouveau groupe de paramètres de cluster de bases de données.
5. Pour Description, entrez une description pour le nouveau groupe de paramètres de cluster de bases de données.
6. Pour Type de moteur, choisissez votre moteur de base de données.
7. Dans la liste Famille de groupe de paramètres, choisissez une famille de groupe de paramètres de base de données.
8. Choisissez Créer.

AWS CLI

Pour créer un groupe de paramètres de cluster de bases de données, utilisez la commande d'AWS CLI `create-db-cluster-parameter-group`.

L'exemple suivant crée un groupe de paramètres de cluster de bases de données nommé mydbclusterparametergroup pour Aurora MySQL version 5.7 avec une description de « My new cluster parameter group » (Mon nouveau groupe de paramètres de cluster).

Incluez les paramètres requis suivants :

- `--db-cluster-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Pour répertorier toutes les familles de groupes de paramètres, utilisez la commande suivante :

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> La sortie contient des doublons.

Example

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --db-parameter-group-family aurora-mysql5.7 \
    --description "My new cluster parameter group"
```

Pour Windows :

```
aws rds create-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --db-parameter-group-family aurora-mysql5.7 ^
    --description "My new cluster parameter group"
```

Le résultat produit lors de l'exécution de cette commande est semblable à ce qui suit :

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "mydbclusterparametergroup",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "My new cluster parameter group",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
    }
```

```
}
```

API RDS

Pour créer un groupe de paramètres de cluster de bases de données, utilisez l'action d'API RDS `CreateDBClusterParameterGroup`.

Incluez les paramètres requis suivants :

- `DBClusterParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Association d'un groupe de paramètres de cluster de bases de données à un cluster de bases de données Amazon Aurora

Vous pouvez créer vos propres groupes de paramètres de cluster de bases de données à l'aide de paramètres personnalisés. Vous pouvez associer un groupe de paramètres de cluster de bases de données à un cluster de bases de données à l'aide de AWS Management Console, de AWS CLI ou de l'API RDS. Vous pouvez le faire lorsque vous créez ou modifiez un cluster de bases de données.

Pour plus d'informations sur la création d'un groupe de paramètres de cluster de bases de données, consultez Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora. Pour plus d'informations sur la création d'un cluster de bases de données, consultez Création d'un cluster de bases de données Amazon Aurora. Pour plus d'informations sur la modification d'un cluster de bases de données, consultez Modification d'un cluster de bases de données Amazon Aurora.

> **Note**
>
> Pour les versions d'Aurora PostgreSQL 15.2, 14.7, 13.10, 12.14 et toutes les versions 11, lorsque vous modifiez le groupe de paramètres du cluster de bases de données associé à un cluster de bases de données, redémarrez chaque instance de réplica pour appliquer les modifications.
> Pour déterminer si l'instance de base de données principale d'un cluster de bases de données doit être redémarrée pour appliquer les modifications, exécutez la commande AWS CLI suivante :
> `aws rds describe-db-clusters --db-cluster-identifier db_cluster_identifier`

Vérifiez la valeur `DBClusterParameterGroupStatus` de l'instance de base de données principale dans la sortie. Si la valeur est `pending-reboot`, redémarrez l'instance de base de données principale du cluster de bases de données.

Console

Pour associer un groupe de paramètres de cluster de bases de données à un cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis sélectionnez le cluster de bases de données que vous souhaitez modifier.
3. Sélectionnez Modify. La page Modify DB cluster (Modifier le cluster DB) s'affiche.
4. Modifiez le paramètre DB cluster parameter group (groupe de paramètres de cluster de bases de données).
5. Choisissez Continuer et vérifiez le récapitulatif des modifications.

   La modification est appliquée immédiatement, quel que soit le paramètre Scheduling of modifications (Planification des modifications).
6. Sur la page de confirmation, examinez vos modifications. Si elles sont correctes, choisissez Modifier le cluster pour enregistrer vos modifications.

   Sinon, choisissez Retour pour modifier vos modifications, ou choisissez Annuler pour les annuler.

AWS CLI

Pour associer un groupe de paramètres de cluster de bases de données à un cluster de bases de données, utilisez la commande d'AWS CLI `modify-db-cluster` avec les options suivantes :

- `--db-cluster-name`
- `--db-cluster-parameter-group-name`

L'exemple suivant associe le groupe de paramètres de base de données `mydbclpg` au cluster de bases de données `mydbcluster`.

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --db-cluster-parameter-group-name mydbclpg
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --db-cluster-parameter-group-name mydbclpg
```

API RDS

Pour associer un groupe de paramètres de cluster de bases de données à un cluster de bases de données, utilisez l'opération d'API RDS `ModifyDBCluster` avec les paramètres suivants :

- `DBClusterIdentifier`
- `DBClusterParameterGroupName`

Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora

Vous pouvez modifier les valeurs des paramètres dans un groupe de paramètres de cluster bases de données créé par le client. Vous ne pouvez pas modifier les valeurs des paramètres dans un groupe de paramètres de cluster de bases de données par défaut. Les modifications apportées à des paramètres dans un groupe de paramètres de cluster de bases de données créé par le client sont appliquées à tous les clusters de bases de données qui sont associés au groupe de paramètres de cluster de bases de données.

Console

Pour modifier un groupe de paramètres de cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Dans la liste, sélectionnez le groupe de paramètres que vous souhaitez modifier.
4. Sous Parameter group actions (Actions de groupe de paramètres), choisissez Edit (Modifier).
5. Modifiez les valeurs des paramètres que vous souhaitez remplacer. Vous pouvez parcourir les paramètres en utilisant les touches fléchées en haut à droite de la boîte de dialogue.

   Vous ne pouvez pas modifier les valeurs dans un groupe de paramètres par défaut.
6. Sélectionnez Enregistrer les modifications.
7. Redémarrez l'instance de base de données principale (d'enregistreur) dans le cluster pour y appliquer les modifications.
8. Redémarrez ensuite les instances de base de données du lecteur pour leur appliquer les modifications.

   Si vous ne redémarrez pas les instances de base de données, l'opération de basculement peut prendre plus de temps que d'habitude.

AWS CLI

Pour modifier un groupe de paramètres de cluster de bases de données, utilisez la commande `modify-db-cluster-parameter-group` de l'AWS CLI avec les paramètres requis suivants :

- `--db-cluster-parameter-group-name`
- `--parameters`

L'exemple suivant modifie les valeurs `server_audit_logging` et `server_audit_logs_upload` dans le groupe de paramètres de cluster de bases de données nommé mydbclusterparametergroup.

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" \

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

La commande produit un résultat similaire à ce qui suit :

```
DBCLUSTERPARAMETERGROUP  mydbclusterparametergroup
```

API RDS

Pour modifier un groupe de paramètres de cluster de bases de données, utilisez la commande d'API RDS `ModifyDBClusterParameterGroup` avec les paramètres requis suivants :

- `DBClusterParameterGroupName`
- `Parameters`

Réinitialisation des paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora

Vous pouvez réinitialiser les paramètres à leurs valeurs par défaut dans un groupe de paramètres de cluster de bases de données créé par le client. Les modifications apportées à des paramètres dans un groupe de paramètres de cluster de bases de données créé par le client sont appliquées à tous les clusters de bases de données qui sont associés au groupe de paramètres de cluster de bases de données.

> **Note**
>
> Dans un groupe de paramètres de cluster de bases de données par défaut, les paramètres sont toujours définis sur leurs valeurs par défaut.

Console

Pour réinitialiser les paramètres d'un groupe de paramètres de cluster de bases de données à leurs valeurs par défaut

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Dans la liste, sélectionnez le groupe de paramètres.
4. Sous Parameter group actions (Actions de groupe de paramètres), choisissez Edit (Modifier).
5. Sélectionnez les paramètres que vous souhaitez réinitialiser à leurs valeurs par défaut. Vous pouvez parcourir les paramètres en utilisant les touches fléchées en haut à droite de la boîte de dialogue.

   Vous ne pouvez pas réinitialiser les valeurs dans un groupe de paramètres par défaut.
6. Choisissez Réinitialiser, puis confirmez en sélectionnant Réinitialiser les paramètres.
7. Redémarrez l'instance de base de données principale dans le cluster de bases de données pour appliquer les modifications à toutes les instances de base de données dans le cluster de bases de données.

AWS CLI

Pour réinitialiser les paramètres d'un groupe de paramètres de cluster DB à leurs valeurs par défaut, utilisez la commande `reset-db-cluster-parameter-group` de l'AWS CLI avec l'option requise suivante : `--db-cluster-parameter-group-name`.

Pour réinitialiser tous les paramètres du groupe de paramètres du cluster de bases de données, spécifiez l'option `--reset-all-parameters`. Pour réinitialiser des paramètres spécifiques, spécifiez l'option `--parameters`.

L'exemple suivant réinitialise tous les paramètres du groupe de paramètres de base de données nommé mydbparametergroup à leurs valeurs par défaut.

Example

Pour Linux, macOS ou Unix :

```
aws rds reset-db-cluster-parameter-group \
```

```
    --db-cluster-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Pour Windows :

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

L'exemple suivant restaure les valeurs par défaut de `server_audit_logging` et `server_audit_logs_upload` dans le groupe de paramètres de cluster de bases de données nommé mydbclusterparametergroup.

Example

Pour Linux, macOS ou Unix :

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name mydbclusterparametergroup \
    --parameters "ParameterName=server_audit_logging,ApplyMethod=immediate" \
                 "ParameterName=server_audit_logs_upload,ApplyMethod=immediate"
```

Pour Windows :

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name mydbclusterparametergroup ^
    --parameters
 "ParameterName=server_audit_logging,ParameterValue=1,ApplyMethod=immediate" ^

 "ParameterName=server_audit_logs_upload,ParameterValue=1,ApplyMethod=immediate"
```

La commande produit un résultat similaire à ce qui suit :

```
DBClusterParameterGroupName  mydbclusterparametergroup
```

API RDS

Pour réinitialiser les paramètres d'un groupe de paramètres de cluster de bases de données à leurs valeurs par défaut, utilisez la commande `ResetDBClusterParameterGroup` de l'API RDS avec le paramètre obligatoire suivant : `DBClusterParameterGroupName`.

Pour réinitialiser tous les paramètres du groupe de paramètres du cluster de bases de données, définissez le paramètre `ResetAllParameters` sur `true`. Pour réinitialiser des paramètres spécifiques, spécifiez le paramètre `Parameters`.

Copie d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora

Vous pouvez copier des groupes de paramètres de cluster de bases de données personnalisés que vous créez. La copie d'un groupe de paramètres est une solution pratique lorsque vous avez déjà créé un groupe de paramètres de cluster de bases de données et que vous souhaitez inclure la plupart des valeurs et des paramètres personnalisés de ce groupe dans un nouveau groupe de paramètres de cluster de bases de données. Vous pouvez copier un groupe de paramètres de cluster de bases de données en utilisant la commande copy-db-cluster-parameter-group de l'AWS CLI ou l'opération d'API RDS CopyDBClusterParameterGroup.

Après avoir copié un groupe de paramètres de base de données, attendez au moins cinq minutes avant de créer un cluster de bases de données qui utilise ce groupe de paramètres de base de données. Cela permet à Amazon RDS de copier entièrement le groupe de paramètres avant qu'il ne soit utilisé par le nouveau cluster de bases de données. Vous pouvez utiliser la page Parameter groups (Groupe de paramètres) de la console Amazon RDS ou la commande describe-db-cluster-parameters pour vérifier que votre groupe de paramètres de cluster de bases de données a été créé.

> **Note**
>
> Vous ne pouvez pas copier un groupe de paramètres par défaut. Toutefois, vous pouvez créer un nouveau groupe de paramètres basé sur un groupe de paramètres par défaut. Vous ne pouvez pas copier un groupe de paramètres de cluster de bases de données dans un autre Compte AWS ou une autre Région AWS.

Console

Pour copier un groupe de paramètres de cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Dans la liste, choisissez le groupe de paramètres personnalisé que vous souhaitez copier.

4. Sous Parameter group actions (Actions de groupe de paramètres), choisissez Copy (Copier).
5. Dans New DB parameter group identifier (Nouvel identifiant de groupe de paramètres de base de données), saisissez un nom pour le nouveau groupe de paramètres.
6. Dans Description, saisissez une description pour le nouveau groupe de paramètres.
7. Choisissez Copy.

AWS CLI

Pour copier un groupe de paramètres de cluster de bases de données, utilisez la commande `copy-db-cluster-parameter-group` de l'AWS CLI avec les paramètres requis suivants :

- `--source-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-identifier`
- `--target-db-cluster-parameter-group-description`

L'exemple suivant crée un groupe de paramètres de cluster de bases de données nommé `mygroup2` qui est une copie du groupe de paramètres de cluster de bases de données `mygroup1`.

Example

Pour Linux, macOS ou Unix :

```
aws rds copy-db-cluster-parameter-group \
    --source-db-cluster-parameter-group-identifier mygroup1 \
    --target-db-cluster-parameter-group-identifier mygroup2 \
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

Pour Windows :

```
aws rds copy-db-cluster-parameter-group ^
    --source-db-cluster-parameter-group-identifier mygroup1 ^
    --target-db-cluster-parameter-group-identifier mygroup2 ^
    --target-db-cluster-parameter-group-description "DB parameter group 2"
```

API RDS

Pour copier un groupe de paramètres de cluster de bases de données, utilisez l'opération d'API RDS `CopyDBClusterParameterGroup` avec les paramètres requis suivants :

- `SourceDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupIdentifier`
- `TargetDBClusterParameterGroupDescription`

Répertorier les groupes de paramètres de cluster de bases de données dans Amazon Aurora

Vous pouvez répertorier les groupes de paramètres de cluster de bases de données que vous avez créés pour votre compte AWS.

> **Note**
>
> Les groupes de paramètres par défaut sont automatiquement créés à partir d'un modèle de paramètre par défaut lorsque vous créez un cluster de bases de données pour une version et un moteur de base de données spécifiques. Ces groupes de paramètres par défaut contiennent des valeurs de paramètres préférentielles et ne peuvent pas être modifiés. Lorsque vous créez un groupe de paramètres personnalisé, vous pouvez modifier les réglages des paramètres.

Console

Pour répertorier tous les groupes de paramètres de cluster de bases de données pour un compte AWS

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.

   Les groupes de paramètres de cluster de bases de données apparaissent dans la liste avec Groupe de paramètres de cluster de bases de données pour le Type.

AWS CLI

Pour répertorier tous les groupes de paramètres de cluster de bases de données pour un compte AWS, utilisez la commande `describe-db-cluster-parameter-groups` de la AWS CLI.

Example

L'exemple suivant répertorie tous les groupes de paramètres de cluster de bases de données disponibles pour un compte AWS.

```
aws rds describe-db-cluster-parameter-groups
```

L'exemple suivant décrit le groupe de paramètres mydbclusterparametergroup.

Pour Linux, macOS ou Unix :

```
aws rds describe-db-cluster-parameter-groups \
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

Pour Windows :

```
aws rds describe-db-cluster-parameter-groups ^
    --db-cluster-parameter-group-name mydbclusterparametergroup
```

La commande renvoie une réponse telle que la suivante :

```
{
    "DBClusterParameterGroups": [
        {
            "DBClusterParameterGroupName": "mydbclusterparametergroup",
            "DBParameterGroupFamily": "aurora-mysql5.7",
            "Description": "My new cluster parameter group",
            "DBClusterParameterGroupArn": "arn:aws:rds:us-east-1:123456789012:cluster-
pg:mydbclusterparametergroup"
        }
    ]
}
```

API RDS

Pour répertorier tous les groupes de paramètres de cluster de bases de données pour un compte AWS, utilisez l'action DescribeDBClusterParameterGroups d'API RDS.

Affichage des valeurs de paramètres pour un groupe de paramètres de cluster de bases de données dans Amazon Aurora

Vous pouvez obtenir une liste de tous les paramètres dans un groupe de paramètres de cluster de bases de données et de leurs valeurs.

Console

Pour afficher les valeurs de paramètres pour un groupe de paramètres de cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.

   Les groupes de paramètres de cluster de bases de données apparaissent dans la liste avec Groupe de paramètres de cluster de bases de données pour le Type.
3. Choisissez le nom du groupe de paramètres du cluster de bases de données pour afficher la liste des paramètres associée.

AWS CLI

Pour afficher les valeurs de paramètre d'un groupe de paramètres de cluster de bases de données, utilisez la commande `describe-db-cluster-parameters` de l'AWS CLI avec le paramètre requis suivant.

- `--db-cluster-parameter-group-name`

Example

L'exemple suivant répertorie les paramètres et les valeurs de paramètres pour un groupe de paramètres de cluster de bases de données nommé mydbclusterparametergroup au format JSON.

La commande renvoie une réponse telle que la suivante :

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-
name mydbclusterparametergroup
```

```
{
```

```
    "Parameters": [
        {
            "ParameterName": "allow-suspicious-udfs",
            "Description": "Controls whether user-defined functions that have only an
 xxx symbol for the main function can be loaded",
            "Source": "engine-default",
            "ApplyType": "static",
            "DataType": "boolean",
            "AllowedValues": "0,1",
            "IsModifiable": false,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },
        {
            "ParameterName": "aurora_binlog_read_buffer_size",
            "ParameterValue": "5242880",
            "Description": "Read buffer size used by master dump thread when the switch
 aurora_binlog_use_large_read_buffer is ON.",
            "Source": "engine-default",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "8192-536870912",
            "IsModifiable": true,
            "ApplyMethod": "pending-reboot",
            "SupportedEngineModes": [
                "provisioned"
            ]
        },

...
```

API RDS

Pour afficher les valeurs de paramètre d'un groupe de paramètres de cluster de bases de données, utilisez la commande d'API RDS `DescribeDBClusterParameters` avec le paramètre requis suivant.

- `DBClusterParameterGroupName`

Dans certains cas, les valeurs autorisées pour un paramètre ne sont pas affichées. Il s'agit toujours de paramètres dont la source est la valeur par défaut du moteur de base de données.

Pour afficher les valeurs de ces paramètres, vous pouvez exécuter les instructions SQL suivantes :

- MySQL :

```
-- Show the value of a particular parameter
mysql$ SHOW VARIABLES LIKE '%parameter_name%';

-- Show the values of all parameters
mysql$ SHOW VARIABLES;
```

- PostgreSQL :

```
-- Show the value of a particular parameter
postgresql=> SHOW parameter_name;

-- Show the values of all parameters
postgresql=> SHOW ALL;
```

Suppression d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora

Vous pouvez supprimer un groupe de paramètres de cluster base de données à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API RDS. Un groupe de paramètres de cluster de bases de données peut être supprimé uniquement s'il n'est pas associé à un cluster de bases de données.

Console

Pour supprimer des groupes de paramètres

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.

   Les groupes de paramètres s'affichent dans une liste.
3. Choisissez le nom du groupe de paramètres du cluster de bases de données à supprimer.
4. Choisissez Actions, puis Supprimer.

5. Vérifiez les noms des groupes de paramètres, puis choisissez Supprimer.

AWS CLI

Pour supprimer un groupe de paramètres de cluster de bases de données, utilisez la commande `delete-db-cluster-parameter-group` de l'AWS CLI avec le paramètre requis suivant.

- `--db-parameter-group-name`

Example

L'exemple suivant supprime un groupe de paramètres de cluster de bases de données nommé mydbparametergroup.

```
aws rds delete-db-cluster-parameter-group --db-parameter-group-name mydbparametergroup
```

API RDS

Pour supprimer un groupe de paramètres de cluster de bases de données, utilisez la commande `DeleteDBClusterParameterGroup` de l'API RDS avec le paramètre requis suivant.

- `DBParameterGroupName`

Groupes de paramètres de base de données pour les instances de base de données Amazon Aurora

Les instances de base de données utilisent des groupes de paramètres de base de données. Les sections suivantes décrivent la configuration et la gestion des groupes de paramètres d'une instance de base de données.

Rubriques

- Création d'un groupe de paramètres de base de données dans Amazon Aurora
- Association d'un groupe de paramètres de base de données à une instance de base de données dans Amazon Aurora
- Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora

- Réinitialisation des valeurs par défaut des paramètres d’un groupe de paramètres de base de données dans Amazon Aurora
- Copie d’un groupe de paramètres de base de données dans Amazon Aurora
- Répertorier les groupes de paramètres de base de données dans Amazon Aurora
- Affichage des valeurs de paramètres pour un groupe de paramètres de base de données dans Amazon Aurora
- Suppression d’un groupe de paramètres de base de données dans Amazon Aurora

Création d’un groupe de paramètres de base de données dans Amazon Aurora

Vous pouvez créer un nouveau groupe de paramètres de base de données à l'aide de l'API AWS Management Console, de AWS CLI, ou de l'API RDS.

Les limites suivantes s’appliquent aux noms de groupes de paramètres de base de données :

- Ces noms doivent comporter entre 1 et 255 lettres, chiffres ou traits d’union.

  Les noms des groupes de paramètres par défaut peuvent inclure un point, par exemple `default.mysql8.0`. Toutefois, les noms de groupes de paramètres personnalisés ne peuvent pas inclure de point.
- Le premier caractère doit être une lettre.
- Les noms ne peuvent pas se terminer par un trait d’union ni contenir deux traits d’union consécutifs.

Console

Pour créer un groupe de paramètres de base de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Choisissez Créer un groupe de paramètres.
4. Pour Nom de groupe de paramètres, entrez le nom du nouveau groupe de paramètres de base de données.
5. Pour Description, saisissez une description du nouveau groupe de paramètres de bases de données.

6. Pour Type de moteur, choisissez votre moteur de base de données.
7. Dans la liste Famille de groupe de paramètres, choisissez une famille de groupe de paramètres de base de données.
8. Pour Type, le cas échéant, choisissez Groupe de paramètres de base de données.
9. Choisissez Créer.

AWS CLI

Pour créer un groupe de paramètres de base de données, utilisez la AWS CLI `create-db-parameter-group`commande. L'exemple suivant crée un groupe de paramètres de base de données nommé mydbparametergroup pour MySQL version 8.0 avec la description « My new parameter group » (Mon nouveau groupe de paramètres).

Incluez les paramètres requis suivants :

- `--db-parameter-group-name`
- `--db-parameter-group-family`
- `--description`

Pour répertorier toutes les familles de groupes de paramètres, utilisez la commande suivante :

```
aws rds describe-db-engine-versions --query "DBEngineVersions[].DBParameterGroupFamily"
```

> **Note**
>
> La sortie contient des doublons.

Example

Pour Linux, macOS ou Unix :

```
aws rds create-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --db-parameter-group-family aurora-mysql5.7 \
    --description "My new parameter group"
```

Pour Windows :

```
aws rds create-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --db-parameter-group-family aurora-mysql5.7 ^
    --description "My new parameter group"
```

Le résultat produit lors de l'exécution de cette commande est semblable à ce qui suit :

```
DBPARAMETERGROUP  mydbparametergroup  aurora-mysql5.7  My new parameter group
```

API RDS

Pour créer un groupe de paramètres de base de données, utilisez l'opération d'API RDS `CreateDBParameterGroup`.

Incluez les paramètres requis suivants :

- `DBParameterGroupName`
- `DBParameterGroupFamily`
- `Description`

Association d'un groupe de paramètres de base de données à une instance de base de données dans Amazon Aurora

Vous pouvez créer vos propres groupes de paramètres de base de données avec des paramètres personnalisés. Vous pouvez associer un groupe de paramètres de base de données à une instance de base de données à l' AWS Management Console aide de l'API AWS CLI, de, ou de l'API RDS. Vous pouvez le faire lorsque vous créez ou modifiez une instance de base de données.

Pour plus d'informations sur la création d'un groupe de paramètres de base de données, consultez Création d'un groupe de paramètres de base de données dans Amazon Aurora. Pour plus d'informations sur la modification d'une instance de base de données, consultez Modification d'une instance de base de données dans un cluster de bases de données.

> **Note**
>
> Lorsque vous associez un nouveau groupe de paramètres de base de données à une instance de base de données, les paramètres statiques et dynamiques modifiés sont appliqués uniquement après que l'instance de base de données est redémarrée. Toutefois,

si vous modifiez des paramètres dynamiques dans le groupe de paramètres de base de données après l'avoir associé à l'instance de base de données, ces modifications sont appliquées immédiatement sans redémarrage.

Console

Associer un groupe de paramètres de base de données à une instance de base de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Bases de données, puis l'instance de base de données que vous souhaitez modifier.
3. Sélectionnez Modify (Modifier). La page Modifier l'instance de base de données s'affiche.
4. Modifiez le paramètre DB parameter group (groupe de paramètres de base de données).
5. Choisissez Continuer et vérifiez le récapitulatif des modifications.
6. (Facultatif) Choisissez Appliquer immédiatement pour appliquer les modifications immédiatement. La sélection de cette option peut entraîner une interruption de service dans certains cas.
7. Sur la page de confirmation, examinez vos modifications. Si elles sont correctes, choisissez Modify DB instance (Modifier l'instance de base de données) pour enregistrer vos modifications.

   Ou choisissez Retour pour revoir vos modifications, ou choisissez Annuler pour les annuler.

AWS CLI

Pour associer un groupe de paramètres de base de données à une instance de base de données, utilisez la AWS CLI `modify-db-instance`commande avec les options suivantes :

- `--db-instance-identifier`
- `--db-parameter-group-name`

L'exemple suivant associe le groupe de paramètres de base de données `mydbpg` à l'instance de base de données `database-1`. Les modifications sont appliquées immédiatement en utilisant `--apply-immediately`. Utilisez `--no-apply-immediately` pour appliquer les modifications pendant le créneau de maintenance suivant.

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-instance \
    --db-instance-identifier database-1 \
    --db-parameter-group-name mydbpg \
    --apply-immediately
```

Pour Windows :

```
aws rds modify-db-instance ^
    --db-instance-identifier database-1 ^
    --db-parameter-group-name mydbpg ^
    --apply-immediately
```

API RDS

Pour associer un groupe de paramètres de base de données à une instance de base de données, utilisez l'opération d'API RDS `ModifyDBInstance` avec les paramètres suivants :

- `DBInstanceName`
- `DBParameterGroupName`

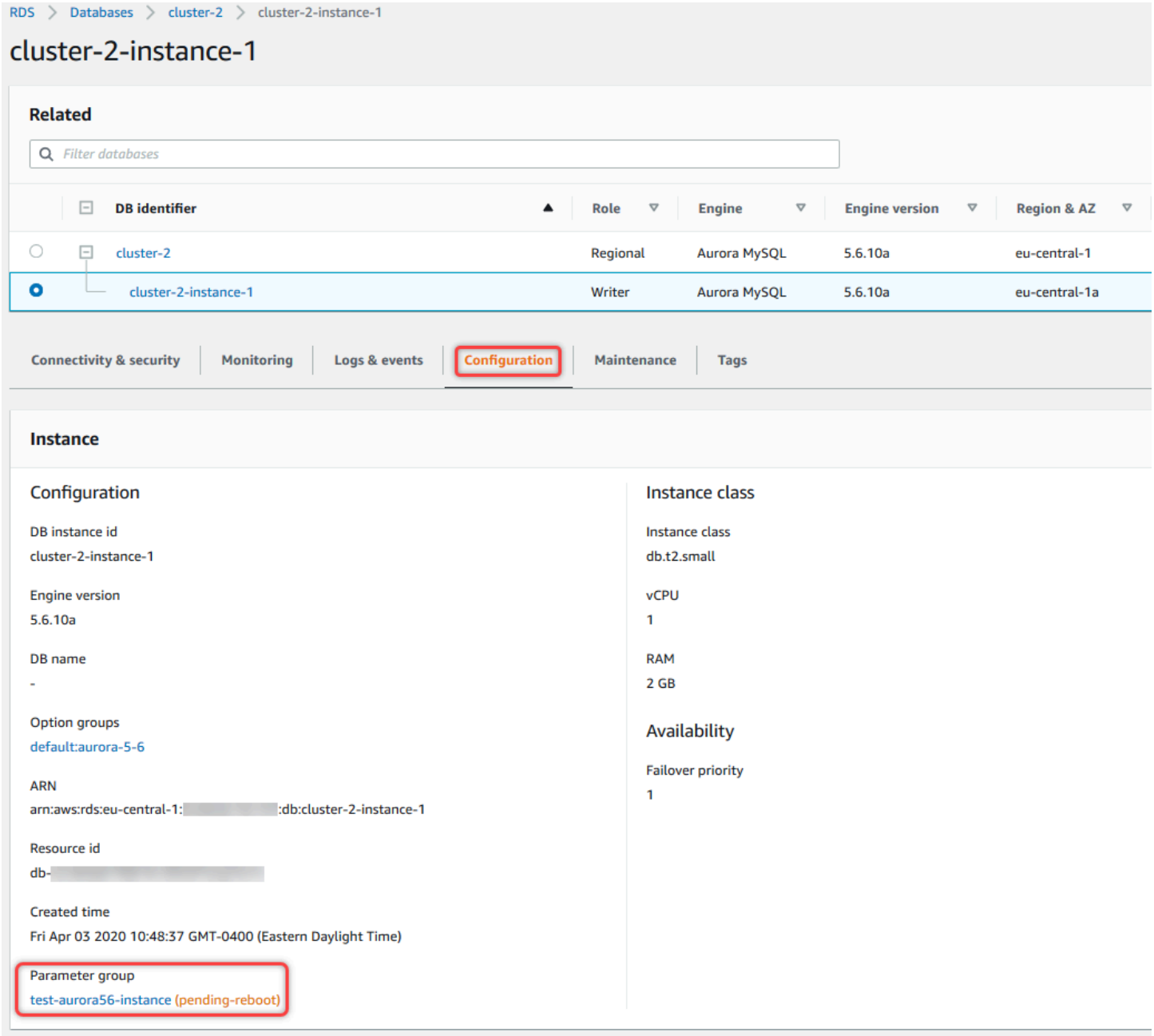
Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora

Vous pouvez modifier des valeurs de paramètres dans un groupe de paramètres de base de données créé par le client. Par contre, vous ne pouvez pas modifier les valeurs de paramètres dans un groupe de paramètres de base de données par défaut. Les modifications apportées à des paramètres dans un groupe de paramètres DB créé par le client sont appliquées à toutes les instances de base de données qui sont associées au groupe de paramètres DB.

Il existe deux types de paramètres : les paramètres dynamiques et les paramètres statiques. Les modifications apportées aux paramètres dynamiques sont appliquées à l'instance de base de données immédiatement sans redémarrage. Les modifications apportées aux paramètres statiques ne sont appliquées qu'après le redémarrage de l'instance de base de données.

La console RDS affiche le statut du groupe de paramètres de base de données associé à une instance de base de données dans l'onglet Configuration. Par exemple, si l'instance de base de

données n'utilise pas les dernières modifications apportées à son groupe de paramètres de base de données associé, la console RDS affiche le groupe de paramètres de base de données avec le statut suivant : pending-reboot. Pour appliquer les modifications de paramètre les plus récentes apportées à cette instance de base de données, vous devez la redémarrer manuellement.



Console

Pour modifier des paramètres dans un groupe de paramètres de bases de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Dans la liste, sélectionnez le groupe de paramètres que vous souhaitez modifier.
4. Sous Parameter group actions (Actions de groupe de paramètres), choisissez Edit (Modifier).
5. Modifiez les valeurs des paramètres que vous souhaitez remplacer. Vous pouvez parcourir les paramètres en utilisant les touches fléchées en haut à droite de la boîte de dialogue.

   Vous ne pouvez pas modifier les valeurs dans un groupe de paramètres par défaut.
6. Sélectionnez Save Changes.

AWS CLI

Pour modifier un groupe de paramètres de base de données, utilisez la AWS CLI `modify-db-parameter-group`commande avec les options requises suivantes :

- `--db-parameter-group-name`
- `--parameters`

L'exemple suivant modifie les valeurs `max_connections` et `max_allowed_packet` dans le groupe de paramètres de base de données nommé mydbparametergroup.

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" \

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters
 "ParameterName=max_connections,ParameterValue=250,ApplyMethod=immediate" ^

 "ParameterName=max_allowed_packet,ParameterValue=1024,ApplyMethod=immediate"
```

La commande produit un résultat similaire à ce qui suit :

```
DBPARAMETERGROUP  mydbparametergroup
```

API RDS

Pour modifier un groupe de paramètres de base de données, utilisez l'opération d'API RDS `ModifyDBParameterGroup` avec les paramètres requis suivants :

- `DBParameterGroupName`
- `Parameters`

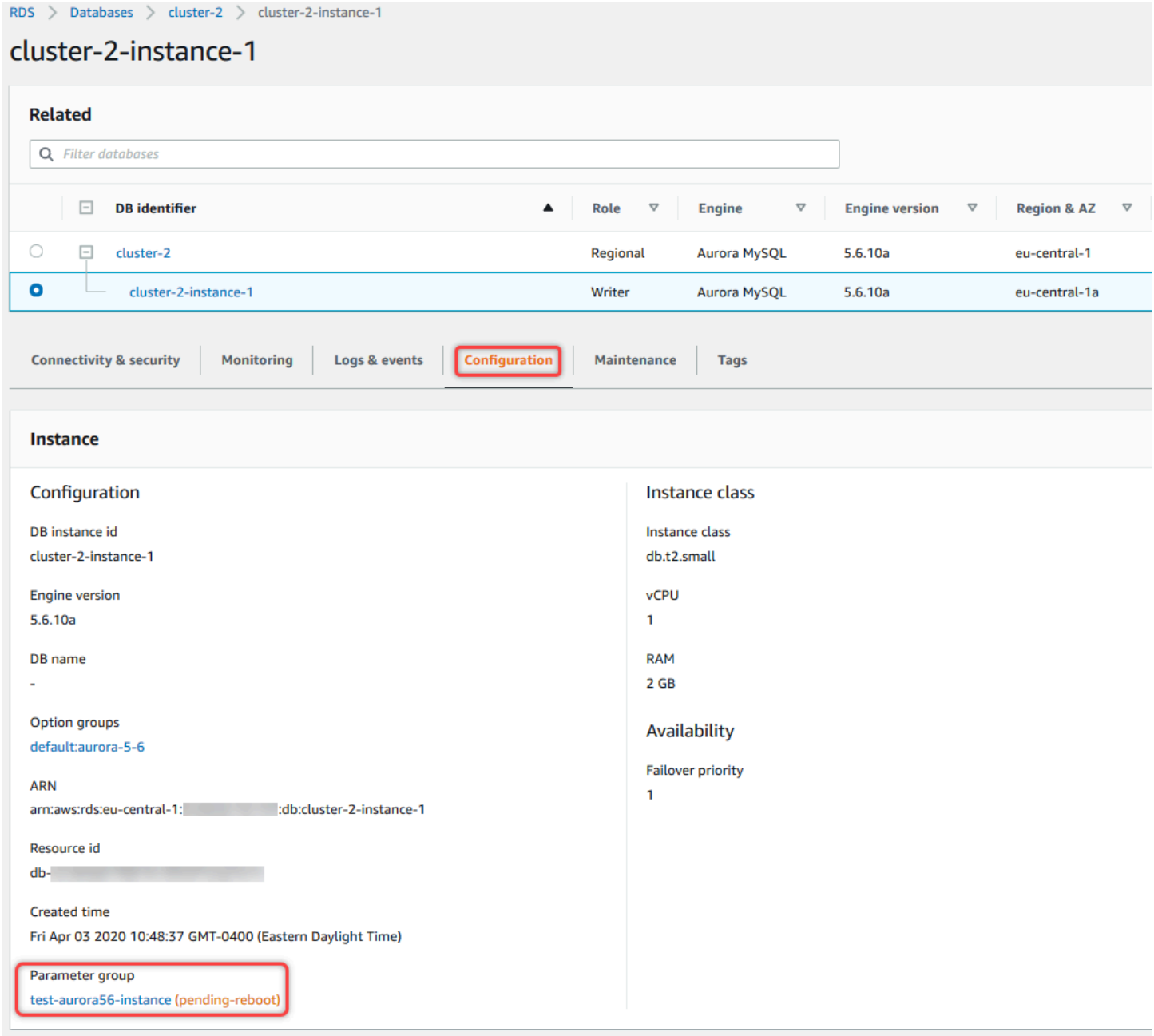
Réinitialisation des valeurs par défaut des paramètres d'un groupe de paramètres de base de données dans Amazon Aurora

Vous pouvez rétablir les valeurs par défaut des paramètres d'un groupe de paramètres de base de données créé par le client. Les modifications apportées à des paramètres dans un groupe de paramètres DB créé par le client sont appliquées à toutes les instances de base de données qui sont associées au groupe de paramètres DB.

Lorsque vous utilisez la console, vous pouvez rétablir les valeurs par défaut de paramètres spécifiques. Cependant, vous ne pouvez pas facilement réinitialiser tous les paramètres du groupe de paramètres de base de données simultanément. Lorsque vous utilisez l'API AWS CLI ou RDS, vous pouvez rétablir les valeurs par défaut de certains paramètres. Vous pouvez également réinitialiser tous les paramètres du groupe de paramètres de base de données simultanément.

Les modifications apportées à certains paramètres sont appliquées immédiatement à l'instance de base de données sans redémarrage. Les modifications apportées à d'autres paramètres s'appliquent uniquement après le redémarrage de l'instance de base de données. La console RDS affiche le statut du groupe de paramètres de base de données associé à une instance de base de données dans l'onglet Configuration. Par exemple, supposons que l'instance de base de données n'utilise pas

les dernières modifications apportées à son groupe de paramètres de base de données associé. Si tel est le cas, la console RDS affiche le groupe de paramètres de base de données avec le statut suivant : pending-reboot. Pour appliquer les modifications de paramètre les plus récentes apportées à cette instance de base de données, vous devez la redémarrer manuellement.



> **Note**
>
> Dans un groupe de paramètres de base de données par défaut, les paramètres sont toujours définis sur leurs valeurs par défaut.

Console

Pour réinitialiser les valeurs par défaut des paramètres d'un groupe de paramètres de base de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Dans la liste, sélectionnez le groupe de paramètres.
4. Sous Parameter group actions (Actions de groupe de paramètres), choisissez Edit (Modifier).
5. Sélectionnez les paramètres que vous souhaitez réinitialiser à leurs valeurs par défaut. Vous pouvez parcourir les paramètres en utilisant les touches fléchées en haut à droite de la boîte de dialogue.

   Vous ne pouvez pas réinitialiser les valeurs dans un groupe de paramètres par défaut.
6. Choisissez Réinitialiser, puis confirmez en sélectionnant Réinitialiser les paramètres.

AWS CLI

Pour réinitialiser certains ou tous les paramètres d'un groupe de paramètres de base de données, utilisez la AWS CLI `reset-db-parameter-group`commande avec l'option requise suivante :`--db-parameter-group-name`.

Pour réinitialiser tous les paramètres du groupe de paramètres de base de données, spécifiez l'option `--reset-all-parameters`. Pour réinitialiser des paramètres spécifiques, spécifiez l'option `--parameters`.

L'exemple suivant réinitialise tous les paramètres du groupe de paramètres de base de données nommé mydbparametergroup à leurs valeurs par défaut.

Example

Pour Linux, macOS ou Unix :

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --reset-all-parameters
```

Pour Windows :

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --reset-all-parameters
```

L'exemple suivant réinitialise les valeurs par défaut des options `max_connections` et `max_allowed_packet` du groupe de paramètres de base de données mydbparametergroup.

Example

Pour Linux, macOS ou Unix :

```
aws rds reset-db-parameter-group \
    --db-parameter-group-name mydbparametergroup \
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" \
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

Pour Windows :

```
aws rds reset-db-parameter-group ^
    --db-parameter-group-name mydbparametergroup ^
    --parameters "ParameterName=max_connections,ApplyMethod=immediate" ^
                 "ParameterName=max_allowed_packet,ApplyMethod=immediate"
```

La commande produit un résultat similaire à ce qui suit :

```
DBParameterGroupName  mydbparametergroup
```

API RDS

Pour réinitialiser les valeurs par défaut des paramètres d'un groupe de paramètres de base de données, utilisez la commande `ResetDBParameterGroup` de l'API RDS avec le paramètre obligatoire suivant : `DBParameterGroupName`.

Pour réinitialiser tous les paramètres du groupe de paramètres de base de données, définissez le `ResetAllParameters` paramètre sur `true`. Pour réinitialiser des paramètres spécifiques, spécifiez le paramètre `Parameters`.

Copie d'un groupe de paramètres de base de données dans Amazon Aurora

Vous pouvez copier des groupes de paramètres DB personnalisés que vous créez. La copie d'un groupe de paramètres peut s'avérer une solution pratique. Par exemple, lorsque vous avez créé un

groupe de paramètres de base de données et que vous souhaitez inclure la plupart de ses valeurs et paramètres personnalisés dans un nouveau groupe de paramètres de base de données. Vous pouvez copier un groupe de paramètres de base de données à l'aide du AWS Management Console. Vous pouvez également utiliser la AWS CLI copy-db-parameter-groupcommande ou l'opération RDS API Copy DBParameter Group.

Après avoir copié un groupe de paramètres de base de données, patientez au moins 5 minutes avant de créer votre première instance de base de données utilisant ce groupe comme groupe de paramètres par défaut. Cela permet à Amazon RDS de terminer complètement l'action de copie avant l'utilisation du groupe de paramètres. Cela est particulièrement important pour les paramètres qui sont essentiels lors de la création de la base de données par défaut d'une instance de base de données. Parmi ces paramètres, citons par exemple le jeu de caractères de la base de données par défaut défini par le paramètre `character_set_database`. Utilisez l'option Groupes de paramètres de la console Amazon RDS ou la describe-db-parameterscommande pour vérifier que votre groupe de paramètres de base de données est créé.

> **Note**
>
> Vous ne pouvez pas copier un groupe de paramètres par défaut. Toutefois, vous pouvez créer un nouveau groupe de paramètres basé sur un groupe de paramètres par défaut. Vous ne pouvez pas copier un groupe de paramètres de base de données vers un autre Compte AWS ou Région AWS.

Console

Pour copier un groupe de paramètres DB

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Dans la liste, choisissez le groupe de paramètres personnalisé que vous souhaitez copier.
4. Sous Parameter group actions (Actions de groupe de paramètres), choisissez Copy (Copier).
5. Dans New DB parameter group identifier (Nouvel identifiant de groupe de paramètres de base de données), saisissez un nom pour le nouveau groupe de paramètres.
6. Dans Description, saisissez une description pour le nouveau groupe de paramètres.
7. Choisissez Copy.

AWS CLI

Pour copier un groupe de paramètres de base de données, utilisez la AWS CLI `copy-db-parameter-group`commande avec les options requises suivantes :

- `--source-db-parameter-group-identifier`
- `--target-db-parameter-group-identifier`
- `--target-db-parameter-group-description`

L'exemple suivant crée un groupe de paramètres DB nommé `mygroup2` qui est une copie du groupe de paramètres DB `mygroup1`.

Example

Pour Linux, macOS ou Unix :

```
aws rds copy-db-parameter-group \
    --source-db-parameter-group-identifier mygroup1 \
    --target-db-parameter-group-identifier mygroup2 \
    --target-db-parameter-group-description "DB parameter group 2"
```

Pour Windows :

```
aws rds copy-db-parameter-group ^
    --source-db-parameter-group-identifier mygroup1 ^
    --target-db-parameter-group-identifier mygroup2 ^
    --target-db-parameter-group-description "DB parameter group 2"
```

API RDS

Pour copier un groupe de paramètres de base de données, utilisez l'opération d'API RDS `CopyDBParameterGroup` avec les paramètres requis suivants :

- `SourceDBParameterGroupIdentifier`
- `TargetDBParameterGroupIdentifier`
- `TargetDBParameterGroupDescription`

Répertorier les groupes de paramètres de base de données dans Amazon Aurora

Vous pouvez répertorier les groupes de paramètres de base de données que vous avez créés pour votre AWS compte.

> **Note**
>
> Les groupes de paramètres par défaut sont automatiquement créés à partir d'un modèle de paramètre par défaut lorsque vous créez une instance de base de données pour une version et un moteur de base de données spécifiques. Ces groupes de paramètres par défaut contiennent des valeurs de paramètres préférentielles et ne peuvent pas être modifiés. Lorsque vous créez un groupe de paramètres personnalisé, vous pouvez modifier les réglages des paramètres.

Console

Pour répertorier tous les groupes de paramètres de base de données pour un AWS compte

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.

   Les groupes de paramètres DB s'affichent dans une liste.

AWS CLI

Pour répertorier tous les groupes de paramètres de base de données d'un AWS compte, utilisez la AWS CLI `describe-db-parameter-groups`commande.

Example

L'exemple suivant répertorie tous les groupes de paramètres DB disponibles pour un compte AWS .

```
aws rds describe-db-parameter-groups
```

La commande renvoie une réponse telle que la suivante :

```
DBPARAMETERGROUP  default.mysql8.0     mysql8.0  Default parameter group for MySQL8.0
```

```
DBPARAMETERGROUP  mydbparametergroup   mysql8.0  My new parameter group
```

L'exemple suivant décrit le groupe de paramètres mydbparamgroup1.

Pour Linux, macOS ou Unix :

```
aws rds describe-db-parameter-groups \
    --db-parameter-group-name mydbparamgroup1
```

Pour Windows :

```
aws rds describe-db-parameter-groups ^
    --db-parameter-group-name mydbparamgroup1
```

La commande renvoie une réponse telle que la suivante :

```
DBPARAMETERGROUP  mydbparametergroup1  mysql8.0  My new parameter group
```

API RDS

Pour répertorier tous les groupes de paramètres de base de données d'un AWS compte, utilisez l'DescribeDBParameterGroupsopération d'API RDS.

Affichage des valeurs de paramètres pour un groupe de paramètres de base de données dans Amazon Aurora

Vous pouvez obtenir une liste de tous les paramètres dans un groupe de paramètres DB et de leurs valeurs.

Console

Pour afficher les valeurs de paramètres pour un groupe de paramètres DB

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.

   Les groupes de paramètres DB s'affichent dans une liste.
3. Choisissez le nom du groupe de paramètres pour consulter la liste des paramètres associée.

AWS CLI

Pour afficher les valeurs des paramètres d'un groupe de paramètres de base de données, utilisez la AWS CLI `describe-db-parameters`commande avec le paramètre obligatoire suivant.

- `--db-parameter-group-name`

Example

L'exemple suivant répertorie les paramètres et les valeurs de paramètres pour un groupe de paramètres de base de données nommé mydbparametergroup.

```
aws rds describe-db-parameters --db-parameter-group-name mydbparametergroup
```

La commande renvoie une réponse telle que la suivante :

```
DBPARAMETER  Parameter Name            Parameter Value  Source           Data Type
 Apply Type  Is Modifiable
DBPARAMETER  allow-suspicious-udfs                      engine-default   boolean
 static      false
DBPARAMETER  auto_increment_increment                   engine-default   integer
 dynamic     true
DBPARAMETER  auto_increment_offset                      engine-default   integer
 dynamic     true
DBPARAMETER  binlog_cache_size         32768            system           integer
 dynamic     true
DBPARAMETER  socket                    /tmp/mysql.sock  system           string
 static      false
```

API RDS

Pour afficher les valeurs de paramètre d'un groupe de paramètres de base de données, utilisez la commande d'API RDS `DescribeDBParameters` avec le paramètre requis suivant :

- `DBParameterGroupName`

Suppression d'un groupe de paramètres de base de données dans Amazon Aurora

Vous pouvez supprimer un groupe de paramètres de base de données à l'aide de l'API AWS Management Console AWS CLI, ou RDS. Un groupe de paramètres peut être supprimé uniquement s'il n'est pas associé à une instance de base de données.

Console

Pour supprimer un groupe de paramètres de base de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.

   Les groupes de paramètres DB s'affichent dans une liste.
3. Choisissez le nom des groupes de paramètres à supprimer.
4. Choisissez Actions, puis Supprimer.
5. Vérifiez les noms des groupes de paramètres, puis choisissez Supprimer.

AWS CLI

Pour supprimer un groupe de paramètres de base de données, utilisez la AWS CLI `delete-db-parameter-group`commande avec le paramètre obligatoire suivant.

- `--db-parameter-group-name`

Example

L'exemple suivant illustre la suppression d'un groupe de paramètres de base de données nommé mydbparametergroup.

```
aws rds delete-db-parameter-group --db-parameter-group-name mydbparametergroup
```

API RDS

Pour supprimer un groupe de paramètres de base de données, utilisez la commande `DeleteDBParameterGroup` de l'API RDS avec le paramètre requis suivant.

- `DBParameterGroupName`

Comparaison des groupes de paramètres de bases de données

Vous pouvez utiliser la AWS Management Console pour afficher les différences entre deux groupes de paramètres de base de données.

Les groupes de paramètres doivent tous deux être des groupes de paramètres de base de données, ou bien des groupes de paramètres de cluster de bases de données. Cela est vrai même si le moteur de base de données et la version sont identiques. Par exemple, vous ne pouvez pas comparer un groupe de paramètres de base de données `aurora-mysql8.0` (Aurora MySQL version 3) et un groupe de paramètres de cluster de bases de données `aurora-mysql8.0`.

Vous pouvez comparer des groupes de paramètres de base de données Aurora MySQL et RDS for MySQL, même pour des versions différentes, mais vous ne pouvez pas comparer des groupes de paramètres de base de données Aurora PostgreSQL et RDS pour PostgreSQL.

Pour comparer deux groupes de paramètres de base de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Dans la liste, choisissez les deux groupes de paramètres que vous souhaitez comparer.

   > **Note**
   >
   > Pour comparer un groupe de paramètres par défaut à un groupe de paramètres personnalisé, choisissez d'abord le groupe de paramètres par défaut dans l'onglet Par défaut, puis choisissez le groupe de paramètres personnalisés dans l'onglet Personnalisé.

4. Dans Actions, sélectionnez Comparer.

Spécification des paramètres de base de données

Les types de paramètres de base de données sont les suivants :

- Entier
- Booléen
- Chaîne
- Long
- Double
- Horodatage

- Objet d’autres types de données définis
- Tableau de valeurs de type entier, booléen, chaîne, long, double, horodatage ou objet

Vous pouvez également spécifier des paramètres entiers et booléens au moyen d’expressions, de formules et de fonctions.

Table des matières

Des formules de paramètre de bases de données

Une formule de paramètre de base de données est une expression qui se réduit à une valeur entière ou booléenne. Vous insérez l’expression entre des accolades : {}. Vous pouvez utiliser une formule pour une valeur de paramètre de base de données ou en tant qu’argument pour une fonction de paramètre de base de données.

Syntaxe

```
{FormulaVariable}
{FormulaVariable*Integer}
{FormulaVariable*Integer/Integer}
{FormulaVariable/Integer}
```

Variables de formule de paramètre de bases de données

Chaque variable de formule renvoie une valeur entière ou booléenne. Les noms des variables sont sensibles à la casse.

AllocatedStorage

Renvoie un entier qui représente la taille du volume de données en octets.

DBInstanceClassMemory

Renvoie un entier correspondant au nombre d'octets de mémoire disponibles pour le processus de base de données. Ce nombre est calculé en interne, en commençant par la quantité totale de mémoire pour la classe d'instance de base de données. Le calcul en soustrait la mémoire réservée au système d'exploitation et aux processus RDS qui gèrent l'instance. Par conséquent, ce nombre est toujours légèrement inférieur aux chiffres de mémoire affichés dans les tables de classes d'instance dans Classes d'instance de base de données Amazon Aurora. La valeur exacte dépend d'une combinaison de facteurs. Ils incluent la classe d'instance, le moteur de base de données, et si elle s'applique à une instance RDS ou à une instance faisant partie d'un cluster Aurora.

DBInstanceVCPU

Renvoie un entier qui représente le nombre d'unités de traitement centralisées virtuelles (vCPU) utilisées par Amazon RDS for gérer l'instance.

EndPointPort

Renvoie un entier qui représente le port utilisé lors de la connexion à l'instance de base de données.

TrueIfReplica

Renvoie `1` si l'instance de base de données est un réplica en lecture et `0` si ce n'est pas le cas. Il s'agit de la valeur par défaut du paramètre `read_only` dans Aurora MySQL.

Opérateurs de formule de paramètre de bases de données

Les formules de paramètre DB prennent en charge deux opérateurs : division et multiplication.

Opérateur de division : /

Divise le dividende par le diviseur, en renvoyant un quotient entier. Les décimales dans le quotient sont tronquées, pas arrondies.

Syntaxe

```
dividend / divisor
```

Les arguments de dividende et de diviseur doivent être des expressions entières.

Opérateur de multiplication : *

Multiplie les expressions, affichant ainsi le résultat des expressions. Les décimales dans les expressions sont tronquées, pas arrondies.

Syntaxe

```
expression * expression
```

Les deux expressions doivent être des entiers.

Fonctions de paramètre de bases de données

Vous spécifiez les arguments des fonctions de paramètres de base de données sous la forme d'entiers ou de formules. Chaque fonction doit avoir au moins un argument. Spécifiez plusieurs arguments sous la forme d'une liste séparée par des virgules. Cette liste ne peut pas contenir de membres vides, tels que argument1,,argument3. Les noms de fonctions ne sont pas sensibles à la casse.

IF

Renvoie un argument.

Syntaxe

```
IF(argument1, argument2, argument3)
```

Renvoie le deuxième argument si le premier a la valeur true. Sinon, renvoie le troisième argument.

GREATEST

Renvoie la plus grande valeur depuis une liste d'entiers ou de formules de paramètres.

Syntaxe

```
GREATEST(argument1, argument2,...argumentn)
```

Renvoie un entier.

LEAST

Renvoie la plus petite valeur depuis une liste d'entiers ou de formules de paramètres.

Syntaxe

```
LEAST(argument1, argument2,...argumentn)
```

Renvoie un entier.

SUM

Ajoute les valeurs des formules de paramètres ou d'entiers spécifiés.

Syntaxe

```
SUM(argument1, argument2,...argumentn)
```

Renvoie un entier.

Expressions de journal des paramètres de base de données

Vous pouvez définir une valeur de paramètre de base de données entier à une expression de journal. Vous insérez l'expression entre des accolades : {}. Exemples :

```
{log(DBInstanceClassMemory/8187281418)*1000}
```

La fonction `log` représente la base du journal 2. Cet exemple utilise également la variable de formule `DBInstanceClassMemory`. Consultez Variables de formule de paramètre de bases de données.

Exemples de valeurs de paramètre de bases de données

Ces exemples montrent l'utilisation de formules, de fonctions et d'expressions pour les valeurs des paramètres de base de données.

> **Warning**
>
> La définition incorrecte des paramètres d'un groupe de paramètres de base de données peut avoir des effets indésirables involontaires. Cela peut se manifester par une dégradation des performances et l'instabilité du système. Agissez avec prudence lorsque vous modifiez

des paramètres de base de données et sauvegardez vos données avant de modifier votre groupe de paramètres de base de données. Testez les modifications d'un groupe de paramètres sur une instance de base de données test, créée en utilisant des restaurations à un moment donné, avant d'appliquer ces modifications à vos instances de base de données de production.

Example utilisation de la fonction de paramètre de base de données LEAST

Vous pouvez spécifier la fonction `LEAST` dans une valeur de paramètre Aurora MySQL `table_definition_cache`. Utilisez-la pour définir le nombre de définitions de table qui peuvent être stockées dans le cache de définitions à la moins élevée des valeurs suivantes : `DBInstanceClassMemory`/393040 ou 20 000.

```
LEAST({DBInstanceClassMemory/393040}, 20000)
```

Migration de données vers un cluster de bases de données Amazon Aurora

Vous avez plusieurs options pour la migration des données depuis votre base de données existante vers un cluster de bases de données Amazon Aurora, en fonction de la compatibilité du moteur de base de données. Vos options de migration dépendent également de la base de données à partir de laquelle vous effectuez la migration et de la taille des données que vous migrez.

Migration de données vers un cluster de bases de données Amazon Aurora MySQL

Vous pouvez migrer les données depuis l'une des sources suivantes vers un cluster de bases de données Amazon Aurora MySQL.

- Une instance de base de données RDS pour MySQL
- Une base de données MySQL externe à Amazon RDS
- Une base de données qui n'est pas compatible avec MySQL

Pour plus d'informations, consultez Migration de données vers un cluster de bases de données Amazon Aurora MySQL.

Migration de données vers un cluster de bases de données Amazon Aurora PostgreSQL

Vous pouvez migrer les données depuis l'une des sources suivantes vers un cluster de bases de données Amazon Aurora PostgreSQL.

- Une instance de base de données Amazon RDS PostgreSQL
- Une base de données qui n'est pas compatible avec PostgreSQL

Pour plus d'informations, consultez Migration des données vers Amazon Aurora avec compatibilité PostgreSQL.

Création d’un cache Amazon ElastiCache à l’aide des paramètres d’un cluster de bases de données Aurora

ElastiCache est un service de mise en cache en mémoire entièrement géré qui fournit des latences de lecture et d’écriture en microsecondes qui prennent en charge des cas d’utilisation flexibles en temps réel. ElastiCache peut vous aider à améliorer les performances des applications et des bases de données. Vous pouvez utiliser ElastiCache comme magasin de données principal pour les cas d’utilisation qui ne nécessitent pas de durabilité des données, tels que les classements de jeux, le streaming et l’analyse de données. ElastiCache permet de supprimer la complexité associée au déploiement et à la gestion d’un environnement de calcul distribué. Pour plus d’informations, consultez Cas d’utilisation courants d’ElastiCache et ce que peut vous apporter ElastiCache pour Memcached et Cas d’utilisation courants d’ElastiCache et ce que peut vous apporter ElastiCache pour Redis OSS. Vous pouvez utiliser la console Amazon RDS pour créer un cache ElastiCache.

Vous pouvez utiliser Amazon ElastiCache dans deux formats. Vous pouvez commencer avec un cache sans serveur ou choisir de concevoir votre propre cluster de cache. Si vous choisissez de concevoir votre propre cluster de cache, ElastiCache prend en charge les moteurs de cache Memcached et Redis OSS. Si vous ne savez pas quel moteur vous souhaitez utiliser, consultez Comparaison de Memcached et Redis OSS. Pour plus d’informations sur Amazon ElastiCache, consultez le Guide de l’utilisateur Amazon ElastiCache.

Rubriques

Présentation de la création d’un cache ElastiCache avec les paramètres d’un cluster de bases de données Aurora

Vous pouvez créer un cache ElastiCache à partir d’Amazon RDS en utilisant les mêmes paramètres de configuration qu’un cluster de bases de données Aurora existant ou que vous venez de créer.

Voici quelques cas d’utilisation permettant d’associer un cache ElastiCache à votre cluster de bases de données :

- Vous pouvez réduire les coûts et améliorer vos performances en utilisant ElastiCache avec RDS plutôt que de l’exécuter uniquement sur RDS.

- Vous pouvez utiliser le cache ElastiCache comme magasin de données principal pour les applications qui n'ont pas besoin de durabilité des données. Vos applications existantes qui utilisent Redis OSS ou Memcached peuvent utiliser ElastiCache sans pratiquement aucune modification.

Lorsque vous créez un cache ElastiCache à partir de RDS, il hérite des paramètres suivants du cluster de bases de données Aurora associé :

- Paramètres de connectivité d'ElastiCache
- Paramètres de sécurité d'ElastiCache

Vous pouvez aussi définir les paramètres de configuration du cache en fonction de vos besoins.

Configuration d'ElastiCache dans vos applications

Vos applications doivent être configurées pour utiliser le cache ElastiCache. Vous pouvez également optimiser et améliorer les performances du cache en configurant vos applications pour qu'elles utilisent des stratégies de mise en cache en fonction de vos besoins.

- Pour accéder à votre cluster ElastiCache et démarrer, consultez Mise en route avec Amazon ElastiCache (Redis OSS) et Mise en route avec Amazon ElastiCache (Memcached).
- Pour plus d'informations sur les stratégies de mise en cache, consultez Stratégies de mise en cache et bonnes pratiques pour Memcached et Stratégies de mise en cache et bonnes pratiques pour Redis OSS.
- Pour plus d'informations sur la haute disponibilité dans les clusters ElastiCache (Redis OSS), consultez Haute disponibilité avec les groupes de réplication.
- Vous pouvez encourir des frais liés au stockage des sauvegardes, au transfert de données au sein ou entre les régions, ou à l'utilisation d'AWS Outposts. Pour plus de détails, consultez Tarification Amazon ElastiCache.

Création d'un cache ElastiCache avec les paramètres d'un cluster de bases de données Aurora

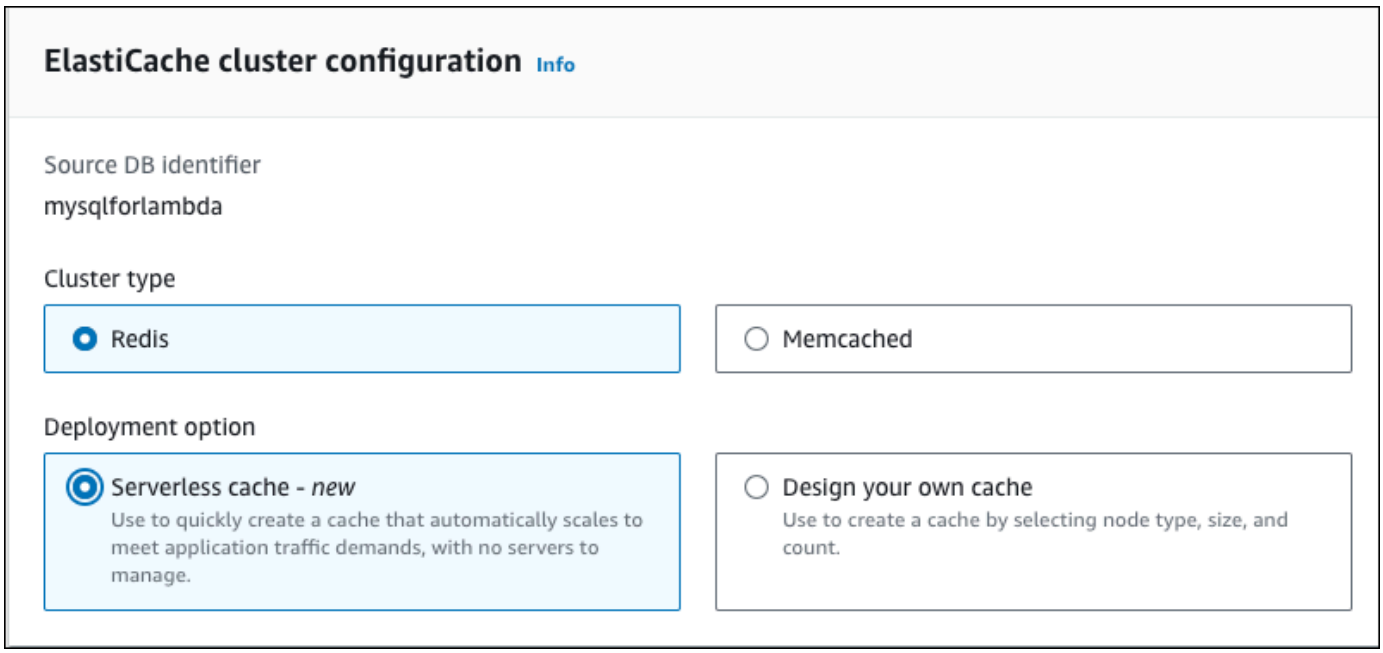
Vous pouvez créer un cache ElastiCache pour vos clusters de bases de données Aurora avec les paramètres hérités du cluster de bases de données.

Création d'un cache ElastiCache avec les paramètres d'un cluster de bases de données

1. Pour créer un cluster de bases de données, suivez les instructions de Création d'un cluster de bases de données Amazon Aurora.
2. Après avoir créé un cluster de bases de données Aurora, la console affiche la fenêtre Modules complémentaires suggérés. Sélectionnez Créer un cluster ElastiCache à partir de RDS à l'aide de vos paramètres de base de données.

   Pour une base de données existante, sur la page Bases de données, sélectionnez le cluster de bases de données nécessaire. Dans le menu déroulant Actions, choisissez Créer un cluster ElastiCache pour créer un cache ElastiCache dans RDS doté des mêmes paramètres que votre cluster de bases de données Aurora existant.

   Dans la section Configuration d'ElastiCache, l'option Identifiant de la base de données source indique le cluster de bases de données dont le cache ElastiCache hérite des paramètres.
3. Choisissez si vous voulez créer un cluster Redis OSS ou Memcached. Pour plus d'informations, consultez Comparaison de Memcached et Redis OSS.



4. Ensuite, choisissez si vous souhaitez créer un cache sans serveur ou concevoir votre propre cache. Pour plus d'informations, consultez Choix entre deux options de déploiement.

   Si vous choisissez le cache sans serveur :

   a. Dans Paramètres du cache, entrez des valeurs pour Nom et Description.

b. Sous Afficher les paramètres par défaut, conservez les paramètres par défaut pour établir la connexion entre le cache et le cluster de bases de données.

c. Vous pouvez également modifier les paramètres par défaut en choisissant Personnaliser les paramètres par défaut. Sélectionnez les paramètres de connectivité ElastiCache, les paramètres de sécurité ElastiCache et les limites d'utilisation maximales.

5. Si vous choisissez Créez votre propre cache :

a. Si vous choisissez Cluster Redis OSS, indiquez si vous souhaitez conserver le mode cluster Activé ou Désactivé. Pour plus d'informations, consultez Réplication : Redis OSS (mode cluster désactivé) vs Redis OSS (mode cluster activé).

b. Saisissez des valeurs pour Nom, Description et Version du moteur.

Pour Version du moteur, la valeur par défaut recommandée est la dernière version du moteur. Vous pouvez également choisir une version du moteur pour le cache ElastiCache qui répond le mieux à vos besoins.

c. Choisissez le type de nœud dans l'option Type de nœud. Pour plus d'informations, consultez Gestion des nœuds.

Si vous choisissez de créer un cluster Redis OSS avec le mode Cluster défini sur Activé, saisissez le nombre de partitions (partitions/groupes de nœuds) dans l'option Nombre de partitions.

Saisissez le nombre de réplicas de chaque partition dans Nombre de réplicas.

> **Note**
>
> Le type de nœud sélectionné, le nombre de partitions et le nombre de réplicas influent tous sur les performances et les coûts de ressources de votre cache. Veillez à ce que ces paramètres correspondent aux besoins de votre base de données. Pour plus d'informations sur la tarification, consultez Tarification Amazon ElastiCache.

d. Sélectionnez les paramètres de connectivité ElastiCache et les paramètres de sécurité ElastiCache. Vous pouvez conserver les paramètres par défaut ou les personnaliser selon vos besoins.

6. Vérifiez les paramètres par défaut et ceux hérités de votre cache ElastiCache. Certains paramètres ne peuvent pas être modifiés après la création.

> **Note**
>
> RDS peut ajuster la fenêtre de sauvegarde de votre cache ElastiCache pour respecter la période minimale requise de 60 minutes. La fenêtre de sauvegarde de votre base de données source reste la même.

7. Lorsque vous êtes prêt, choisissez Créer un cache ElastiCache.

La console affiche une bannière de confirmation de la création du cache ElastiCache. Suivez le lien figurant dans la bannière vers la console ElastiCache pour consulter les détails du cache. La console ElastiCache affiche le cache ElastiCache que vous venez de créer.

Migration automatique de bases de données EC2 vers Amazon Aurora à l'aide de AWS Database Migration Service

Vous pouvez utiliser la console Aurora pour migrer une base de données EC2 vers Aurora. Aurora utilise AWS Database Migration Service (AWS DMS) pour migrer votre base de données EC2 source. AWS DMS vous permet de migrer des bases de données relationnelles dans votre Cloud AWS. Pour plus d'informations sur AWS Database Migration Service, consultez Qu'est-ce que AWS Database Migration Service ? dans le guide de l'utilisateur AWS Database Migration Service.

Pour commencer la migration, vous devez créer un cluster de bases de données Aurora équivalent dans lequel migrer les données . Après avoir créé votre base de données cible, vous pouvez y importer votre base de données EC2. Pour les bases de données source inférieures à 1 Tio, cette action de migration réduit le temps et les ressources nécessaires à la migration de vos données dans Aurora .

Présentation

La console Aurora vous permet de migrer des bases de données EC2 vers des bases de données Aurora équivalentes. Vous devez créer une base de données Aurora pour permettre la migration depuis la console.

Vous pouvez migrer des bases de données EC2 pour les moteurs de base de données suivants :

- MySQL
- PostgreSQL

Le processus de migration englobe les étapes suivantes :

- Créez une base de données équivalente dans Aurora. Pour que les bases de données soient équivalentes, elles doivent avoir le même moteur de base de données et des versions de moteur compatibles. Elles doivent également se trouver dans le même VPC. Pour obtenir des instructions sur la création de votre base de données, consultez Création d'un cluster de bases de données Amazon Aurora .
- Choisissez le type de réplication pour votre base de données :
  - Migration à charge complète : Aurora copie l'intégralité de la base de données source vers la base de données cible, en créant de nouvelles tables dans la cible si nécessaire.

> **Note**
>
> Cette option entraîne une panne de votre base de données Aurora.

- Migration à chargement complet et capture des données (CDC) : similaire à la migration à chargement complet, avec cette option, Aurora copie l'intégralité de la base de données source vers la base de données cible. Toutefois, après la migration à chargement complet, Aurora applique les modifications capturées dans la source à la base de données cible. La capture des données de modification collecte les journaux de base de données à l'aide de l'API native du moteur de base de données.

  > **Note**
  >
  > Cette option entraîne une panne de votre base de données Aurora.

- Capture des données de modification (CDC) : utilisez cette option pour que votre base de données cible reste disponible pendant la migration. Aurora migre les modifications en cours dans votre base de données source vers la base de données cible.

- Aurora crée les ressources réseau nécessaires pour faciliter la migration. Une fois qu'Aurora a créé les ressources requises, elle vous informe des ressources créées et vous permet de lancer le transfert de données.

  Le temps nécessaire pour effectuer la migration dépend du type de réplication et de la taille de la base de données source.

Prérequis

MySQL

Avant de commencer à utiliser une base de données MySQL comme base de données source, veillez à ce que les conditions préalables suivantes soient remplies. Ces prérequis s'appliquent aux sources gérées par AWS.

Vous devez avoir un compte pour AWS DMS doté du rôle d'administrateur de réplication. Ce rôle nécessite les privilèges suivants :

- REPLICATION CLIENT : ce privilège est obligatoire pour les tâches de CDC uniquement. En d'autres termes, les tâches de chargement complet uniquement ne requièrent pas ce privilège.
- REPLICATION SLAVE : ce privilège est obligatoire pour les tâches de CDC uniquement. En d'autres termes, les tâches de chargement complet uniquement ne requièrent pas ce privilège.

L'utilisateur AWS DMS doit également disposer des privilèges SELECT pour les tables sources désignées pour la réplication.

Accordez les privilèges suivants si vous utilisez des évaluations de prémigration spécifiques à MySQL.

```
grant select on mysql.user to <dms_user>;
grant select on mysql.db to <dms_user>;
grant select on mysql.tables_priv to <dms_user>;
grant select on mysql.role_edges to <dms_user>  #only for MySQL version 8.0.11 and
 higher
```

PostgreSQL

Avant de migrer des données à partir d'une base de données source PostgreSQL gérée par AWS, procédez comme suit :

- Nous vous recommandons d'utiliser un compte d'utilisateur AWS avec les autorisations minimales requises pour l'instance de base de données PostgreSQL en tant que compte d'utilisateur pour le point de terminaison source PostgreSQL pour AWS DMS. L'utilisation du compte principal n'est pas recommandée. Le compte doit avoir le rôle `rds_superuser` et le rôle `rds_replication`. Le rôle `rds_replication` accorde les autorisations permettant de gérer des emplacements logiques et de diffuser les données à l'aide d'emplacements logiques.

> **Note**
>
> Certaines transactions AWS DMS sont inactives un certain temps avant que le moteur DMS ne les utilise à nouveau. En utilisant le paramètre `idle_in_transaction_session_timeout` dans PostgreSQL versions 9.6 et ultérieures, vous pouvez provoquer l'expiration et l'échec des transactions inactives.

Limitations

Les limitations suivantes s’appliquent au processus de migration automatique :

- Le statut de votre base de données cible doit être Disponible pour commencer la migration de la base de données source.
- Lorsque vous migrez depuis une base de données source MySQL, votre compte Aurora doit avoir le rôle d’administrateur de réplication. Vous devez également avoir les privilèges appropriés pour ce rôle.
- Votre instance EC2 et votre base de données cible doivent être dans le même VPC.
- Vous ne pouvez pas migrer votre base de données EC2 vers les bases de données cibles suivantes lorsque vous utilisez l’action Migrer les données de la base de données EC2 :
  - Aurora global database
  - Aurora Limitless database
  - Aurora Serverless v1
  - Bases de données avec MySQL version inférieure à 5.7
  - Bases de données avec PostgreSQL version inférieure à 10.4

Création des ressources IAM pour les migrations homogènes

Aurora utilise AWS DMS pour migrer vos données. Pour accéder à vos bases de données et migrer les données, AWS DMS créez un environnement sans serveur pour des migrations de données homogènes. Dans cet environnement, AWS DMS nécessite un accès au peering VPC, aux tables de routage, aux groupes de sécurité et à d'autres ressources. AWS AWS DMSStocke également les journaux, les statistiques et la progression de chaque migration de données sur Amazon CloudWatch. Pour créer un projet de migration de données, il AWS DMS faut avoir accès à ces services.

AWS DMSNécessite également l'accès aux secrets qui représentent un ensemble d'informations d'identification utilisateur pour authentifier la connexion à la base de données pour les connexions source et cible.

> **Note**
>
> En utilisant l'action Migrer les données depuis l' EC2 instance, vous pouvez utiliser la console Aurora pour générer ces ressources IAM. Ignorez cette étape si vous utilisez les ressources IAM générées par la console.

Pour cette procédure, vous avez besoin des ressources IAM suivantes :

Rubriques

- Création d’une politique IAM pour les migrations de données homogènes
- Création d’un rôle IAM pour les migrations de données homogènes
- Création d’une stratégie d’accès secrète et d’un rôle
- Création d’un rôle IAM pour AWS DMS pour gérer Amazon VPC

Création d’une politique IAM pour les migrations de données homogènes

Au cours de cette étape, vous créez une politique IAM qui donne accès AWS DMS à Amazon EC2 et à ses CloudWatch ressources. Ensuite, créez un rôle IAM et attachez cette politique.

Pour créer une politique IAM pour la migration de données

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le panneau de navigation, choisissez Politiques.
3. Choisissez Create Policy (Créer une politique).
4. Sur la page Créer une politique, choisissez l’onglet JSON.
5. Collez le code JSON suivant dans l’éditeur.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeVpcPeeringConnections",
                "ec2:DescribeVpcs",
                "ec2:DescribePrefixLists",
                "logs:DescribeLogGroups"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "servicequotas:GetServiceQuota"
            ],
            "Resource": "arn:aws:servicequotas:*:*:vpc/L-0EA8095F"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:dms-data-migration-*:log-
stream:dms-data-migration-*"
        },
        {
            "Effect": "Allow",
            "Action": "cloudwatch:PutMetricData",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateRoute",
                "ec2:DeleteRoute"
            ],
```

```
            "Resource": "arn:aws:ec2:*:*:route-table/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:security-group-rule/*",
                "arn:aws:ec2:*:*:route-table/*",
                "arn:aws:ec2:*:*:vpc-peering-connection/*",
                "arn:aws:ec2:*:*:vpc/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group-rule/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress"
            ],
            "Resource": "arn:aws:ec2:*:*:security-group/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AcceptVpcPeeringConnection",
                "ec2:ModifyVpcPeeringConnectionOptions"
            ],
            "Resource": "arn:aws:ec2:*:*:vpc-peering-connection/*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:AcceptVpcPeeringConnection",
```

```
            "Resource": "arn:aws:ec2:*:*:vpc/*"
        }
    ]
}
```

6. Choisissez Next: Tags (Suivant : Balises), puis Next: Review (Suivant : Vérification).
7. Entrez **`HomogeneousDataMigrationsPolicy`** pour Nom*, puis choisissez Créer une politique.

Création d'un rôle IAM pour les migrations de données homogènes

Au cours de cette étape, vous créez un rôle IAM qui donne accès à AWS Secrets Manager Amazon EC2 et CloudWatch.

Pour créer un rôle IAM pour les migrations de données

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Dans le panneau de navigation, choisissez Rôles.
3. Sélectionnez Create role (Créer un rôle).
4. Sur la page Sélectionner une entité de confiance, pour Type d'entité approuvée, choisissez Service AWS. Pour Cas d'utilisation d'autres services AWS, choisissez DMS.
5. Cochez la case DMS et choisissez Suivant.
6. Sur la page Ajouter des autorisations, choisissez HomogeneousDataMigrationsPolicycelle que vous avez créée auparavant. Choisissez Suivant.
7. Sur la page Nommer, vérifier et créer, entrez **`HomogeneousDataMigrationsRole`** pour Nom du rôle et choisissez Créer un rôle.
8. Sur la page Rôles, entrez **`HomogeneousDataMigrationsRole`** pour Nom du rôle. Sélectionnez HomogeneousDataMigrationsRole.
9. Sur la HomogeneousDataMigrationsRolepage, choisissez l'onglet Relations de confiance. Choisissez Modifier la politique d'approbation.
10. Sur la page Modifier la politique d'approbation, collez le code JSON suivant dans l'éditeur en remplaçant le texte existant.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "dms-data-migrations.amazonaws.com",
                    "dms.your_region.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Dans l'exemple précédent, remplacez *your_region* par le nom de votreRégion AWS.

La politique basée sur les ressources précédente donne aux responsables de AWS DMS service les autorisations nécessaires pour effectuer des tâches conformément à la politique gérée par le client. HomogeneousDataMigrationsPolicy

11. Choisissez Mettre à jour une politique.

Création d’une stratégie d’accès secrète et d’un rôle

Suivez les procédures ci-dessous pour créer votre stratégie d’accès secrète et votre rôle permettant à DMS d’accéder aux informations d’identification utilisateur de vos bases de données source et cible.

Pour créer la politique et le rôle d'accès secret, qui permettent à Amazon RDS d'accéder AWS Secrets Manager à votre secret approprié

1. Connectez-vous à la console Gestion des identités et des accès AWS (IAM) AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Sélectionnez Politiques, puis Créer une politique.

3. Choisissez JSON et entrez la politique suivante pour permettre d'accéder à votre secret et de le déchiffrer.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-1:111122223333:secret:SecretName-ABCDEF"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
    ]
}
```

   *secret_arn*Voici l'ARN de votre secret, que vous pouvez obtenir de l'un ou l'autre `SecretsManagerSecretId` selon le cas, et *kms_key_arn* l'ARN de la AWS KMS clé que vous utilisez pour chiffrer votre secret, comme dans l'exemple suivant.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-
east-2:123456789012:secret:MySQLTestSecret-qeHamH"
```

```
        },
        {
             "Effect": "Allow",
             "Action": [
                       "kms:Decrypt",
                       "kms:DescribeKey"
                      ],
             "Resource": "arn:aws:kms:us-
east-2:123456789012:key/761138dc-0542-4e58-947f-4a3a8458d0fd"
        }
    ]
}
```

> **Note**
>
> Si vous utilisez la clé de chiffrement par défaut créée parAWS Secrets Manager, il n'est pas nécessaire de spécifier les AWS KMS autorisations pour*kms_key_arn*.
> Si vous souhaitez que votre politique donne accès aux deux secrets, il vous suffit de spécifier un objet de ressource JSON supplémentaire pour l'autre*secret_arn*.

4. Vérifiez et créez la politique avec un nom convivial et une description facultative.
5. Choisissez Rôles, puis Créer un rôle.
6. Choisissez Service AWS comme type d'entité de confiance.
7. Choisissez DMS dans la liste des services comme service de confiance, puis choisissez Suivant : Autorisations.
8. Recherchez et attachez la politique que vous avez créée à l'étape 4, puis ajoutez des balises et passez en revue votre rôle. À ce stade, modifiez les relations d'approbation du rôle afin d'utiliser votre principal de service régional Amazon RDS comme entité de confiance. Ce principal a le format suivant.

```
dms.region-name.amazonaws.com
```

Ici, *region-name* est le nom de votre région, par exemple `us-east-1`. Il est suivi par un principal de service régional Amazon RDS pour cette région.

```
dms.us-east-1.amazonaws.com
dms-data-migrations.amazonaws.com
```

Création d’un rôle IAM pour AWS DMS pour gérer Amazon VPC

Vous devez créer un rôle IAM pour AWS DMS pour gérer les paramètres VPC de vos ressources. Ce rôle doit être disponible pour que la migration soit réussie.

Création du **`dms-vpc-role`** pour la migration de base de données

<result>

Cela crée le rôle permettant au DMS de gérer les paramètres VPC pour la migration.
</result>

1. Connectez-vous à la AWS Management Console, puis ouvrez la console IAM à l’adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation de la console, choisissez Rôles, puis Créer un rôle.
3. Choisissez l’option Service AWS pour l’option Sélectionner une entité de confiance.

   Pour Cas d’utilisation, sélectionnez DMS.
4. Pour l’étape Ajouter des autorisations, sélectionnez `AmazonDMSVPCManagementRole` et choisissez Suivant.
5. Sur la page Nommer, vérifier et créer, définissez le Nom du rôle sur `dms-vpc-role` et choisissez Créer un rôle.

Configuration de la migration des données pour la base de données EC2

Pour commencer à migrer des données depuis votre base de données source EC2, vous devez créer une base de données Aurora équivalente. Pour obtenir des instructions sur la création de votre base de données, consultez Création d’un cluster de bases de données Amazon Aurora .

Après avoir créé votre base de données cibles, procédez comme suit pour configurer la migration des données :

Configuration d’un projet de migration des données

1. Sélectionnez la base de données cible sur la page Bases de données de la console RDS.
2. Choisissez le menu déroulant Actions et sélectionnez l’option Migrer les données depuis la base de données EC2. Pour obtenir la liste des bases de données cibles prises en charge, consultez Limitations.
3. Dans la section Sélectionner la base de données EC2 source :

1. Vérifiez le type de moteur et assurez-vous qu'il est identique à celui de votre base de données source.

   Vérifiez également si les versions du moteur sont compatibles.
2. Pour Instance EC2, choisissez l'instance EC2 où réside votre base de données source.
3. Pour Port, entrez le port sur lequel votre base de données source autorise le trafic.
4. Pour Secret, choisissez Créer et utiliser un nouveau secret si vous n'en avez pas déjà un. Ajoutez le Nom d'utilisateur et le Mot de passe de votre base de données source. Choisissez également la clé KMS avec laquelle chiffrer votre secret.

   Si vous utilisez un secret existant, sélectionnez Utiliser un secret existant puis choisissez un secret dans la liste déroulante.
5. Pour rôle IAM pour le secret, si vous avez un rôle IAM existant, sélectionnez Utiliser un rôle IAM existant et choisissez un rôle AM dans la liste déroulante qui peut accéder à l'ID secret de l'étape précédente.

   Si vous ne disposez pas de rôle IAM, choisissez Créer et utiliser un nouveau rôle IAM. Pour Nom du rôle IAM, entrez un nouveau nom pour votre rôle. Vous pouvez consulter les autorisations associées à ce rôle dans le lien ci-dessous.

4. Dans la section Afficher la base de données RDS cible :

   1. Confirmez les paramètres de votre base de données cible en haut de la section.
   2. Pour Secret, choisissez Créer et utiliser un nouveau secret si vous n'en avez pas déjà un qui comporte les informations d'identification de votre base de données cible.

      Si vous utilisez un secret existant, sélectionnez-le dans la liste déroulante.
   3. Pour Rôle IAM pour le secret, sélectionnez un rôle IAM qui peut accéder au secret depuis l'étape précédente. Vous pouvez également créer un nouveau rôle IAM si vous n'avez pas de rôle IAM existant.

      Si la liste déroulante ne contient pas les rôles IAM, spécifiez l'ARN du rôle IAM au format `arn:aws:iam:`*`account_id`*`:role/`*`roleName`*.

5. Dans la section Configurer la migration des données :

   1. Sélectionnez le type de migration des données en choisissant entre Chargement complet, Capture des données de chargement complet et de modification (CDC) ou Capture de données modifiées (CDC). Pour plus d'informations sur ces options, consultez Présentation.

Vous ne pouvez pas modifier le type de migration une fois celle-ci lancée.

2. Pour Rôle IAM pour la migration des données, si vous avez un rôle IAM existant, sélectionnez Utiliser un rôle IAM existant et choisissez un rôle IAM dans la liste déroulante qui accorde à DMS les autorisations nécessaires pour créer les ressources nécessaires à la migration. Si vous ne disposez pas de rôle IAM, choisissez Créer et utiliser un nouveau rôle IAM.

6. Vérifiez que l'onglet Afficher les paramètres de migration affiche les paramètres requis pour que votre migration de données soit correctement configurée.
7. Sélectionnez Migrer pour terminer la configuration de la migration.

Une fois ces étapes terminées, vous pouvez voir les ressources en cours de configuration pour la migration des données en choisissant Afficher les détails dans la bannière de progression de la console. Une fois les ressources requises configurées, la migration démarre automatiquement. Si vous créez

Pour migrer plusieurs bases de données vers la base de données cible, recommencez ce processus avec des informations sur la nouvelle base de données EC2.

Gestion des migrations de données

Après avoir utilisé l'action Migrer les données depuis la base de données EC2 depuis la console RDS, Aurora démarre automatiquement la migration.

Si vous avez utilisé la console AWS DMS pour créer les ressources de migration, vous pouvez démarrer le processus de migration.

Démarrage de la migration des données

Pour démarrer la migration des données, procédez comme suit :

Démarrage d'une migration des données

1. Choisissez la base de données cible sur la page Bases de données de la console RDS.
2. Sur la page de détails de la base de données, sélectionnez l'onglet Migrations de données.
3. Sous l'onglet Migrations de données, Migrations de données associées répertorie les migrations de données disponibles.

   Les migrations configurées à l'aide de la console Aurora démarrent automatiquement une fois que les ressources requises sont configurées.

Les migrations configurées à l’aide de la console DMS sont définies sur Prêt.

Pour commencer ces migrations, sélectionnez le menu déroulant Actions, puis sélectionnez Démarrer.

4. Cela commence la migration des données pour votre base de données EC2.

Arrêt de la migration des données

Pour les migrations de données dont le type de réplication est à pleine charge, l’arrêt de la migration entraîne l’arrêt du processus et ne peut pas être repris. Une fois la migration arrêtée, vous devez redémarrer la migration.

Pour les migrations dont le type de réplication est défini pour modifier la capture des données (CDC) ou pour le chargement complet et le CDC, vous pouvez arrêter le processus de réplication continue et le reprendre ultérieurement.

Arrêt d’une migration des données

1. Choisissez la base de données cible sur la page Bases de données de la console RDS.
2. Sur la page de détails de la base de données, sélectionnez l’onglet Migrations de données.
3. Sous l’onglet Migrations de données, Migrations de données associées répertorie les migrations de données en cours.

   Pour arrêter une migration, sélectionnez une migration de données, puis sélectionnez Arrêter dans le menu déroulant Actions.
4. Cela arrête la migration des données pour votre base de données EC2.

Reprise de la migration des données

Pour les migrations de données dont le type de réplication est Capture des données de chargement complet et de modification (CDC) ou Capture de données modifiées (CDC), vous pouvez reprendre le processus CDC depuis le dernier arrêt.

Reprise d’une migration des données

1. Choisissez la base de données cible sur la page Bases de données de la console RDS.
2. Sur la page de détails de la base de données, sélectionnez l’onglet Migrations de données.

3. Sous l’onglet Migrations de données, Migrations de données associées répertorie les migrations de données arrêtées.

   Pour reprendre une migration, sélectionnez une migration de données, puis sélectionnez Reprendre le traitement dans le menu déroulant Actions.
4. Cela reprend la migration des données pour votre base de données EC2.

Suppression de la migration des données

Pour supprimer une migration de données associée, suivez les instructions suivantes

Suppression d’une migration des données

1. Choisissez la base de données cible sur la page Bases de données de la console RDS.
2. Sur la page de détails de la base de données, sélectionnez l’onglet Migrations de données.
3. Pour supprimer une migration, sélectionnez une migration de données, puis sélectionnez Supprimer dans le menu déroulant Actions.
4. Cela supprime la migration des données.

La suppression d’une migration de données en cours n’a aucune incidence sur les données déjà chargées dans la base de données cible.

Redémarrage de la migration des données

Pour redémarrer une migration de données associée depuis un point de départ CDC, suivez les instructions suivantes

Redémarrage d’une migration des données

1. Choisissez la base de données cible sur la page Bases de données de la console RDS.
2. Sur la page de détails de la base de données, sélectionnez l’onglet Migrations de données.
3. Pour redémarrer une migration, sélectionnez une migration de données, puis sélectionnez Redémarrer dans le menu déroulant Actions.
4. Cela redémarre la migration des données à partir d’un point de départ du CDC.

Le redémarrage d’une migration de données en cours n’a aucune incidence sur les données déjà chargées dans la base de données cible.

Surveillance des migrations de données

Une fois les migrations de données démarrées, vous pouvez surveiller leur statut et leur progression. Les migrations de jeux de données de grande taille prennent des heures. Pour gérer la fiabilité, la disponibilité et les performances de votre migration de données, surveillez régulièrement sa progression.

Pour vérifier le statut et la progression de votre migration de données

1. Choisissez la base de données cible sur la page Bases de données de la console RDS.
2. Sur la page de détails de la base de données, sélectionnez l'onglet Migrations de données.
3. La section Migrations de données associées répertorie vos migrations de données. Vérifiez la colonne Statut.
4. Pour les migrations de données en cours, la colonne Processus de migration affiche le pourcentage de données migrées.
5. Pour surveiller le processus dans CloudWatch, utilisez le lien figurant dans la colonne CloudWatch.

Statuts de migration

Pour chaque migration de données que vous exécutez, la console Aurora affiche le Statut. La liste suivante inclut les statuts :

- `Ready` : la migration de données est prête à démarrer.
- `Starting` : Aurora crée l'environnement sans serveur pour votre migration de données.
- `Load running` : Aurora effectue la migration de chargement complet.
- `Load complete, replication ongoing` : Aurora a terminé le chargement complet et réplique désormais les modifications en cours. Ce statut s'applique uniquement aux migrations à chargement complet et aux migrations de type CDC.
- `Replication ongoing` : Aurora réplique les modifications en cours. Ce statut s'applique uniquement aux migrations de type CDC.
- `Stopping` : Aurora arrête les migrations de données. Ce statut s'applique lorsque vous choisissez d'arrêter la migration des données à partir du menu Actions.
- `Stopped` : Aurora a arrêté les migrations de données.

- `Failed` : la migration de données a échoué. Pour plus d’informations, consultez les fichiers journaux.
- `Restarting` : la migration des données a relancé une réplication de données en cours depuis un point de départ CDC.

Didacticiel : Création d'un cluster de bases de données à l'aide d'un groupe de paramètres personnalisé

Dans ce didacticiel, vous créez un cluster de bases de données MySQL à l'aide d'un groupe de paramètres personnalisé. Pour plus d'informations sur les groupes de paramètres et les groupes , consultez Groupes de paramètres de cluster de bases de données pour les clusters de bases de données Amazon Aurora.

> **Important**
>
> Il n'y a pas de frais pour la création d'un compte AWS. Toutefois, au cours de ce didacticiel, des coûts peuvent être générés par l'utilisation des ressources AWS. Vous pouvez supprimer ces ressources après avoir terminé le didacticiel si elles ne sont plus nécessaires.

Pour créer un cluster de bases de données avec des configurations et des paramètres personnalisés, vous pouvez utiliser un groupe de paramètres personnalisé . Les groupes de paramètres personnalisés sont particulièrement utiles si vous travaillez avec plusieurs bases de données et souhaitez configurer leurs paramètres de manière uniforme.

En suivant ces étapes, vous apprendrez à :

- Comment utiliser Amazon Aurora pour créer un cluster de bases de données à l'aide d'un groupe de paramètres personnalisé .
- Comment utiliser des paramètres spécifiques pour des clusters de bases de données MySQL.

Pour suivre ce didacticiel, effectuez les tâches suivantes :

1. Créez un groupe de paramètres de cluster de bases de données à l'aide du paramètre MySQL `default_password_lifetime`.
2. Créez un cluster de bases de données MySQL à l'aide d'un groupe de paramètres de cluster de bases de données personnalisé que vous avez créés.

Rubriques

- Modification de la valeur de paramètre dans votre groupe de paramètres personnalisé
- Création d'un cluster de bases de données MySQL à l'aide d'un groupe de paramètres de cluster de bases de données

Prérequis

Ce didacticiel nécessite que vous disposiez d'un Compte AWS et d'un utilisateur disposant d'un accès administratif. Si vous ne l'avez pas encore fait, suivez les étapes indiquées dans les sections ci-dessous pour vous préparer :

- Inscrivez-vous pour un Compte AWS
- Création d'un utilisateur doté d'un accès administratif

Création d'un groupe de paramètres de cluster de bases de données Amazon Aurora

Dans ce didacticiel, vous apprendrez à créer un groupe de paramètres personnalisé avec default_password_lifetime pour un cluster de bases de données MySQL dans la console. Le paramètre `default_password_lifetime` contrôle le nombre de jours avant l'expiration automatique du mot de passe du client. Pour plus d'informations sur les autres paramètres disponibles pour les clusters de bases de données MySQL, consultez Paramètres de configuration d'Aurora MySQL .

Pour créer un groupe de paramètres

1. Ouvrez la console Amazon RDS et choisissez Groupes de paramètres.
2. Sur la page Groupes de paramètres personnalisés, choisissez Créer un groupe de paramètres.
3. Définissez les détails du groupe de paramètres.
    1. Entrez un nom pour le groupe de paramètres.
    2. Entrez une description du groupe de paramètres.
    3. Pour Type de moteur, choisissez Aurora MySQL.
    4. Pour Famille de groupes de paramètres, choisissez aurora-mysql8.0.
    5. Pour Type, choisissez Groupe de paramètres de cluster de bases de données.
4. Choisissez Créer.

Le nouveau groupe de paramètres de cluster de bases de données apparaît sur la page Groupes de paramètres de la console Amazon RDS. Les étapes suivantes montrent comment modifier les valeurs des paramètres pour personnaliser votre groupe de paramètres.

Modification de la valeur de paramètre dans votre groupe de paramètres personnalisé

Procédez comme suit pour modifier la valeur de paramètre dans le groupe de paramètres que vous avez créé dans Création d'un groupe de paramètres de cluster de bases de données Amazon Aurora.

Pour modifier les valeurs de paramètre dans votre groupe de paramètres

1. Ouvrez la console Amazon RDS et choisissez Groupes de paramètres.
2. Pour Groupes de paramètres personnalisés, choisissez le nom du groupe de paramètres de cluster de bases de données que vous avez créé.
3. Choisissez Modifier.
4. Dans la zone de recherche Paramètres de filtre, recherchez le paramètre personnalisé `default_password_lifetime`.
5. Cochez la case à côté du paramètre et entrez une valeur correspondant au nombre de jours à définir pour ce paramètre de durée de vie du mot de passe.
6. Sélectionnez Save Changes (Enregistrer les modifications).

Le groupe de paramètres personnalisé peut désormais être associé à Amazon Aurora pour le cluster de bases de données MySQL 8.0.

Création d'un cluster de bases de données MySQL à l'aide d'un groupe de paramètres de cluster de bases de données

Enfin, créez un cluster de bases de données MySQL avec le groupe de paramètres personnalisé que vous avez créé lors des étapes précédentes. Les étapes suivantes montrent comment créer le cluster de bases de données MySQL avec votre groupe de paramètres personnalisé .

Pour créer un cluster de bases de données à l'aide d'un groupe de paramètres personnalisé et d'un nouveau groupe d'options

1. Ouvrez la console Amazon RDS et choisissez Bases de données.

2. Choisissez Create database (Créer une base de données).
3. Pour Choisir une méthode de création de base de données, choisissez Création standard.
4. Pour Options de moteur, choisissez Aurora (compatible MySQL).
5. Choisissez Configuration supplémentaire.
    - Pour Nom de la base de données initiale, choisissez un nom pour votre cluster de bases de données.
    - Dans le menu déroulant du groupe de paramètres de cluster BASE DE DONNÉES, choisissez le nom du groupe de paramètres de cluster de bases de données que vous avez créé auparavant.
6. Pour ce didacticiel, vous pouvez conserver les paramètres par défaut pour tous les autres paramètres de base de données ou les modifier selon vos besoins.
7. Choisissez Create database (Créer une base de données).

RDS crée un nouveau cluster de bases de données MySQL à l'aide d'un groupe de paramètres personnalisé. Pour plus d'informations sur cette base de données, consultez la page Bases de données de la console Amazon RDS.

Dans ce didacticiel, vous créez MySQL un cluster de bases de données MySQL à l'aide d'un groupe de paramètres personnalisé . le cluster de bases de données tout juste créé gère la durée de vie du mot de passe utilisateur à l'aide du paramètre `default_password_lifetime`. Pour optimiser votre base de données, vous pouvez appliquer des paramètres supplémentaires à votre groupe de paramètres personnalisé et ajouter des options.

Une fois que vous avez terminé de créer votre cluster de bases de données personnalisé, vous devez supprimer vos ressources pour éviter d'encourir des coûts indésirables. Pour supprimer un cluster de bases de données, suivez les instructions de Suppression de clusters de bases de données Aurora et d'instances de bases de données.

Gestion d’un cluster de bases de données Amazon Aurora

Cette section explique comment gérer et maintenir votre cluster de bases de données Aurora. Aurora implique des clusters de serveurs de base de données qui sont connectés dans une topologie de réplication. Par conséquent, la gestion d’Aurora implique souvent de déployer des modifications sur plusieurs serveurs et de s’assurer que tous les réplicas Aurora suivent le serveur source. Étant donné qu’Aurora dimensionne le stockage sous-jacent en toute transparence à mesure que vos données augmentent, la gestion d’Aurora exige relativement peu de gestion du stockage sur disque. De la même manière, étant donné qu’Aurora effectue automatiquement des sauvegardes continues, un cluster Aurora nécessite peu de planification ou de durée d’indisponibilité pour la réalisation des sauvegardes.

Rubriques

Arrêt et démarrage d’un cluster de bases de données Amazon Aurora

L’arrêt et le démarrage des clusters de bases de données Aurora vous permettent de maîtriser les coûts liés aux environnements de développement et de test. Vous pouvez arrêter temporairement toutes les instances de base de données de votre cluster au lieu de configurer et de détruire toutes les instances de base de données chaque fois que vous utilisez le cluster.

Rubriques

Présentation de l’arrêt et du démarrage d’un cluster de bases de données Aurora

Pendant les périodes où vous n’avez pas besoin d’un cluster de bases de données Aurora, vous pouvez arrêter toutes les instances du cluster en une seule opération. Vous pouvez à tout moment redémarrer le cluster dès que vous avez besoin de l’utiliser. Le démarrage et l’arrêt simplifie les processus de configuration et de destruction des clusters utilisés à des fins de développement, de test ou d’activités similaires qui ne nécessitent pas une disponibilité continue. Vous pouvez exécuter toutes les procédures d’AWS Management Console en question en une seule opération, quel que soit le nombre d’instances présentes dans le cluster.

Pendant que votre cluster de bases de données est à l’arrêt, vous êtes facturé uniquement pour le stockage du cluster, des instantanés manuels et des sauvegardes automatiques dans le cadre de la période de conservation que vous avez spécifiée. Aucune heure d’instance de base de données ne vous est facturée.

> **Important**
>
> Vous pouvez arrêter un cluster de bases de données pendant sept jours au maximum. Si vous ne démarrez pas manuellement votre cluster de bases de données après sept jours,

votre cluster de bases de données est automatiquement démarré afin qu'il ne prenne pas de retard dans les mises à jour de maintenance requises.

Pour limiter les frais pour un cluster Aurora à faible charge, vous pouvez arrêter le cluster plutôt que de supprimer tous ses réplicas Aurora. Pour les clusters constitués d'une ou deux instances, il n'est pas commode de supprimer et de récréer fréquemment les instances de base de données, à moins d'utiliser l'AWS CLI ou l'API Amazon RDS. Il n'est pas non plus évident d'effectuer ces opérations dans le bon ordre. Par exemple, pour éviter l'activation du mécanisme de basculement, il convient de supprimer tous les réplicas Aurora avant de supprimer l'instance principale.

De même, évitez de démarrer et d'arrêter votre cluster de bases de données s'il doit s'exécuter en permanence, mais que sa capacité est supérieure à vos besoins. Si votre cluster est trop coûteux ou sous-utilisé, supprimez une ou plusieurs instances de base de données ou attribuez leur à toutes une classe d'instance de petite taille (small). Vous ne pouvez pas arrêter une instance de base de données Aurora individuelle.

Le délai d'arrêt de votre cluster de bases de données varie en fonction de facteurs tels que les classes d'instance de base de données, l'état du réseau et de la base de données, ainsi que le type de moteur de base de données. Ce processus peut prendre plusieurs minutes. Le service Amazon RDS effectue les actions suivantes :

- Il arrête les processus du moteur de base de données.
- Il arrête les processus de la plateforme RDS.
- Il met fin à l'instance Amazon EC2 sous-jacente.

Le délai de redémarrage de votre cluster de bases de données varie en fonction de facteurs tels que la taille de la base de données, les classes d'instance de base de données, l'état du réseau, le type de moteur de base de données et l'état de la base de données au moment où le cluster a été arrêté. Le processus de démarrage dure en général quelques minutes, mais peut durer jusqu'à plusieurs heures. Nous vous recommandons de tenir compte du caractère variable de la durée de démarrage lorsque vous créez votre plan de disponibilité.

Pour démarrer le cluster de bases de données, le service exécute des actions telles que les suivantes :

- Il met en service les instances Amazon EC2 sous-jacentes.
- Il démarre les processus de la plateforme RDS.

- Il démarre les processus du moteur de base de données.
- Il restaure les instances de base de données (la restauration a lieu même après un arrêt normal).

Limites liées à l’arrêt et au démarrage des clusters de bases de données Aurora

Certains clusters Aurora ne peuvent pas être arrêtés et démarrés :

- Vous ne pouvez arrêter et démarrer un cluster faisant partie d’une base de données globale Aurora que s’il s’agit du seul cluster dans la base de données globale.
- Vous ne pouvez pas arrêter et démarrer un cluster qui possède un réplica en lecture entre plusieurs régions.
- Vous ne pouvez pas arrêter ni démarrer un cluster faisant partie d’un déploiement bleu/vert.
- Vous ne pouvez pas arrêter et démarrer un cluster Aurora Serverless v1. Avec Aurora Serverless v2, vous pouvez arrêter et démarrer le cluster.

Arrêt d’un cluster de bases de données Aurora

Pour utiliser un cluster de bases de données Aurora ou effectuer des tâches d’administration, vous partez toujours d’un cluster de bases de données Aurora en cours d’exécution, vous l’arrêtez, puis le redémarrez. Pendant que votre cluster est à l’arrêt, vous êtes facturé pour le stockage du cluster, des instantanés manuels et des sauvegardes automatiques dans le cadre de votre fenêtre de rétention spécifiée, mais pas pour les heures d’instance de base de données.

L’opération d’arrêt stoppe d’abord les instances de réplica Aurora, puis l’instance principale, pour éviter l’activation du mécanisme de basculement.

Console

Pour arrêter un cluster Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, sélectionnez Bases de données, choisissez un cluster. Vous pouvez effectuer l’opération d’arrêt soit à partir de cette page, soit en accédant à la page de détails du cluster de bases de données que vous voulez arrêter.

3. Pour Actions, choisissez Stop temporarily (Arrêter temporairement).
4. Dans la fenêtre Stop DB cluster temporarily (Arrêter temporairement le cluster de bases de données), sélectionnez l'accusé de réception indiquant que le cluster de bases de données redémarrera automatiquement au bout de 7 jours.
5. Choisissez Stop temporarily (Arrêter temporairement) pour arrêter le cluster de bases de données ou choisissez Cancel (Annuler) pour annuler l'opération.

AWS CLI

Pour arrêter une instance de base de données à partir de l'AWS CLI, appelez la commande stop-db-cluster avec les paramètres suivants :

- `--db-cluster-identifier` – Nom du cluster Aurora.

Example

```
aws rds stop-db-cluster --db-cluster-identifier mydbcluster
```

API RDS

Pour arrêter une instance de base de données à partir de l'API Amazon RDS, appelez l'opération StopDBCluster avec le paramètre suivant :

- `DBClusterIdentifier` – Nom du cluster Aurora.

Opérations possibles pendant qu'un cluster de bases de données Aurora est à l'arrêt

Pendant qu'un cluster Aurora est à l'arrêt, vous pouvez effectuer une restauration à un instant dans le passé à n'importe quel moment dans votre fenêtre de rétention de sauvegarde automatisée spécifiée. Pour plus d'informations sur la réalisation d'une restauration à un instant dans le passé, consultez Restauration des données.

Vous ne pouvez pas modifier la configuration d'un cluster de bases de données Aurora ou de l'une de ses instances pendant que le cluster est à l'arrêt. De même, vous ne pouvez pas ajouter ou supprimer des instances de base de données au niveau du cluster, ni supprimer le cluster si une ou

plusieurs instances de base de données lui sont toujours associées. Vous devez démarrer le cluster avant d'effectuer des opérations d'administration de ce type.

L'arrêt d'un cluster de bases de données supprime les actions en attente, à l'exception du groupe de paramètres du cluster de bases de données ou des groupes de paramètres de base de données des instances du cluster de bases de données.

Aurora applique la maintenance planifiée à votre cluster arrêté une fois qu'il a redémarré. N'oubliez pas qu'après sept jours, Aurora démarre automatiquement les clusters arrêtés pour éviter qu'ils soient trop en retard par rapport à leur état de maintenance.

Par ailleurs, Aurora n'effectue aucune sauvegarde automatisée parce que les données sous-jacentes ne peuvent pas changer pendant que le cluster est à l'arrêt. Aurora ne prolonge pas la période de rétention des sauvegardes pendant que le cluster est à l'arrêt.

Démarrage d'un cluster de bases de données Aurora

Le cluster de bases de données Aurora que vous démarrez est toujours un cluster Aurora qui est déjà à l'état arrêté (ou « stopped »). Lorsque vous démarrez le cluster, toutes ses instances de base de données redeviennent disponibles. Le cluster conserve ses paramètres de configuration, notamment les points de terminaison, les groupes de paramètres et les groupes de sécurité VPC.

Le redémarrage d'un cluster de bases de données prend généralement plusieurs minutes.

Console

Pour démarrer un cluster Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, sélectionnez Bases de données, choisissez un cluster. Vous pouvez effectuer l'opération de démarrage à partir de cette page, ou accéder à la page de détails du cluster de bases de données que vous voulez démarrer.
3. Pour Actions, choisissez Start (Démarrer).

AWS CLI

Pour démarrer un cluster de bases de données à partir de l'AWS CLI, appelez la commande start-db-cluster avec les paramètres suivants :

- `--db-cluster-identifier` – Nom du cluster Aurora. Ce nom est soit l'identifiant de cluster que vous avez choisi au moment de créer le cluster, soit l'identifiant d'instance de base de données que vous avez choisi et auquel la terminaison `-cluster` a été ajoutée.

Example

```
aws rds start-db-cluster --db-cluster-identifier mydbcluster
```

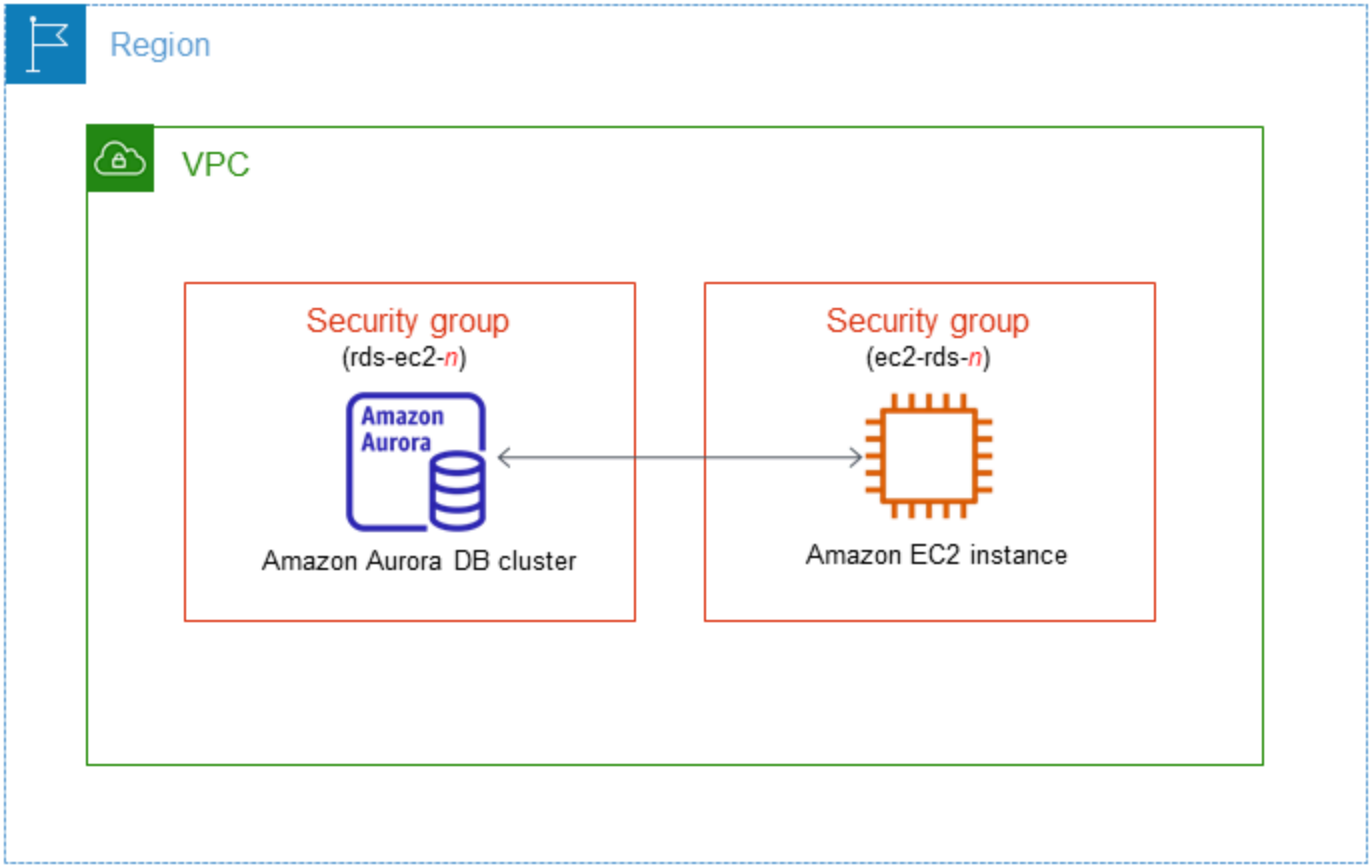
API RDS

Pour démarrer un cluster de bases de données Aurora à partir de l'API Amazon RDS, appelez l'opération StartDBCluster avec le paramètre suivant :

- `DBCluster` – Nom du cluster Aurora. Ce nom est soit l'identifiant de cluster que vous avez choisi au moment de créer le cluster, soit l'identifiant d'instance de base de données que vous avez choisi et auquel la terminaison `-cluster` a été ajoutée.

Connexion automatique d’une instance EC2 et d’un cluster de bases de données Aurora

Vous pouvez utiliser la console Amazon RDS pour simplifier la configuration d’une connexion entre une instance Amazon Elastic Compute Cloud (Amazon EC2) et un cluster de bases de données Aurora. Souvent, votre cluster de bases de données se trouve dans un sous-réseau privé et votre instance EC2 se trouve dans un sous-réseau public au sein d’un VPC. Vous pouvez utiliser un client SQL sur votre instance EC2 pour vous connecter à votre cluster de bases de données. L’instance EC2 peut également exécuter des serveurs Web ou des applications qui accèdent à votre cluster de bases de données.



Si vous souhaitez vous connecter à une instance EC2 qui ne figure pas dans le même VPC que le cluster de bases de données Aurora, consultez les scénarios dans Scénarios d’accès à un cluster de bases de données d’un VPC.

Rubriques

- Présentation de la connectivité automatique avec une instance EC2

- Connexion automatique d’une instance EC2 et d’un cluster de bases de données Aurora
- Affichage des ressources de calcul connectées
- Connexion à une instance de base de données qui exécute un moteur de base de données spécifique

Présentation de la connectivité automatique avec une instance EC2

Lorsque vous configurez une connexion entre une instance EC2 et un cluster de bases de données Aurora, Amazon RDS configure automatiquement le groupe de sécurité VPC pour votre instance EC2 et pour votre cluster de bases de données.

Voici les conditions requises pour connecter une instance EC2 avec un cluster de bases de données Aurora :

- L’instance EC2 doit exister dans le même VPC que le cluster de bases de données.

  S’il n’y a pas d’instances EC2 dans le même VPC, la console fournit un lien pour en créer une.
- Actuellement, le cluster de bases de données ne peut pas être un cluster de bases de données Aurora Serverless ou une partie d’une base de données globale Aurora.
- L’utilisateur qui configure la connectivité doit avoir les autorisations nécessaires pour effectuer les opérations Amazon EC2 suivantes :
  - `ec2:AuthorizeSecurityGroupEgress`
  - `ec2:AuthorizeSecurityGroupIngress`
  - `ec2:CreateSecurityGroup`
  - `ec2:DescribeInstances`
  - `ec2:DescribeNetworkInterfaces`
  - `ec2:DescribeSecurityGroups`
  - `ec2:ModifyNetworkInterfaceAttribute`
  - `ec2:RevokeSecurityGroupEgress`

Si l’instance de base de données et l’instance EC2 se trouvent dans des zones de disponibilité différentes, votre compte peut être confronté à des coûts croisés entre zones de disponibilité.

Lorsque vous configurez une connexion à une instance EC2, Amazon RDS agit en fonction de la configuration actuelle des groupes de sécurité associés au cluster de bases de données et à l'instance EC2, comme décrit dans le tableau suivant.

Configuration du groupe de sécurité RDS actuel	Configuration du groupe de sécurité EC2 actuel	Action RDS
Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-ec2-`*n* (où *n* est un nombre). Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité comprend une seule règle entrante avec le groupe de sécurité du VPC de l'instance EC2 comme source.	Un ou plusieurs groupes de sécurité sont associés à l'instance EC2 avec un nom qui correspond au modèle `ec2-rds-`*n* (où *n* est un nombre). Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité comprend une seule règle de sortie avec le groupe de sécurité du VPC du cluster de bases de données comme source.	RDS ne fait rien. Une connexion était déjà configurée automatiquement entre l'instance EC2 et le cluster de bases de données. Comme une connexion existe déjà entre l'instance EC2 et la base de données RDS, les groupes de sécurité ne sont pas modifiés.
L'une des conditions suivantes s'applique : • Aucun groupe de sécurité n'est associé au cluster de bases de données avec un nom qui correspond au modèle `rds-ec2-`*n*. • Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-ec2-`*n*. Toutefois, Amazon RDS ne peut	L'une des conditions suivantes s'applique : • Aucun groupe de sécurité n'est associé à l'instance EC2 avec un nom qui correspond au modèle `ec2-rds-`*n*. • Un ou plusieurs groupes de sécurité sont associés à l'instance EC2 avec un nom qui correspond au modèle `ec2-rds-`*n*. Toutefois, Amazon RDS ne peut utiliser aucun de ces	RDS action: create new security groups

Configuration du groupe de sécurité RDS actuel	Configuration du groupe de sécurité EC2 actuel	Action RDS
utiliser aucun de ces groupes de sécurité pour la connexion avec l'instance EC2. Amazon RDS ne peut pas utiliser comme source un groupe de sécurité dépourvu de toute règle entrante avec le groupe de sécurité VPC de l'instanc e EC2. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié. Des exemples de modifications incluent l'ajout d'une règle ou la modification du port d'une règle existante.	groupes de sécurité pour la connexion avec le cluster de bases de données. Amazon RDS ne peut pas utiliser comme source un groupe de sécurité dépourvu de toute règle sortante avec le groupe de sécurité VPC du cluster de bases de données. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.	

Configuration du groupe de sécurité RDS actuel	Configuration du groupe de sécurité EC2 actuel	Action RDS
Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-ec2-`*n*. Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité comprend une seule règle entrante avec le groupe de sécurité du VPC de l'instance EC2 comme source.	Un ou plusieurs groupes de sécurité sont associés à l'instance EC2 avec un nom qui correspond au modèle `ec2-rds-`*n*. Toutefois, Amazon RDS ne peut utiliser aucun de ces groupes de sécurité pour la connexion avec le cluster de bases de données. Amazon RDS ne peut pas utiliser comme source un groupe de sécurité dépourvu de toute règle sortante avec le groupe de sécurité VPC du cluster de bases de données. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.	RDS action: create new security groups
Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-ec2-`*n*. Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité comprend une seule règle entrante avec le groupe de sécurité du VPC de l'instance EC2 comme source.	Il existe un groupe de sécurité EC2 valide pour la connexion , mais il n'est pas associé à l'instance EC2. Le nom de ce groupe de sécurité correspond au modèle `ec2-rds-`*n*. Il n'a pas été modifié. Il comprend une seule règle de sortie avec le groupe de sécurité du VPC du cluster de bases de données comme source.	RDS action: associate EC2 security group

Configuration du groupe de sécurité RDS actuel	Configuration du groupe de sécurité EC2 actuel	Action RDS
L'une des conditions suivantes s'applique : • Aucun groupe de sécurité n'est associé au cluster de bases de données avec un nom qui correspond au modèle `rds-ec2-`*n*. • Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-ec2-`*n*. Toutefois, Amazon RDS ne peut utiliser aucun de ces groupes de sécurité pour la connexion avec l'instance EC2. Amazon RDS ne peut pas utiliser comme source un groupe de sécurité dépourvu de toute règle entrante avec le groupe de sécurité VPC de l'instance EC2. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.	Un ou plusieurs groupes de sécurité sont associés à l'instance EC2 avec un nom qui correspond au modèle `ec2-rds-`*n*. Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité comprend une seule règle de sortie avec le groupe de sécurité du VPC du cluster de bases de données comme source.	RDS action: create new security groups

Action RDS : créer de nouveaux groupes de sécurité

Amazon RDS entreprend les actions suivantes :

- Crée un nouveau groupe de sécurité qui correspond au modèle `rds-ec2-`*`n`*. Ce groupe de sécurité comprend une règle entrante avec le groupe de sécurité du VPC de l'instance EC2 comme source. Ce groupe de sécurité est associé au cluster de bases de données et permet à l'instance EC2 d'accéder au cluster de bases de données.
- Crée un nouveau groupe de sécurité qui correspond au modèle `ec2-rds-`*`n`*. Ce groupe de sécurité comprend une règle sortante avec le groupe de sécurité du VPC du cluster de bases de données comme cible. Ce groupe de sécurité est associé à l'instance EC2 et permet à l'instance EC2 d'envoyer du trafic vers le cluster de bases de données.

Action RDS : associer un groupe de sécurité EC2

Amazon RDS associe le groupe de sécurité EC2 existant valide à l'instance EC2. Ce groupe de sécurité permet à l'instance EC2 d'envoyer du trafic vers le cluster de bases de données.

Connexion automatique d'une instance EC2 et d'un cluster de bases de données Aurora

Avant de configurer une connexion entre une instance EC2 et un cluster de bases de données Aurora, assurez-vous de répondre aux exigences décrites dans Présentation de la connectivité automatique avec une instance EC2.

Si vous modifiez ces groupes de sécurité après avoir configuré la connectivité, cela peut affecter la connexion entre l'instance EC2 et le cluster de bases de données Aurora.

> **Note**
>
> Vous pouvez uniquement configurer automatiquement une connexion entre une instance EC2 et un cluster de bases de données Aurora à l'aide de la AWS Management Console. Vous ne pouvez pas configurer de connexion automatiquement avec l'AWS CLI ou l'API RDS.

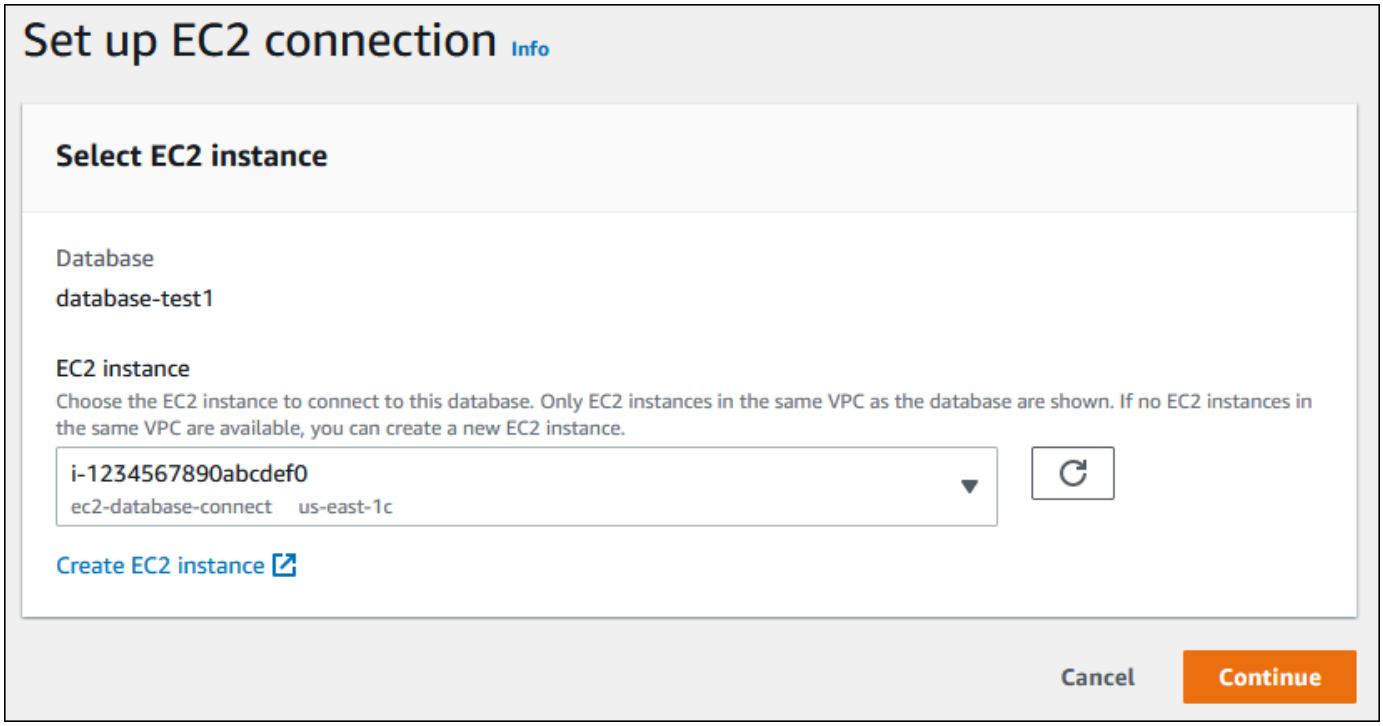
Connecter automatiquement une instance EC2 et un cluster de bases de données Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis Cluster de bases de données.

3. Pour Actions, choisissez Configurer la connexion EC2.

   La page Set up EC2 connection (Configurer la connexion EC2) s'affiche.

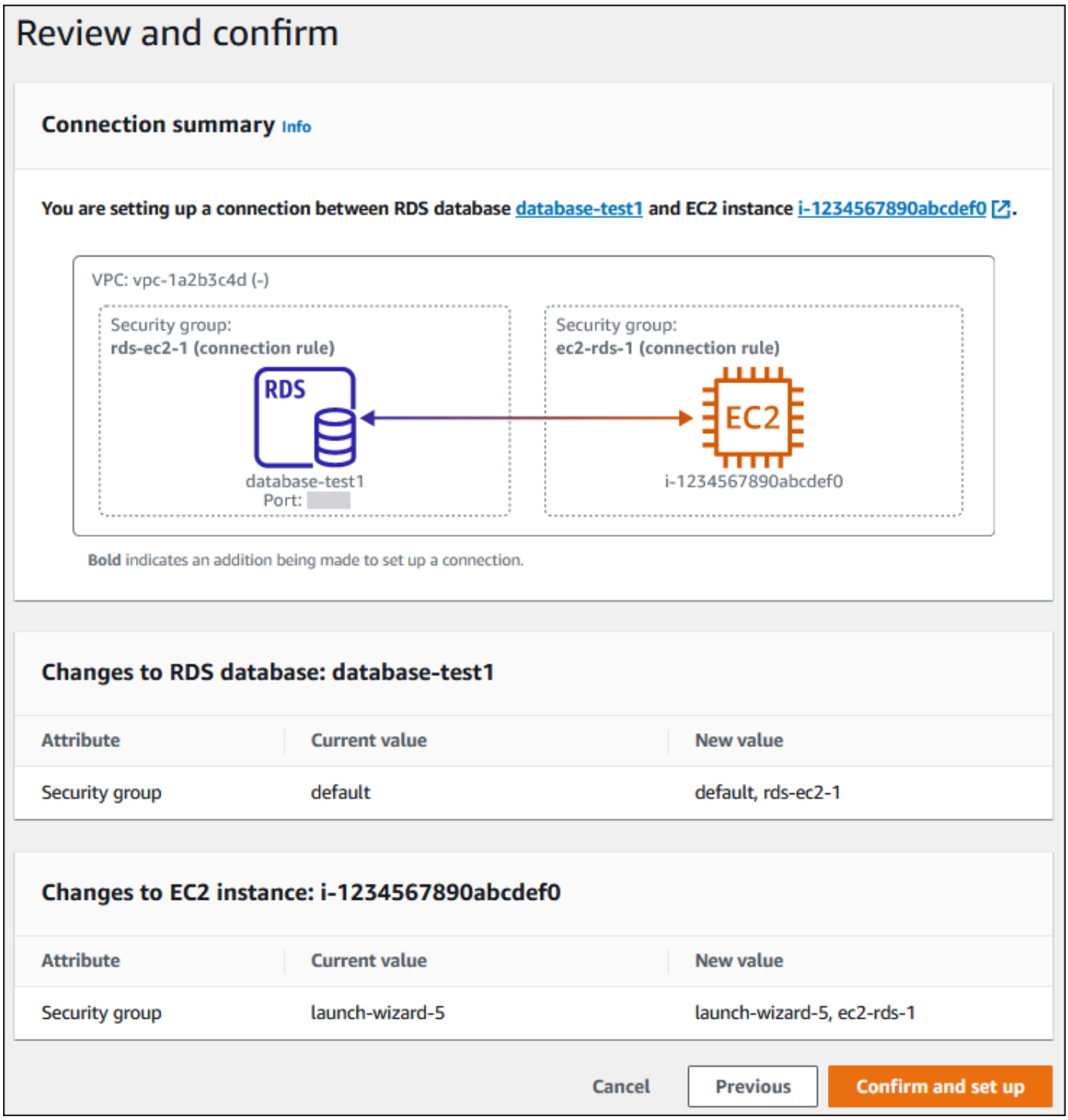
4. Sur la page Set up EC2 connection (Configurer la connexion EC2), choisissez l'instance EC2.



   Si aucune instance EC2 n'existe dans le même VPC, choisissez Create EC2 instance (Créer une instance EC2) pour en créer une. Dans ce cas, assurez-vous que la nouvelle instance EC2 se trouve dans le même VPC que le cluster de bases de données.

5. Choisissez Continuer.

   La page Review and confirm (Vérifier et confirmer) s'affiche.



6. Sur la page Review and confirm (Vérifier et confirmer), passez en revue les modifications que RDS apportera pour configurer la connectivité avec l'instance EC2.

   Si les modifications sont correctes, choisissez Confirmer et configurer.

   Si les modifications ne sont pas correctes, choisissez Previous (Précédent) ou Cancel (Annuler).

Affichage des ressources de calcul connectées

Vous pouvez utiliser la AWS Management Console pour afficher les ressources de calcul qui sont connectées à un cluster de base de données Aurora. Les ressources affichées comprennent les connexions de ressources de calcul qui ont été configurées automatiquement. Vous pouvez configurer automatiquement la connectivité avec les ressources de calcul de la manière suivante :

- Vous pouvez sélectionner la ressource de calcul lorsque vous créez la base de données.

  Pour plus d'informations, consultez Création d'un cluster de bases de données Amazon Aurora.
- Vous pouvez configurer la connectivité entre une base de données existante et une ressource de calcul.

  Pour plus d'informations, consultez Connexion automatique d'une instance EC2 et d'un cluster de bases de données Aurora.

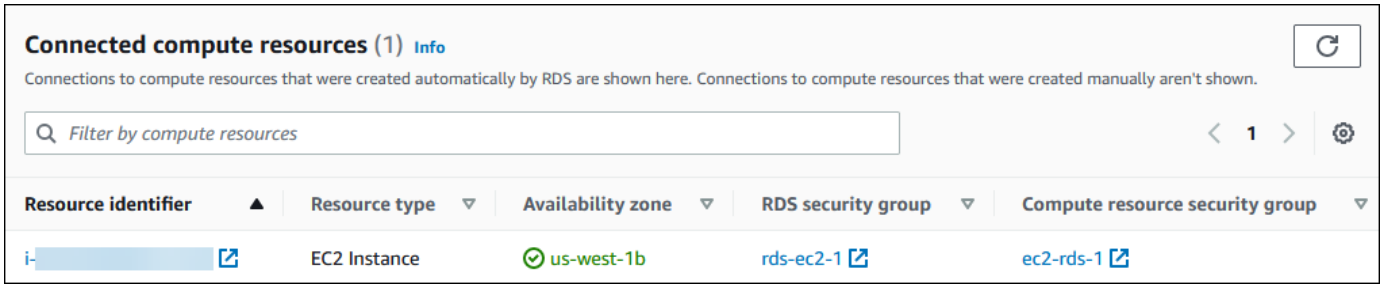
Les ressources de calcul répertoriées n'incluent pas celles qui ont été connectées manuellement à la base de données. Par exemple, vous pouvez autoriser une ressource de calcul à accéder manuellement à une base de données en ajoutant une règle au groupe de sécurité du VPC associé à la base de données.

Pour qu'une ressource de calcul soit répertoriée, les conditions suivantes doivent s'appliquer :

- Le nom du groupe de sécurité associé à la ressource de calcul correspond au modèle `ec2-rds-`*n* (où *n* est un nombre).
- Le groupe de sécurité associé à la ressource de calcul possède une règle sortante avec la plage de ports définie sur le port utilisé par le cluster de bases de données.
- Le groupe de sécurité associé à la ressource de calcul possède une règle de sortie dont la source est définie sur un groupe de sécurité associé au cluster de bases de données.
- Le nom du groupe de sécurité associé au cluster de bases de données correspond au modèle `rds-ec2-`*n* (où *n* est un nombre).
- Le groupe de sécurité associé au cluster de bases de données possède une règle entrante avec la plage de ports définie sur le port utilisé par le cluster de bases de données.
- Le groupe de sécurité associé au cluster de bases de données possède une règle d'entrée dont la source est définie sur un groupe de sécurité associé à la ressource informatique.

Pour visualiser les ressources de calcul connectées à un cluster de bases de données Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis le nom du cluster de bases de données.
3. Dans l'onglet Connectivity & security (Connectivité et sécurité), affichez les ressources de calcul dans Connected compute resources (Ressources de calcul connectées).



Connexion à une instance de base de données qui exécute un moteur de base de données spécifique

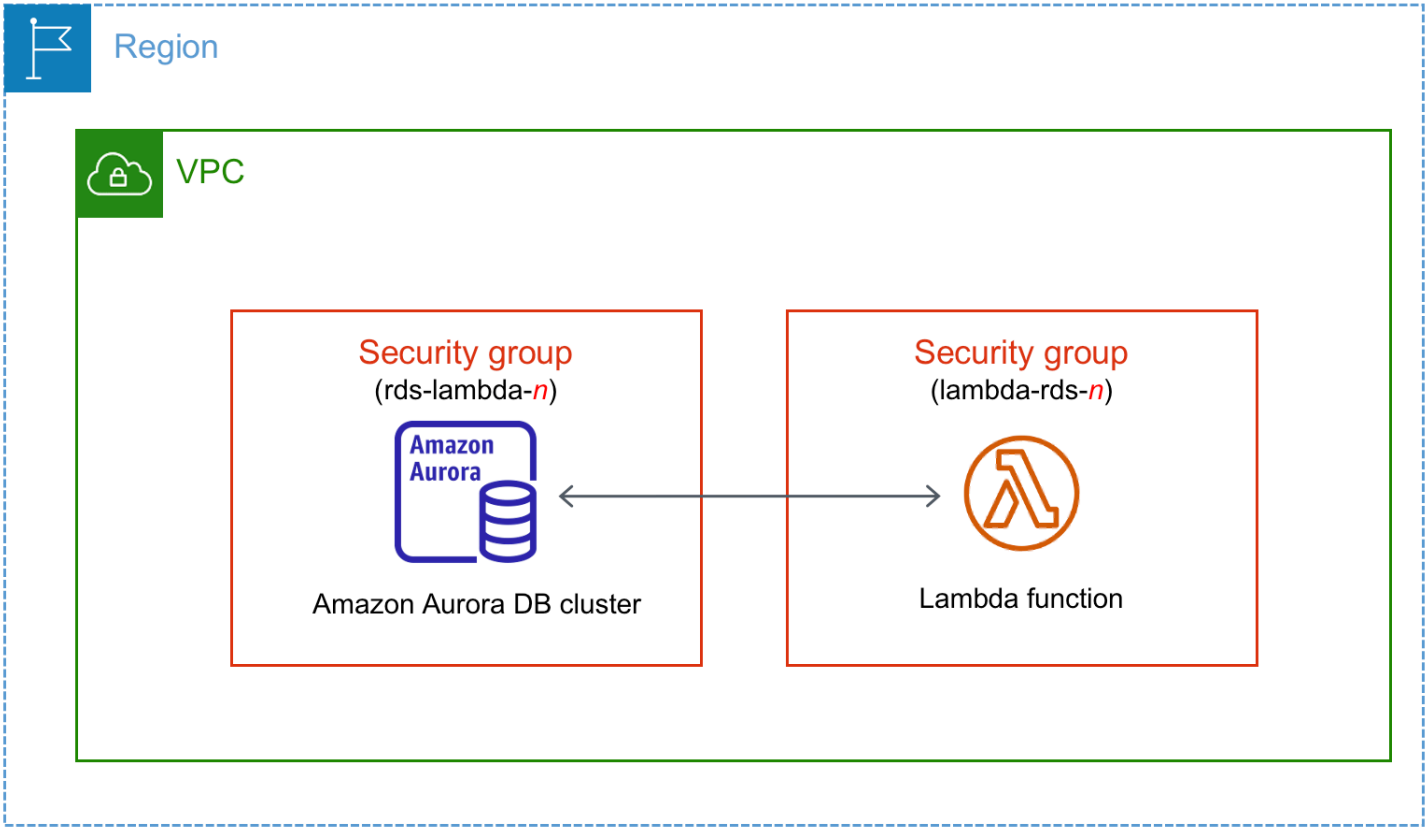
Pour plus d'informations sur la connexion à une instance de base de données qui exécute un moteur de base de données spécifique, suivez les instructions relatives à votre moteur de base de données :

- Connexion à un cluster de bases de données Amazon Aurora MySQL
- Connexion à un cluster de bases de données Amazon Aurora PostgreSQL

Connexion automatique d’une fonction Lambda et d’un cluster de bases de données Aurora

Vous pouvez utiliser la console Amazon RDS pour simplifier la configuration d’une connexion entre une fonction Lambda et un cluster de bases de données Aurora. Souvent, votre cluster de bases de données se trouve dans un sous-réseau privé au sein d’un VPC. La fonction Lambda peut être utilisée par les applications pour accéder à votre cluster de bases de données privé.

L’image suivante montre une connexion directe entre votre cluster de bases de données et votre fonction Lambda.

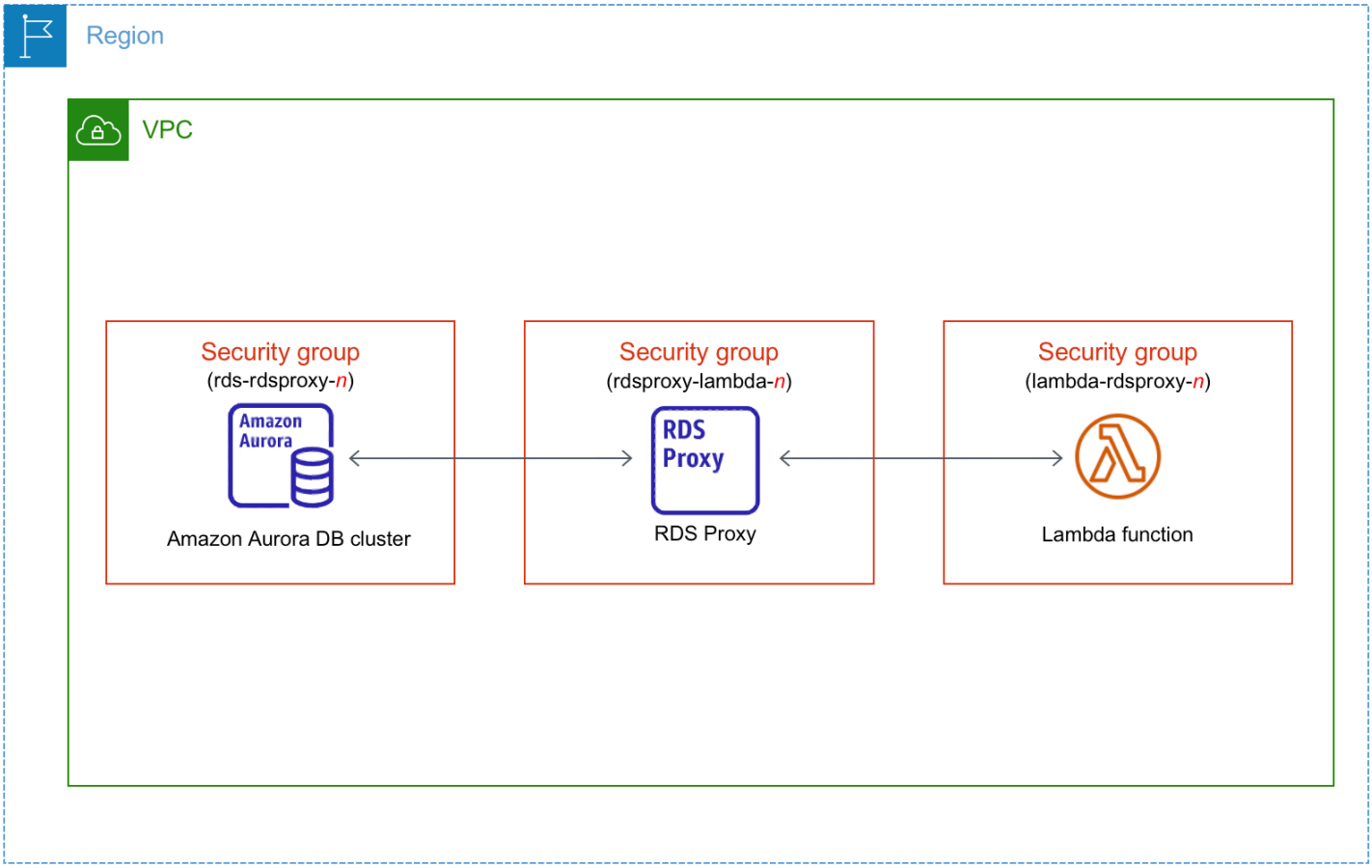


Vous pouvez configurer la connexion entre votre fonction Lambda et votre cluster de bases de données via un proxy RDS pour améliorer les performances et la résilience de votre base de données. Souvent, les fonctions Lambda établissent des connexions de base de données courtes et fréquentes qui bénéficient du regroupement de connexions offert par le proxy RDS. Vous pouvez profiter de toute authentification Gestion des identités et des accès AWS (IAM) dont vous disposez déjà pour les fonctions Lambda, plutôt que de gérer les informations d’identification de base de

données dans votre code d'application Lambda. Pour plus d'informations, consultez Proxy Amazon RDS pour Aurora.

Lorsque vous utilisez la console pour vous connecter à un proxy existant, Amazon RDS met à jour le groupe de sécurité du proxy pour autoriser les connexions depuis votre cluster de bases de données et la fonction Lambda.

Vous pouvez également créer un nouveau proxy à partir de la même page de console. Lorsque vous créez un proxy dans la console, pour accéder au cluster de bases de données, vous devez saisir vos informations d'identification de base de données ou sélectionner un secret AWS Secrets Manager.



> **Tip**
>
> Pour connecter rapidement une fonction Lambda à une un cluster de bases de données Aurora, vous pouvez également utiliser l'assistant intégré à la console. Procédez comme suit pour ouvrir l'assistant :
>
> 1. Ouvrez la page Functions (Fonctions) de la console Lambda.
> 2. Sélectionnez la fonction à laquelle vous souhaitez connecter une base de données.

3. Dans l’onglet Configuration, sélectionnez Bases de données RDS.
4. Choisissez Se connecter à la base de données RDS.

Après avoir connecté votre fonction à une base de données, vous pouvez créer un proxy en choisissant Ajouter un proxy.

Rubriques

Vue d’ensemble de la connectivité automatique avec une fonction Lambda

Voici les conditions requises pour connecter une fonction Lambda avec un cluster de bases de données Aurora :

- La fonction Lambda doit exister dans le même VPC que le cluster de bases de données.
- Actuellement, le cluster de bases de données ne peut pas être un cluster de bases de données Aurora Serverless ou une partie d’une base de données globale Aurora.
- L’utilisateur qui configure la connectivité doit avoir les autorisations nécessaires pour effectuer les opérations Amazon RDS, Amazon EC2, Lambda, Secrets Manager et IAM suivantes :
  - Amazon RDS
    - `rds:CreateDBProxies`
    - `rds:DescribeDBClusters`
    - `rds:DescribeDBProxies`
    - `rds:ModifyDBCluster`
    - `rds:ModifyDBProxy`
    - `rds:RegisterProxyTargets`
  - Amazon EC2
    - `ec2:AuthorizeSecurityGroupEgress`
    - `ec2:AuthorizeSecurityGroupIngress`
    - `ec2:CreateSecurityGroup`

    - `ec2:DeleteSecurityGroup`
    - `ec2:DescribeSecurityGroups`
    - `ec2:RevokeSecurityGroupEgress`
    - `ec2:RevokeSecurityGroupIngress`
- Lambda
    - `lambda:CreateFunctions`
    - `lambda:ListFunctions`
    - `lambda:UpdateFunctionConfiguration`
- Secrets Manager
    - `secretsmanager:CreateSecret`
    - `secretsmanager:DescribeSecret`
- IAM
    - `iam:AttachPolicy`
    - `iam:CreateRole`
    - `iam:CreatePolicy`
- AWS KMS
    - `kms:describeKey`

> **Note**
>
> Si le cluster de bases de données et la fonction Lambda se trouvent dans des zones de disponibilité différentes, votre compte peut être confronté à des coûts croisés entre zones de disponibilité.

Lorsque vous configurez une connexion entre une fonction Lambda et un cluster de bases de données Aurora, Amazon RDS configure le groupe de sécurité VPC pour votre fonction et pour votre cluster de bases de données. Si vous utilisez un proxy RDS, Amazon RDS configure également le groupe de sécurité VPC pour le proxy. Amazon RDS agit en fonction de la configuration actuelle des groupes de sécurité associés au cluster de bases de données, à la fonction Lambda et au proxy, comme décrit dans le tableau suivant.

Configuration du groupe de sécurité RDS actuel	Configuration actuelle du groupe de sécurité Lambda	Configuration actuelle du groupe de sécurité du proxy	Action RDS
Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-lambda-` *n* ou si un proxy est déjà connecté à votre cluster de bases de données, RDS vérifie si l'élément `TargetHealth` d'un proxy associé a pour valeur `AVAILABLE` . Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité comprend une seule règle entrante avec le groupe de sécurité VPC de la fonction Lambda ou du proxy comme source.	Un ou plusieurs groupes de sécurité sont associés à la fonction Lambda avec un nom qui correspond au modèle `lambda-rds-` *n* ou `lambda-rdsproxy-` *n* (où *n* est un nombre). Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité ne possède qu'une seule règle sortante avec le groupe de sécurité VPC du cluster de bases de données ou du proxy comme destination.	Un ou plusieurs groupes de sécurité sont associés au proxy avec un nom qui correspond au modèle `rdsproxy-lambda-` *n* (où *n* est un nombre). Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité possède des règles entrantes et sortantes avec les groupes de sécurité VPC de la fonction Lambda et du cluster de bases de données.	Amazon RDS n'entreprend aucune action. Une connexion a déjà été configuré e automatiquement entre la fonction Lambda, le proxy (facultatif) et le cluster de bases de données. Comme une connexion existe déjà entre la fonction, le proxy et la base de données, les groupes de sécurité ne sont pas modifiés.
L'une des conditions suivantes s'applique :	L'une des conditions suivantes s'applique :	L'une des conditions suivantes s'applique :	RDS action: create new security groups

Configuration du groupe de sécurité RDS actuel	Configuration actuelle du groupe de sécurité Lambda	Configuration actuelle du groupe de sécurité du proxy	Action RDS
• Aucun groupe de sécurité n'est associé au cluster de bases de données avec un nom qui correspond au modèle `rds-lambda-` *n* ou si l'élément `TargetHealth` d'un proxy associé a pour valeur `AVAILABLE`. • Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-lambda-` *n* ou si l'élément `TargetHealth` d'un proxy associé a pour valeur `AVAILABLE`. Toutefois, aucun de ces groupes de sécurité ne peut être utilisé pour la connexion à la fonction Lambda.	• Aucun groupe de sécurité n'est associé à la fonction Lambda avec un nom qui correspond au modèle `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. • Un ou plusieurs groupes de sécurité sont associés à la fonction Lambda avec un nom qui correspond au modèle `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. Toutefois, Amazon RDS ne peut utiliser aucun de ces groupes de sécurité pour la connexion avec le cluster de bases de données. Amazon RDS ne peut pas utiliser comme destination un groupe de sécurité dépourvu de	• Aucun groupe de sécurité n'est associé au proxy avec un nom qui correspond au modèle `rdsproxy-lambda-` *n*. • Un ou plusieurs groupes de sécurité sont associés au proxy avec un nom qui correspond à `rdsproxy-lambda-` *n*. Toutefois, Amazon RDS ne peut utiliser aucun de ces groupes de sécurité pour la connexion avec le cluster de bases de données ou la fonction Lambda. Amazon RDS ne peut pas utiliser un groupe de sécurité dépourvu de règles entrantes et sortantes avec le groupe de sécurité VPC du cluster de bases de	

Configuration du groupe de sécurité RDS actuel	Configuration actuelle du groupe de sécurité Lambda	Configuration actuelle du groupe de sécurité du proxy	Action RDS
Amazon RDS ne peut pas utiliser comme source un groupe de sécurité dépourvu de toute règle entrante avec le groupe de sécurité VPC de la fonction Lambda ou du proxy. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié. Des exemples de modifications incluent l'ajout d'une règle ou la modification du port d'une règle existante.	toute règle sortante avec le groupe de sécurité VPC du cluster de bases de données ou du proxy. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.	données et de la fonction Lambda. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.	

Configuration du groupe de sécurité RDS actuel	Configuration actuelle du groupe de sécurité Lambda	Configuration actuelle du groupe de sécurité du proxy	Action RDS
Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-lambda-` *n* ou si l'élément `TargetHealth` d'un proxy associé a pour valeur `AVAILABLE` . Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité comprend une seule règle entrante avec le groupe de sécurité VPC de la fonction Lambda ou du proxy comme source.	Un ou plusieurs groupes de sécurité sont associés à la fonction Lambda avec un nom qui correspond au modèle `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. Toutefois, Amazon RDS ne peut utiliser aucun de ces groupes de sécurité pour la connexion avec le cluster de bases de données. Amazon RDS ne peut pas utiliser comme destination un groupe de sécurité dépourvu de toute règle sortante avec le groupe de sécurité VPC du cluster de bases de données ou du proxy. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.	Un ou plusieurs groupes de sécurité sont associés au proxy avec un nom qui correspond au modèle `rdsproxy-lambda-` *n*. Toutefois, Amazon RDS ne peut utiliser aucun de ces groupes de sécurité pour la connexion avec le cluster de bases de données ou la fonction Lambda. Amazon RDS ne peut pas utiliser un groupe de sécurité dépourvu de règles entrantes et sortantes avec le groupe de sécurité VPC du cluster de bases de données et de la fonction Lambda. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.	RDS action: create new security groups

Configuration du groupe de sécurité RDS actuel	Configuration actuelle du groupe de sécurité Lambda	Configuration actuelle du groupe de sécurité du proxy	Action RDS
Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-lambda-` *n* ou si l'élément `TargetHealth` d'un proxy associé a pour valeur `AVAILABLE` . Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité comprend une seule règle entrante avec le groupe de sécurité VPC de la fonction Lambda ou du proxy comme source.	Il existe un groupe de sécurité Lambda valide pour la connexion, mais il n'est pas associé à la fonction Lambda. Le nom de ce groupe de sécurité correspond au modèle `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. Il n'a pas été modifié. Il ne possède qu'une seule règle sortante avec le groupe de sécurité VPC du cluster de bases de données ou du proxy comme destination.	Il existe un groupe de sécurité de proxy valide pour la connexion, mais il n'est pas associé au proxy. Le nom de ce groupe de sécurité correspond au modèle `rdsproxy-lambda-` *n*. Il n'a pas été modifié. Il possède des règles entrantes et sortantes avec le groupe de sécurité VPC du cluster de bases de données et la fonction Lambda.	RDS action: associate Lambda security group

Configuration du groupe de sécurité RDS actuel	Configuration actuelle du groupe de sécurité Lambda	Configuration actuelle du groupe de sécurité du proxy	Action RDS
L'une des conditions suivantes s'applique : • Aucun groupe de sécurité n'est associé au cluster de bases de données avec un nom qui correspond au modèle `rds-lambda-` *n* ou si l'élément `TargetHealth` d'un proxy associé a pour valeur `AVAILABLE`. • Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-lambda-` *n* ou si l'élément `TargetHealth` d'un proxy associé a pour valeur `AVAILABLE`. Toutefois, Amazon RDS ne peut utiliser aucun de ces groupes	Un ou plusieurs groupes de sécurité sont associés à la fonction Lambda avec un nom qui correspond au modèle `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité ne possède qu'une seule règle sortante avec le groupe de sécurité VPC de l'instance de base de données ou du proxy comme destination.	Un ou plusieurs groupes de sécurité sont associés au proxy avec un nom qui correspond au modèle `rdsproxy-lambda-` *n*. Un groupe de sécurité qui correspond au modèle n'a pas été modifié. Ce groupe de sécurité possède des règles entrantes et sortantes avec le groupe de sécurité VPC du cluster de bases de données et la fonction Lambda.	RDS action: create new security groups

Configuration du groupe de sécurité RDS actuel	Configuration actuelle du groupe de sécurité Lambda	Configuration actuelle du groupe de sécurité du proxy	Action RDS
de sécurité pour la connexion avec la fonction Lambda ou le proxy. Amazon RDS ne peut pas utiliser comme source un groupe de sécurité dépourvu de toute règle entrante avec le groupe de sécurité VPC de la fonction Lambda ou du proxy. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.			

Configuration du groupe de sécurité RDS actuel	Configuration actuelle du groupe de sécurité Lambda	Configuration actuelle du groupe de sécurité du proxy	Action RDS
L'une des conditions suivantes s'applique : • Aucun groupe de sécurité n'est associé au cluster de bases de données avec un nom qui correspond au modèle `rds-lambda-` *n* ou si l'élément `TargetHealth` d'un proxy associé a pour valeur `AVAILABLE`. • Un ou plusieurs groupes de sécurité sont associés au cluster de bases de données avec un nom qui correspond au modèle `rds-lambda-` *n* ou si l'élément `TargetHealth` d'un proxy associé a pour valeur `AVAILABLE`. Toutefois, Amazon RDS ne peut utiliser aucun de ces groupes	L'une des conditions suivantes s'applique : • Aucun groupe de sécurité n'est associé à la fonction Lambda avec un nom qui correspond au modèle `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. • Un ou plusieurs groupes de sécurité sont associés à la fonction Lambda avec un nom qui correspond au modèle `lambda-rds-` *n* ou `lambda-rdsproxy-` *n*. Toutefois, Amazon RDS ne peut utiliser aucun de ces groupes de sécurité pour la connexion avec le cluster de bases de données. Amazon RDS ne peut pas utiliser comme	L'une des conditions suivantes s'applique : • Aucun groupe de sécurité n'est associé au proxy avec un nom qui correspond au modèle `rdsproxy-lambda-` *n*. • Un ou plusieurs groupes de sécurité sont associés au proxy avec un nom qui correspond à `rdsproxy-lambda-` *n*. Toutefois, Amazon RDS ne peut utiliser aucun de ces groupes de sécurité pour la connexion avec le cluster de bases de données ni la fonction Lambda. Amazon RDS ne peut pas utiliser un groupe de sécurité dépourvu de règles entrantes et sortantes	RDS action: create new security groups

Configuration du groupe de sécurité RDS actuel	Configuration actuelle du groupe de sécurité Lambda	Configuration actuelle du groupe de sécurité du proxy	Action RDS
de sécurité pour la connexion avec la fonction Lambda ou le proxy. Amazon RDS ne peut pas utiliser comme source un groupe de sécurité dépourvu de toute règle entrante avec le groupe de sécurité VPC de la fonction Lambda ou du proxy. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.	source un groupe de sécurité dépourvu de toute règle sortante avec le groupe de sécurité VPC du cluster de bases de données ou du proxy. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.	avec le groupe de sécurité VPC du cluster de bases de données et de la fonction Lambda. Amazon RDS ne peut pas non plus utiliser un groupe de sécurité qui a été modifié.	

Action RDS : créer de nouveaux groupes de sécurité

Amazon RDS entreprend les actions suivantes :

- Crée un nouveau groupe de sécurité qui correspond au modèle `rds-lambda-`*n* ou `rds-rdsproxy-`*n* (si vous choisissez d'utiliser un proxy RDS). Ce groupe de sécurité comprend une règle entrante avec le groupe de sécurité VPC de la fonction Lambda ou du proxy comme source. Ce groupe de sécurité est associé au cluster de bases de données et permet à la fonction ou au proxy d'accéder au cluster de bases de données.
- Crée un nouveau groupe de sécurité qui correspond au modèle `lambda-rds-`*n* ou `lambda-rdsproxy-`*n*. Ce groupe de sécurité possède une règle sortante avec le groupe de sécurité VPC du cluster de bases de données ou du proxy comme destination. Ce groupe de sécurité est associé

à la fonction Lambda et permet à cette dernière d'envoyer du trafic vers le cluster de bases de données ou d'envoyer du trafic via un proxy.

- Crée un nouveau groupe de sécurité qui correspond au modèle `rdsproxy-lambda-`*n*. Ce groupe de sécurité possède des règles entrantes et sortantes avec le groupe de sécurité VPC du cluster de bases de données et la fonction Lambda.

Action RDS : associer un groupe de sécurité Lambda

Amazon RDS associe le groupe de sécurité Lambda valide et existant à la fonction Lambda. Ce groupe de sécurité permet à la fonction Lambda d'envoyer du trafic vers le cluster de bases de données ou d'envoyer du trafic via un proxy.

Connexion automatique d'une fonction Lambda et d'un cluster de bases de données Aurora

Vous pouvez utiliser la console Amazon RDS pour connecter automatiquement une fonction Lambda à votre cluster de bases de données. Cela simplifie le processus de configuration d'une connexion entre ces ressources.

Vous pouvez également utiliser un proxy RDS pour inclure un proxy dans votre connexion. Les fonctions Lambda établissent des connexions de base de données courtes et fréquentes qui bénéficient du regroupement de connexions offert par le proxy RDS. Vous pouvez également utiliser toute authentification IAM que vous avez déjà configurée pour votre fonction Lambda, plutôt que de gérer les informations d'identification de base de données dans votre code d'application Lambda.

Vous pouvez connecter un cluster de bases de données existant aux fonctions Lambda nouvelles et existantes à l'aide de la page Configurer une connexion Lambda. Le processus de configuration configure automatiquement les groupes de sécurité requis pour vous.

Avant de configurer une connexion entre une fonction Lambda et un cluster de bases de données, assurez-vous que :

- Votre fonction Lambda et le cluster de bases de données se trouvent dans le même VPC.
- Vous disposez des autorisations appropriées pour votre compte d'utilisateur. Pour plus d'informations sur les exigences, consultez Vue d'ensemble de la connectivité automatique avec une fonction Lambda.

Si vous modifiez les groupes de sécurité après avoir configuré la connectivité, ces modifications peuvent affecter la connexion entre la fonction Lambda et le cluster de bases de données.

> **Note**
>
> Vous pouvez configurer automatiquement une connexion entre un cluster de bases de données et une fonction Lambda uniquement dans la AWS Management Console. Pour connecter une fonction Lambda, toutes les instances dans le cluster de bases de données doivent être dans l'état Disponible.

Pour connecter automatiquement une fonction Lambda et un cluster de bases de données

<result>

Une fois que vous avez confirmé la configuration, Amazon RDS commence le processus de connexion de votre fonction Lambda, du proxy RDS (si vous avez utilisé un proxy) et du cluster de bases de données. La console affiche la boîte de dialogue Détails de connexion, qui répertorie les modifications de groupe de sécurité qui permettent les connexions entre vos ressources.
</result>

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis choisissez le cluster de bases de données que vous voulez connecter à une fonction Lambda.
3. Pour Actions, choisissez Configurer la connexion Lambda.
4. Sur la page Configurer la connexion Lambda, sous Sélectionner une fonction Lambda, effectuez l'une des opérations suivantes :
   - Si vous avez déjà une fonction Lambda dans le même VPC que votre cluster de bases de données, choisissez Choisir une fonction existante, puis choisissez la fonction.
   - Si vous ne disposez pas d'une fonction Lambda dans le même VPC, choisissez Créer une nouvelle fonction, puis saisissez le Nom de la fonction. L'environnement d'exécution par défaut est défini sur Nodejs.18. Vous pouvez modifier les paramètres de votre nouvelle fonction Lambda dans la console Lambda après avoir terminé la configuration de la connexion.
5. (Facultatif) Sous Proxy RDS, sélectionnez Se connecter via un proxy RDS, puis effectuez l'une des opérations suivantes :

- Si vous souhaitez utiliser un proxy existant, choisissez Choisir un proxy existant, puis choisissez le proxy.
- Si vous n'avez pas de proxy et que vous souhaitez qu'Amazon RDS en crée un automatiquement pour vous, choisissez Créer un nouveau proxy. Ensuite, pour Informations d'identification de la base de données, effectuez l'une des opérations suivantes :

  a. Choisissez Nom d'utilisateur et mot de passe de base de données, puis saisissez le Nom d'utilisateur et le Mot de passe de votre cluster de bases de données.

  b. Choisissez Secret Secrets Manager. Ensuite, pour Sélectionner un secret, choisissez un secret AWS Secrets Manager. Si vous n'avez pas de secret Secrets Manager, choisissez Créer un nouveau secret Secrets Manager pour créer un nouveau secret. Après avoir créé le secret, pour Sélectionner un secret, choisissez le nouveau secret.

  Après avoir créé le nouveau proxy, choisissez Choisir un proxy existant, puis choisissez le proxy. Notez qu'il peut s'écouler un certain temps avant que votre proxy soit disponible pour la connexion.

6. (Facultatif) Développez Récapitulatif de la connexion et vérifiez les mises à jour en surbrillance pour vos ressources.
7. Choisissez Set up (Configurer).

Affichage des ressources de calcul connectées

Vous pouvez utiliser la AWS Management Console pour visualiser les fonctions Lambda connectées à votre cluster de bases de données. Les ressources affichées incluent les connexions de ressources de calcul qu'Amazon RDS a configurées automatiquement.

Les ressources de calcul répertoriées n'incluent pas celles qui sont connectées manuellement au cluster de bases de données. Par exemple, vous pouvez autoriser une ressource de calcul à accéder manuellement à votre cluster de bases de données en ajoutant une règle à votre groupe de sécurité VPC associé à la base de données.

Pour que la console répertorie une fonction Lambda, les conditions suivantes doivent s'appliquer :

- Le nom du groupe de sécurité associé à la ressource de calcul correspond au modèle `lambda-rds-`*n* ou `lambda-rdsproxy-`*n* (où *n* est un nombre).

- Le groupe de sécurité associé à la ressource de calcul possède une règle sortante avec la plage de ports définie sur le port du cluster de bases de données ou d'un proxy associé. La destination de la règle sortante doit être définie sur un groupe de sécurité associé au cluster de bases de données ou un proxy associé.
- Si la configuration inclut un proxy, le nom du groupe de sécurité attaché au proxy associé à votre base de données correspond au modèle `rdsproxy-lambda-`*n* (où *n* est un nombre).
- Le groupe de sécurité associé à la fonction possède une règle sortante avec le port défini sur le port utilisé par le cluster de bases de données ou le proxy associé. La destination doit être définie sur un groupe de sécurité associé au cluster de bases de données ou au proxy associé.

Pour afficher les ressources de calcul automatiquement connectées à un cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis choisissez le cluster de bases de données.
3. Dans l'onglet Connectivité et sécurité, examinez les ressources de calcul sous Ressources de calcul connectées.

Modification d’un cluster de bases de données Amazon Aurora

Vous pouvez modifier les paramètres d’un cluster de bases de données pour effectuer certaines tâches, comme modifier sa période de rétention des sauvegardes ou son port de base de données. Vous pouvez également modifier les instances de base de données d’un cluster de bases de données pour effectuer certaines tâches, comme modifier sa classe d’instance de base de données ou activer Performance Insights. Cette rubrique vous guide tout au long du processus de modification d’un cluster de bases de données Aurora et de ses instances, et décrit leurs paramètres.

Nous vous recommandons de tester les modifications apportées à un cluster ou une instance de base de données de test avant de modifier un cluster ou une instance de base de données de production afin de bien comprendre l’impact de chaque modification. Cela est particulièrement important lors de la mise à niveau de versions de base de données.

Rubriques

Modification du cluster de bases de données à partir de la console, de l’CLI (CLI) et de l’API

Vous pouvez modifier un cluster de bases de données à partir d’AWS Management Console, de l’AWS CLI ou de l’API RDS.

> **Note**
>
> La plupart des modifications peuvent être appliquées immédiatement ou au cours de la prochaine fenêtre de maintenance planifiée. Certaines modifications, telles que l’activation de la protection contre la suppression, sont appliquées immédiatement, quel que soit le moment où vous choisissez de les appliquer.
> La modification du mot de passe principal dans la AWS Management Console est toujours appliquée immédiatement.

Si vous utilisez des points de terminaison SSL et modifiez l'identifiant du cluster de bases de données, arrêtez et redémarrez le cluster de bases de données pour mettre à jour les points de terminaison SSL. Pour plus d'informations, consultez Arrêt et démarrage d'un cluster de bases de données Amazon Aurora.

Console

Pour modifier un cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis sélectionnez le cluster de bases de données que vous souhaitez modifier.
3. Sélectionnez Modify. La page Modify DB cluster (Modifier le cluster DB) s'affiche.
4. Modifiez les paramètres de votre choix. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour Amazon Aurora.

   > **Note**
   >
   > Dans AWS Management Console, certaines modifications au niveau de l'instance s'appliquent uniquement à l'instance de base de données active, tandis que d'autres s'appliquent à l'intégralité du cluster de bases de données. Pour savoir si un paramètre s'applique à l'instance de base de données ou au cluster de bases de données, consultez pour en connaître la porté Paramètres pour Amazon Aurora. Pour modifier un paramètre qui modifie l'intégralité du cluster de bases de données au niveau de l'instance dans AWS Management Console, suivez les instructions de la section Modification d'une instance de base de données dans un cluster de bases de données.

5. Lorsque tous les changements vous conviennent, choisissez Continuer et vérifiez le résumé des modifications.
6. Pour immédiatement appliquer les modifications, sélectionnez Apply Immediately (Appliquer immédiatement).
7. Sur la page de confirmation, examinez vos modifications. Si elles sont correctes, choisissez Modifier le cluster pour enregistrer vos modifications.

Sinon, choisissez Retour pour modifier vos modifications, ou choisissez Annuler pour les annuler.

AWS CLI

Pour modifier un cluster de bases de données à partir de l'AWS CLI, appelez la commande modify-db-cluster. Spécifiez l'identifiant du cluster de bases de données, ainsi que les valeurs des paramètres que vous voulez modifier. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour Amazon Aurora.

> **Note**
>
> Certains paramètres s'appliquent uniquement aux instances de base de données. Pour modifier ces paramètres, suivez les instructions de la section Modification d'une instance de base de données dans un cluster de bases de données.

Example

La commande suivante modifie `mydbcluster` en définissant la période de rétention des sauvegardes sur 1 semaine (7 jours).

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --backup-retention-period 7
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --backup-retention-period 7
```

API RDS

Pour modifier un cluster de bases de données à partir de l'API Amazon RDS, appelez l'opération ModifyDBCluster. Spécifiez l'identifiant du cluster de bases de données, ainsi que les valeurs des

paramètres que vous voulez modifier. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour Amazon Aurora.

> **Note**
>
> Certains paramètres s'appliquent uniquement aux instances de base de données. Pour modifier ces paramètres, suivez les instructions de la section Modification d'une instance de base de données dans un cluster de bases de données.

Modification d'une instance de base de données dans un cluster de bases de données

Vous pouvez modifier une instance de base de données dans un cluster de bases de données à partir d'AWS Management Console, de l'AWS CLI ou de l'API RDS.

Quand vous modifiez une instance de base de données, vous pouvez appliquer immédiatement les modifications. Pour appliquer immédiatement les modifications, sélectionnez l'option Apply Immediately (Appliquer immédiatement) dans la AWS Management Console, utilisez le paramètre `--apply-immediately` lorsque vous appelez l'AWS CLI, ou définissez le paramètre `ApplyImmediately` sur `true` lorsque vous utilisez l'API Amazon RDS.

Si vous ne choisissez pas d'appliquer les modifications immédiatement, les modifications sont reportées à la fenêtre de maintenance suivante. Lors de la prochaine fenêtre de maintenance, toutes ces modifications différées sont appliquées. Si vous choisissez d'appliquer les modifications immédiatement, vos nouvelles modifications et toutes les modifications précédemment différées sont appliquées.

Pour voir les modifications en attente pour la prochaine fenêtre de maintenance, utilisez la commande describe-db-clusters de l'AWS CLI et vérifiez le champ `PendingModifiedValues`.

> **Important**
>
> Si les modifications en attente exigent une durée d'indisponibilité, le fait de choisir de les Appliquer immédiatement peut entraîner une indisponibilité imprévue de l'instance de base de données. Il n'y a pas de durée d'indisponibilité pour les autres instances de base de données du cluster de bases de données.
>
> Les modifications que vous différez ne sont pas répertoriées dans la sortie de la commande CLI `describe-pending-maintenance-actions`. Les actions de maintenance incluent

uniquement les mises à niveau système que vous planifiez pour la prochaine fenêtre de maintenance.

Console

Pour modifier une instance de base de données dans un cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis sélectionnez l'instance de base de données que vous souhaitez modifier.
3. Pour Actions, choisissez Modify (Modifier). La page Modifier l'instance de base de données s'affiche.
4. Modifiez les paramètres de votre choix. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour Amazon Aurora.

   > **Note**
   >
   > Certains paramètres s'appliquent à l'intégralité du cluster de bases de données et doivent être modifiés au niveau du cluster. Pour modifier ces paramètres, suivez les instructions de la section Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API.
   > Dans AWS Management Console, certaines modifications au niveau de l'instance s'appliquent uniquement à l'instance de base de données active, tandis que d'autres s'appliquent à l'intégralité du cluster de bases de données. Pour savoir si un paramètre s'applique à l'instance de base de données ou au cluster de bases de données, consultez pour en connaître la porté Paramètres pour Amazon Aurora.

5. Lorsque tous les changements vous conviennent, choisissez Continuer et vérifiez le résumé des modifications.
6. Pour immédiatement appliquer les modifications, sélectionnez Apply Immediately (Appliquer immédiatement).
7. Sur la page de confirmation, examinez vos modifications. Si elles sont correctes, choisissez Modify DB instance (Modifier l'instance de base de données) pour enregistrer vos modifications.

Sinon, choisissez Retour pour modifier vos modifications, ou choisissez Annuler pour les annuler.

AWS CLI

Pour modifier une instance de base de données dans un cluster de bases de données à partir de l'AWS CLI, appelez la commande modify-db-instance. Spécifiez l'identifiant d'instance de base de données et les valeurs des paramètres que vous souhaitez modifier. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour Amazon Aurora.

> **Note**
>
> Certains paramètres s'appliquent à l'intégralité du cluster de bases de données. Pour modifier ces paramètres, suivez les instructions de la section Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API.

Example

Le code suivant modifie `mydbinstance` en définissant la classe d'instance de base de données sur `db.r4.xlarge`. Les modifications sont appliquées pendant le créneau de maintenance suivant à l'aide de `--no-apply-immediately`. Pour appliquer les modifications immédiatement, utilisez `--apply-immediately`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --db-instance-class db.r4.xlarge \
    --no-apply-immediately
```

Pour Windows :

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --db-instance-class db.r4.xlarge ^
    --no-apply-immediately
```

API RDS

Pour modifier une instance de base de données à l'aide de l'API Amazon RDS, appelez l'opération ModifyDBInstance. Spécifiez l'identifiant d'instance de base de données et les valeurs des paramètres que vous souhaitez modifier. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour Amazon Aurora.

> **Note**
>
> Certains paramètres s'appliquent à l'intégralité du cluster de bases de données. Pour modifier ces paramètres, suivez les instructions de la section Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API.

Modification du mot de passe de l'utilisateur principal de la base de données.

Vous pouvez utiliser la commande AWS Management Console pour modifier le mot de passe de l'utilisateur principal.

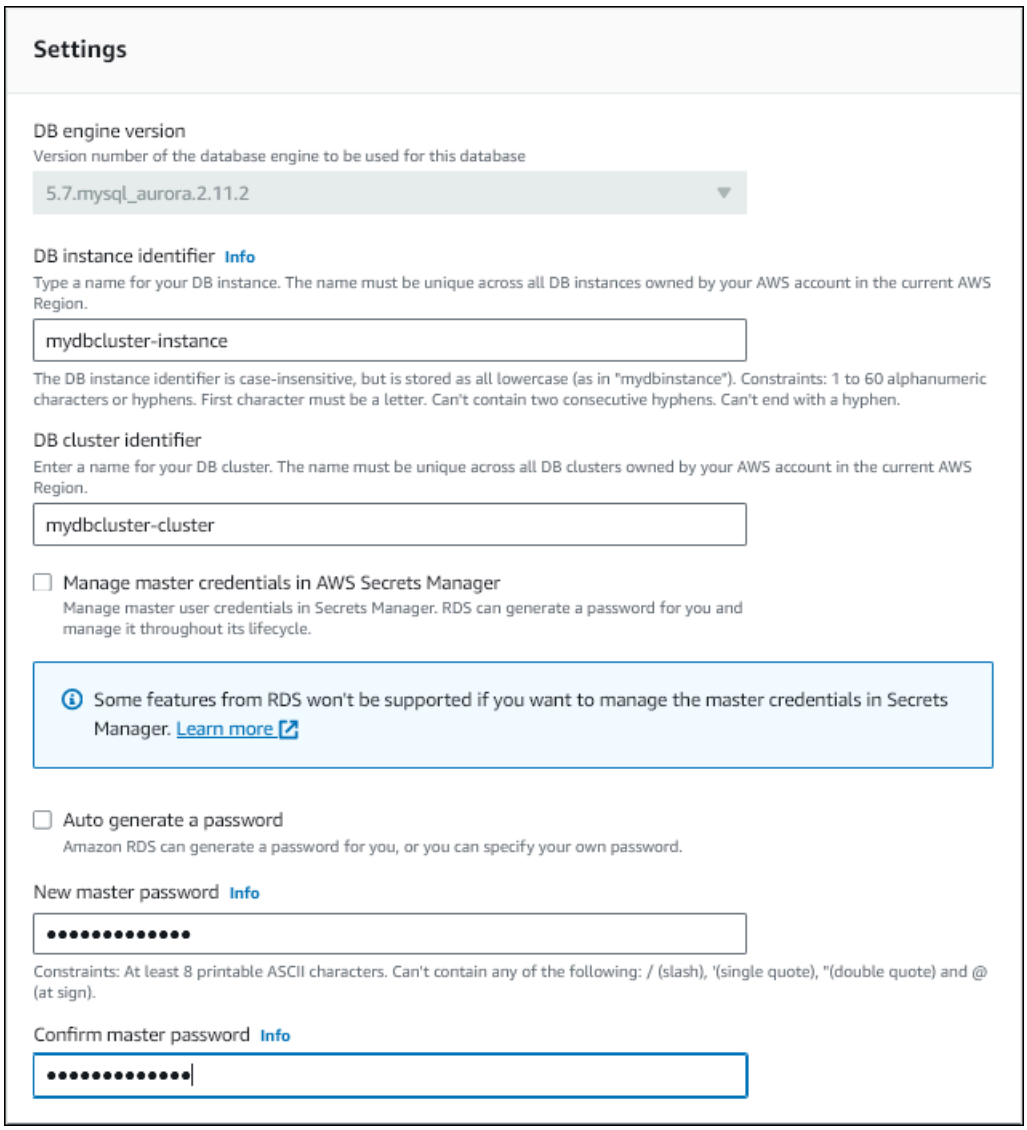
Console

Vous devez modifier l'instance de base de données d'enregistreur pour changer le mot de passe de l'utilisateur principal à l'aide de la AWS Management Console.

Pour modifier le mot de passe de l'utilisateur principal

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis sélectionnez l'instance de base de données que vous souhaitez modifier.
3. Pour Actions, choisissez Modify (Modifier).

   La page Modifier l'instance de base de données s'affiche.
4. Indiquez un nouveau mot de passe dans Nouveau mot de passe principal.
5. Dans Confirmer le mot de passe principal, entrez le même mot de passe.



6. Choisissez Continuer et vérifiez le récapitulatif des modifications.

> **Note**
>
> Les modifications de mot de passe sont toujours appliquées immédiatement.

7. Sur la page de confirmation, choisissez Modifier l'instance de base de données.

Interface de ligne de commande (CLI)

Appelez la commande modify-db-cluster pour modifier le mot de passe de l'utilisateur principal à l'aide de l'AWS CLI. Spécifiez l'identifiant du cluster de bases de données et le nouveau mot de passe, comme indiqué dans les exemples suivants.

Il n'est pas nécessaire de spécifier `--apply-immediately|--no-apply-immediately`, car les modifications de mot de passe sont toujours appliquées immédiatement.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --master-user-password mynewpassword
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --master-user-password mynewpassword
```

Paramètres pour Amazon Aurora

Le tableau suivant contient des détails sur les paramètres que vous pouvez modifier, les méthodes pour les modifier, ainsi que leur portée. La portée détermine si le paramètre s'applique à l'intégralité du cluster de bases de données ou s'il peut être défini uniquement pour des instances de base de données spécifiques.

> **Note**
>
> Des paramètres supplémentaires sont disponibles si vous modifiez un cluster de bases de données Aurora Serverless v1 ou Aurora Serverless v2. Pour plus d'informations sur ces paramètres, consultez Modification d'un cluster de bases de données Aurora Serverless v1 et Gestion des clusters de bases de données Aurora Serverless v2.
> Certains paramètres ne sont pas disponibles pour Aurora Serverless v1 et Aurora Serverless v2 en raison de leurs limitations. Pour plus d'informations, consultez Limites d Aurora Serverless v1 et Exigences et limites relatives à Aurora Serverless v2.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Mise à niveau automatique de versions mineures Indiquez si l'instance de base de données doit recevoir automatiquement les mises à niveau des versions mineures préférées du moteur dès qu'elles sont disponibles. Les mises à niveau sont installées uniquement pendant votre créneau de maintenance planifié. Pour plus d'informations sur les mises à jour de moteur, consultez Mises à jour du moteur de base de données pour Amazon Aurora PostgreSQL et Mises à jour du moteur de base de données pour Amazon Aurora MySQL. Pour plus d'informations sur le paramètre Mise à niveau automatique	**Note** Ce paramètre est activé par défaut. Pour chaque nouveau cluster, choisissez la valeur appropriée pour ce paramètre en fonction de son importance, de sa durée de vie prévue et du nombre de tests de vérification effectués après chaque mise à niveau. Lorsque vous modifiez ce paramètre, effectuez cette modification pour chaque instance de base de données de votre cluster Aurora. Si ce paramètre est	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification. Des interruptions de service se produisent au cours des fenêtres de maintenance suivantes lorsque Aurora applique les mises à niveau automatiques.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
de versions mineures pour Aurora MySQL, consultez Activation des mises à niveau automatiques entre versions mineures Aurora MySQL.	désactivé dans une instance de base de données de votre cluster, le cluster n'est pas automatiquement mis à niveau. À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À l'aide de l'AWS CLI, exécutez la commande modify-db-instance et définissez l'option `--auto-minor-version-upgrade|--no-auto-minor-version-upgrade` . À l'aide de l'API RDS, appelez ModifyDBInstance et définissez le paramètre `AutoMinorVersionUpgrade` .		

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Période de rétention des sauvegardes Le nombre de jours de conservation des sauvegardes automatiques. La valeur minimale est de 1. Pour plus d'informations, consultez Sauvegardes.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À l'aide de l'AWS CLI, exécutez la commande modify-db-cluster et définissez l'option `--backup-retention-period`. À l'aide de l'API RDS, appelez ModifyDBCluster et définissez le paramètre `BackupRetentionPeriod` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Fenêtre de sauvegarde (Heure de début) L'intervalle de temps pendant lequel des sauvegardes automatiques de vos bases de données sont effectuées. Le créneau de sauvegarde correspond à une heure de début en heure UTC (Universal Coordinated Time) et une durée en heures. Les sauvegardes Aurora sont continues et incrémentielles, mais la fenêtre de sauvegarde permet de créer une sauvegarde système quotidienne qui est conservée pendant la période de rétention des sauvegardes. Vous pouvez la copier pour la conserver en dehors de la période de rétention. La fenêtre de maintenance et la	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À l'aide de l'AWS CLI, exécutez la commande modify-db-cluster et définissez l'option `--preferred-backup-window`. À l'aide de l'API RDS, appelez ModifyDBCluster et définissez le paramètre `PreferredBackupWindow` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
fenêtre de sauvegarde de l'instance de cluster ne peuvent pas se chevaucher. Pour plus d'informations, consultez Fenêtre de sauvegarde.			
Paramètres de capacité Propriétés de mise à l'échelle d'un cluster de bases de données Aurora Serverless v1. Vous ne pouvez modifier les propriétés de mise à l'échelle des clusters de bases de données qu'en mode moteur de base de données `serverless` . Pour plus d'informations sur Aurora Serverless v1, consultez Utilisation d'Amazon Aurora Serverless v1.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À l'aide de l'AWS CLI, exécutez la commande modify-db-cluster et définissez l'option `--scaling-configuration` . À l'aide de l'API RDS, appelez ModifyDBCluster et définissez le paramètre `ScalingConfiguration` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification. La modification a lieu immédiatement. Ce paramètre ignore le paramètre Appliquer immédiatement.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Autorité de certification L'autorité de certification (CA) pour le certificat de serveur utilisé par l'instance de base de données.	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À l'aide de l'AWS CLI, exécutez la commande modify-db-instance et définissez l'option `--ca-certificate-identifier` . À l'aide de l'API RDS, appelez ModifyDBInstance et définissez le paramètre `CACertificateIdentifier` .	Uniquement l'instance de base de données spécifiée	Une panne survient uniquement si le moteur de la base de données ne prend pas en charge la rotation sans redémarrage. Vous pouvez utiliser la commande AWS CLI describe-db-engine-versions pour déterminer si le moteur de base de données prend en charge la rotation sans redémarrage.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Configuration du stockage en cluster Type de stockage pour le cluster de bases de données : Aurora I/O-Optimized ou Aurora Standard. Pour plus d'informations, consultez Configurations de stockage pour les clusters de bases de données Amazon Aurora.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À l'aide de l'AWS CLI, exécutez la commande modify-db-cluster et définissez l'option `--storage-type` . À l'aide de l'API RDS, appelez ModifyDBCluster et définissez le paramètre `StorageType` .	L'intégralité du cluster de bases de données	La modification du type de stockage d'un cluster de bases de données Aurora PostgreSQL avec des classes d'instance Optimized Reads entraîne une panne. Cela ne se produit pas lors de la modification des types de stockage pour les clusters ayant d'autres types de classes d'instance. Pour en savoir plus sur les types de classes d'instance de base de données, consultez Types de classes d'instance de base de données.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Copier les balises aux instantanés Sélectionnez cette option pour spécifier que les balises définies pour ce cluster de bases de données sont copiées vers les instantanés de bases de données créés à partir de ce cluster de bases de données. Pour plus d'informations, consultez Marquage des ressources Amazon Aurora et Amazon RDS.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez https://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-cluster.html et définissez l'option `--copy-tags-to-snapshot` ou `--no-copy-tags-to-snapshot` . À partir de l'API RDS, appelez https://docs.aws.amazon.com/AmazonRDS/latest/APIReference/API_ModifyDBCluster.html et définissez le paramètre `CopyTagsToSnapshot` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
API de données Vous pouvez accéder à Aurora Serverless v1 avec des applications basées sur des services web, dont AWS Lambda et AWS AppSync. Ce paramètre s'applique uniquement à un cluster de bases de données Aurora Serverless v1. Pour plus d'informations, consultez Utilisation de l'API de données Amazon RDS.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--enable-http-endpoint` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `EnableHttpEndpoint` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Authentification de base de données L'authentification de base de données que vous souhaitez utiliser. Pour MySQL : • Choisissez Authentification par mot de passe pour authentifier les utilisateurs de base de données avec des mots de passe de base de données uniquement. • Choisissez Mot de passe et authentification de base de données IAM pour authentifier les utilisateurs de bases de données avec des mots de passe de bases de données et des informations d'identification utilisateur via des utilisateurs et rôles	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À l'aide de la AWS CLI , exécutez modify-db-cluster et définissez les options suivantes : • Pour l'authentification IAM, définissez l'option `--enable-iam-database-authentication|--no-enable-iam-database-authentication` . • Pour l'authentification Kerberos, définissez les options `--domain` et `--domain-iam-role-name` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
IAM. Pour plus d'informations, consultez Authentification de base de données IAM. Pour PostgreSQL : • Choisissez IAM database authentication (Authentification de base de données IAM) pour authentifier les utilisateurs de bases de données avec des mots de passe de bases de données et des informations d'identification utilisateur via des utilisateurs et rôles. Pour plus d'informations, consultez Authentification de base de données IAM. • Choisissez Authentification Kerberos pour authentifier les mots de passe de bases de données	À l'aide de l'API RDS, appelez ModifyDBCluster et définissez les paramètres suivants : • Pour l'authentification IAM, définissez le paramètre `EnableIAMDatabaseAuthentication` . • Pour l'authentification Kerberos, définissez les paramètres `Domain` et `DomainIAMRoleName` .		

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
et les informations d'identification utilisateur à l'aide de l'authentification Kerberos. Pour plus d'informations, consultez Utilisation de l'authentification Kerberos avec Aurora PostgreSQL.			
Port de la base de données Port que vous souhaitez utiliser pour accéder au cluster de bases de données.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--port`. À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `Port`.	L'intégralité du cluster de bases de données	Une interruption de service a lieu pendant cette modification. Toutes les instances de base de données du cluster de bases de donnes sont redémarrées immédiatement.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Identifiant du cluster de bases de données Identifiant du cluster de bases de données. Cette valeur est stockée sous la forme d'une chaîne en minuscules. Lorsque vous modifiez l'identifiant du cluster de bases de données, les points de terminaison du cluster de bases de données changent. Les points de terminaison des instances de base de données du cluster de bases de données ne changent pas.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--new-db-cluster-identifier` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `NewDBClusterIdentifier` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Groupe de paramètres de cluster de bases de données Groupe de paramètres de cluster de bases de données que vous souhaitez associer au cluster de bases de données. Pour plus d'informations, consultez Groupes de paramètres pour Amazon Aurora.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--db-cluster-parameter-group-name` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `DBClusterParameterGroupName` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification. Lorsque vous modifiez le groupe de paramètres, les modifications apportées à certains paramètres s'appliquent immédiatement aux instances de base de données du cluster de bases de données, sans redémarrage. Les modifications apportées à d'autres paramètres s'appliquent uniquement après le redémarrage des instances de base de données dans le cluster de bases de données.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Classe d'instance de base de données La classe d'instance de base de données que vous souhaitez utiliser. Pour plus d'informations, consultez Classes d'instance de base de données Amazon Aurora.	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-instance` et définissez l'option `--db-instance-class` . À partir de l'API RDS, appelez `ModifyDBInstance` et définissez le paramètre `DBInstanceClass` .	Uniquement l'instance de base de données spécifiée	Une interruption de service a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Identifiant d'instance de base de données Identifiant de l'instance de base de données. Cette valeur est stockée sous la forme d'une chaîne en minuscules.	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-instance` et définissez l'option `--new-db-instance-identifier`. À partir de l'API RDS, appelez `ModifyDBInstance` et définissez le paramètre `NewDBInstanceIdentifier`.	Uniquement l'instance de base de données spécifiée	Une durée d'indisponibilité a lieu pendant cette modification. RDS redémarre l'instance de base de données pour mettre à jour les éléments suivants : • Aurora MySQL — colonne `SERVER_ID` dans la table `information_schema.replica_host_status` • Aurora PostgreSQL — colonne `server_id` dans la fonction `aurora_replica_status()`

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Groupe de paramètres de base de données Groupe de paramètres de base de données que vous souhaitez associer à l'instance de base de données. Pour plus d'informations, consultez Groupes de paramètres pour Amazon Aurora.	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-instance` et définissez l'option `--db-parameter-group-name` . À partir de l'API RDS, appelez `ModifyDBInstance` et définissez le paramètre `DBParameterGroupName` .	Uniquement l'instance de base de données spécifiée	Aucune interruption de service n'a lieu pendant cette modification. Lorsque vous associez un nouveau groupe de paramètres de base de données à une instance de base de données, les paramètres statiques et dynamiques modifiés sont appliqués uniquement après que l'instance de base de données est redémarrée. Toutefois, si vous modifiez des paramètres dynamiques dans le groupe de paramètres de base de données après l'avoir associé à l'instance de base de données, ces modifications sont appliquées immédiatement sans redémarrage. Pour plus d'informations, consultez

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
			Groupes de paramètres pour Amazon Aurora et Redémarrage d'un cluster de bases de données Amazon Aurora ou d'une instance de base de données Amazon Aurora.
Deletion protection (Protection contre la suppression) Sélectionnez Enable deletion protection (Activer la protection de la suppression) pour empêcher la suppression de votre cluster de bases de données. Pour plus d'informations, consultez Protection contre la suppression pour les clusters Aurora.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--deletion-protection|--no-deletion-protection` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `DeletionProtection` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Version du moteur Version du moteur de base de données que vous souhaitez utiliser. Avant de mettre à niveau votre cluster de bases de données de production, nous vous recommandons de tester le processus de mise à niveau sur un cluster de bases de données de test pour vérifier sa durée et valider vos applications.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--engine-version` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `EngineVersion` .	L'intégralité du cluster de bases de données	Une interruption de service a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Surveillance améliorée Activer la surveillance améliorée permet d'activer la collecte des métriques en temps réel pour le système d'exploitation sur lequel votre instance de base de données s'exécute. Pour plus d'informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-instance` et définissez les options `--monitoring-role-arn` et `--monitoring-interval` . À partir de l'API RDS, appelez `ModifyDBInstance` et définissez les paramètres `MonitoringRoleArn` et `MonitoringInterval` .	Uniquement l'instance de base de données spécifiée	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Exportations des journaux Sélectionnez les types de journal à publier dans Amazon CloudWatch Logs. Pour plus d'informations, consultez Fichiers journaux de base de données Aurora MySQL.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--cloudwatch-logs-export-configuration` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `CloudwatchLogsExportConfiguration` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Fenêtre de maintenance L'intervalle de temps pendant lequel la maintenance du système a lieu. La maintenance du système inclut les mises à niveau, le cas échéant. Le fenêtre de maintenance correspond à une heure de début en heure UTC (Universal Coordinated Time) et une durée en heures. Si vous définissez la fenêtre sur l'heure actuelle, il doit y avoir au moins 30 minutes entre l'heure actuelle et la fin du créneau afin de garantir l'application des modifications en attente. Vous pouvez définir la fenêtre de maintenance du cluster de bases de données et de chaque instance	Pour modifier la fenêtre de maintenance du cluster de bases de données à partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. Pour modifier la fenêtre de maintenance d'une instance de base de données à partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. Pour modifier la fenêtre de maintenance du cluster de bases de données à partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--preferred-`	L'intégralité du cluster de bases de données ou une seule instance de base de données	S'il y a en attente une ou plusieurs actions entraînant une panne, et que la fenêtre de maintenance est modifiée pour inclure l'heure actuelle, alors les actions en attente sont appliquées immédiatement et une panne se produit.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
de base de données du cluster de façon indépendante. Lorsque la portée d'une modification s'étend à l'intégralité du cluster de bases de données, la modification s'effectue pendant la fenêtre de maintenance du cluster. Lorsque la portée d'une modification se limite à une instance de base de données, la modification s'effectue pendant la fenêtre de maintenance de l'instance de base de données. La fenêtre de maintenance et la fenêtre de sauvegarde de l'instance de cluster ne peuvent pas se chevaucher. Pour plus d'informations, consultez Fenêtre de maintenance Amazon RDS.	`maintenance-window` . Pour modifier la fenêtre de maintenance d'une instance de base de données à partir de l'AWS CLI, exécutez `modify-db-instance` et définissez l'option `--preferred-maintenance-window` . Pour modifier la fenêtre de maintenance du cluster de bases de données à partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `PreferredMaintenanceWindow` . Pour modifier la fenêtre de maintenance d'une instance de base de données à partir de l'API RDS, appelez `ModifyDBInstance` et définissez		

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
	le paramètre `PreferredMaintenanceWindow`.		

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Gérer les informations d'identification principales dans AWS Secrets Manager Sélectionnez Gérer les informations d'identification principales dans AWS Secrets Manager pour gérer le mot de passe d'utilisateur principal dans un secret, dans Secrets Manager. Vous pouvez éventuellement choisir une clé KMS à utiliser pour protéger le secret. Choisissez l'une des clés KMS de votre compte ou entrez la clé d'un autre compte. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager. Si Aurora gère déjà le mot de passe de	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez les options `--manage-master-user-password | --no-manage-master-user-password` et `--master-user-secret-kms-key-id` . Pour effectuer immédiatement la rotation du mot de passe de l'utilisateur principal, définissez l'option `--rotate-master-user-password` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez les paramètres `ManageMas`	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
l'utilisateur principal pour le cluster de bases de données, vous pouvez effectuer la rotation du mot de passe de l'utilisateur principal en choisissant Rotate secret immediately (Effectuer immédiatement une rotation du secret). Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager.	`terUserPassword` et `MasterUserSecretKmsKeyId` . Pour effectuer immédiatement la rotation du mot de passe de l'utilisateur principal, définissez le paramètre `RotateMasterUserPassword` sur `true`.		

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Network type (Type de réseau) Les protocoles d'adressage IP pris en charge par le cluster de la base de données. IPv4 pour spécifier que les ressources peuvent communiquer avec le cluster de bases de données uniquement via le protocole d'adressage IPv4. Dual-stack mode (Mode double pile) pour spécifier que les ressources peuvent communiquer avec le cluster de bases de données sur IPv4, IPv6, ou les deux. Utilisez le mode double pile si vous possédez des ressources qui doivent communiquer avec votre cluster de bases de données via le protocole d'adressa	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--network-type` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `NetworkType` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
ge IPv6. Pour utiliser le mode double pile, assurez-vous qu'au moins deux sous-réseaux couvrant deux zones de disponibilité prennent en charge le protocole réseau IPv4 et IPv6. Veillez également à associer un bloc CIDR IPv6 aux sous-réseaux du groupe de sous-réseaux de base de données que vous spécifiez. Pour plus d'informations, consultez Adressage IP Amazon Aurora.			

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
New master password Le mot de passe de votre utilisateur principal. • Pour Aurora MySQL, le mot de passe doit contenir entre 8 et 41 caractères ASCII imprimables. • Pour Aurora PostgreSQL, il doit contenir entre 8 et 99 caractères ASCII imprimables. • Il ne peut pas contenir `/`, `"`, `@` ou un espace.	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--master-user-password` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `MasterUserPassword` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Performance Insights Indiquez si vous activez Performance Insights, outil qui surveille la charge de votre instance de base de données pour vous permettre d'analyser et de résoudre les problèmes de performances de votre base de données. Pour plus d'informations, consultez Surveillance de la charge de la base de données avec Performance Insights sur .	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-instance` et définissez l'option `--enable-performance-insights|--no-enable-performance-insights` . À partir de l'API RDS, appelez `ModifyDBInstance` et définissez le paramètre `EnablePerformanceInsights` .	Uniquement l'instance de base de données spécifiée	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Performance Insights AWS KMS key Identifiant de AWS KMS key pour le chiffrement des données de Performance Insights. L'identifiant de clé KMS est l'ARN (Amazon Resource Name), l'identifiant de clé ou l'alias de clé pour la clé KMS. Pour plus d'informations, consultez Activation ou désactivation de l'Analyse des performances pour Aurora.	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-instance` et définissez l'option `--performance-insights-kms-key-id` . À partir de l'API RDS, appelez `ModifyDBInstance` et définissez le paramètre `PerformanceInsightsKMSKeyId` .	Uniquement l'instance de base de données spécifiée	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Performance Insights retention period (Période de rétention d'Analyse des performances) Durée de conservation, en jours, des données de Performance Insights. Le paramètre de conservation est Par défaut (7 jours). Pour conserver vos données de performance plus longtemps, indiquez 1 à 24 mois. Pour obtenir plus d'informations sur les périodes de conservation, consultez Tarification et conservation des données pour Performance Insights. Pour plus d'informations, consultez Activation ou désactivation de l'Analyse des performances pour Aurora.	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-instance` et définissez l'option `--performance-insights-retention-period` . À partir de l'API RDS, appelez `ModifyDBInstance` et définissez le paramètre `PerformanceInsightsRetentionPeriod` .	Uniquement l'instance de base de données spécifiée	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Promotion tier (Niveau de promotion) Valeur qui spécifie l'ordre dans lequel un réplica Aurora est promu comme instance principale dans un cluster de bases de données après un échec de l'instance principale existante. Pour plus d'informations, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-instance` et définissez l'option `--promotion-tier` . À partir de l'API RDS, appelez `ModifyDBInstance` et définissez le paramètre `PromotionTier` .	Uniquement l'instance de base de données spécifiée	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Accès public Publicly accessible (Accessible publiquement) dote l'instance de base de données d'une adresse IP publique, ce qui signifie qu'elle est accessible en dehors du VPC. Pour être accessible au public, l'instance de base de données doit aussi se trouver dans un sous-réseau public du VPC. Not publicly accessible (Non accessible publiquement) rend l'instance de base de données accessible uniquement à partir de l'intérieur du VPC. Pour plus d'informations, consultez Masquer un cluster de bases de données dans un VPC depuis Internet. Pour se connecter à une instance de base	À partir d'AWS Management Console, Modification d'une instance de base de données dans un cluster de bases de données. À partir de l'AWS CLI, exécutez `modify-db-instance` et définissez l'option `--publicly-accessible|--no-publicly-accessible` . À partir de l'API RDS, appelez `ModifyDBInstance` et définissez le paramètre `PubliclyAccessible` .	Uniquement l'instance de base de données spécifiée	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
de données hors de son Amazon VPC, l'instance de base de données doit être accessible au public, l'accès doit être accordé en utilisant les règles entrantes du groupe de sécurité de l'instance de base de données et d'autres exigences doivent être respectées. Pour plus d'informations, consultez Impossible de se connecter à l'instance de base de données Amazon RDS. Si votre instance de base de données n'est pas accessible publiquement, vous pouvez également utiliser une connexion AWS Site-to-Site VPN ou une connexion Direct Connect pour y accéder à partir d'un réseau privé. Pour plus d'informations, consultez Confident			

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
ialité du trafic inter-réseau.			
Paramètres de capacité Serverless v2 Capacité de base de données d'un cluster de bases de données Aurora Serverless v2, mesurée en unités de capacité Aurora (ACU). Pour plus d'informations, consultez Définition de la plage de capacité Aurora Serverless v2 d'un cluster.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--serverless-v2-scaling-configuration` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `ServerlessV2ScalingConfiguration` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification. La modification a lieu immédiatement. Ce paramètre ignore le paramètre Appliquer immédiatement.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Groupe de sécurité Groupe de sécurité que vous voulez associer au cluster de bases de données. Pour plus d'informations, consultez Contrôle d'accès par groupe de sécurité.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--vpc-security-group-ids` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `VpcSecurityGroupIds` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètre et description	Méthode	Portée	Remarques sur la durée d'indisponibilité
Fenêtre de retour sur trace cible Intervalle de temps au cours duquel vous souhaitez pouvoir effectuer un retour sur trace de votre cluster de bases de données, en secondes. Ce paramètre est disponible uniquement pour Aurora MySQL et seulement si le cluster de bases de données a été créé avec le retour sur trace activé.	À partir d'AWS Management Console, Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. À partir de l'AWS CLI, exécutez `modify-db-cluster` et définissez l'option `--backtrack-window` . À partir de l'API RDS, appelez `ModifyDBCluster` et définissez le paramètre `BacktrackWindow` .	L'intégralité du cluster de bases de données	Aucune interruption de service n'a lieu pendant cette modification.

Paramètres non applicables aux clusters de bases de données Amazon Aurora

Les paramètres suivants dans la commande `modify-db-cluster` de l'AWS CLI et l'opération de l'API RDS `ModifyDBCluster` ne s'appliquent pas aux clusters de bases de données Amazon Aurora.

> **Note**
>
> Vous ne pouvez pas utiliser l'AWS Management Console pour modifier ces paramètres pour les clusters de bases de données Aurora.

Paramètre AWS CLI	Paramètre de l'API RDS
`--allocated-storage`	`AllocatedStorage`
`--auto-minor-version-upgrade | --no-auto-minor-version-upgrade`	`AutoMinorVersionUpgrade`
`--db-cluster-instance-class`	`DBClusterInstanceClass`
`--enable-performance-insights | --no-enable-performance-insights`	`EnablePerformanceInsights`
`--iops`	`Iops`
`--monitoring-interval`	`MonitoringInterval`
`--monitoring-role-arn`	`MonitoringRoleArn`
`--option-group-name`	`OptionGroupName`
`--performance-insights-kms-key-id`	`PerformanceInsightsKMSKeyId`
`--performance-insights-retention-period`	`PerformanceInsightsRetentionPeriod`

Paramètres non applicables aux instances de base de données Amazon Aurora

Les paramètres suivants dans la commande `modify-db-instance` de l'AWS CLI et l'opération `ModifyDBInstance` de l'API RDS ne s'appliquent pas à Amazon Aurora.

> **Note**
>
> Vous ne pouvez pas utiliser l'AWS Management Console pour modifier ces paramètres pour les instances de base de données Aurora.

Paramètre AWS CLI	Paramètre de l'API RDS
`--allocated-storage`	`AllocatedStorage`
`--allow-major-version-upgrade|--no-allow-major-version-upgrade`	`AllowMajorVersionUpgrade`
`--copy-tags-to-snapshot|--no-copy-tags-to-snapshot`	`CopyTagsToSnapshot`
`--domain`	`Domain`
`--db-security-groups`	`DBSecurityGroups`
`--db-subnet-group-name`	`DBSubnetGroupName`
`--domain-iam-role-name`	`DomainIAMRoleName`
`--multi-az|--no-multi-az`	`MultiAZ`
`--iops`	`Iops`
`--license-model`	`LicenseModel`
`--network-type`	`NetworkType`
`--option-group-name`	`OptionGroupName`
`--processor-features`	`ProcessorFeatures`
`--storage-type`	`StorageType`
`--tde-credential-arn`	`TdeCredentialArn`
`--tde-credential-password`	`TdeCredentialPassword`
`--use-default-processor-features|--no-use-default-processor-features`	`UseDefaultProcessorFeatures`

Ajout de réplicas Aurora à un cluster de bases de données

Un cluster de bases de données Aurora utilisant la réplication se compose d'une instance de base de données principale et de 15 réplicas Aurora, au maximum. L'instance de base de données principale prend en charge les opérations de lecture et d'écriture, et effectue toutes les modifications de données du volume de cluster. Les réplicas Aurora se connectent au même volume de stockage que l'instance de base de données principale, mais prennent uniquement en charge les opérations de lecture. Utilisez des réplicas Aurora pour décharger l'instance de base de données principale des charges de travail en lecture. Pour plus d'informations, consultez Réplicas Aurora.

Les réplicas Amazon Aurora ont les limitations suivantes :

- Vous ne pouvez pas créer de réplica Aurora pour un cluster de bases de données Aurora Serverless v1. Aurora Serverless v1 a une seule instance de base de données dont l'échelle augmente et diminue automatiquement pour prendre en charge toutes les opérations de lecture et d'écriture de base de données.

  Toutefois, vous pouvez ajouter des instances de lecteur aux clusters de bases de données Aurora Serverless v2. Pour plus d'informations, consultez Ajout d'un lecteur Aurora Serverless v2.

Nous vous recommandons de répartir l'instance principale et les réplicas Aurora de votre cluster de bases de données Aurora sur plusieurs zones de disponibilité afin d'améliorer la disponibilité de votre cluster de bases de données. Pour plus d'informations, consultez Disponibilité dans les Régions.

Pour supprimer un réplica Aurora d'un cluster de bases de données Aurora, supprimez le réplica en suivant les instructions de la section Auror Suppression d'une instance de base de données d'un cluster de bases de données Aurora.

> **Note**
>
> Amazon Aurora prend également en charge la réplication avec une base de données externe, telle qu'une instance de base de données RDS. L'instance de base de données RDS doit se trouver dans la même AWS région qu'Amazon Aurora. Pour de plus amples informations, veuillez consulter Réplication avec Amazon Aurora.

Vous pouvez ajouter des répliques Aurora à un cluster de base de données à l'aide de l'AWS Management ConsoleAPIAWS CLI, de ou de l'API RDS.

Console

Pour ajouter un réplica Aurora à un cluster de bases de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Bases de données, puis sélectionnez le cluster de bases de données dans lequel vous voulez ajouter la nouvelle instance de base de données.
3. Assurez-vous que le cluster et l'instance principale ont l'état Disponible. Si le cluster de bases de données ou l'instance principale sont dans un état transitoire tel que En cours de création, vous ne pouvez pas ajouter de réplica.

   Si le cluster ne possède pas d'instance principale, créez-en une à l'aide de la create-db-instanceAWS CLIcommande. Cette situation peut se produire si vous avez utilisé la CLI pour restaurer un instantané de cluster de bases de données, puis afficher le cluster dans le AWS Management Console.
4. Pour Actions, choisissez Add reader (Ajouter un lecteur).

   La page Add reader (Ajouter un lecteur) s'affiche.
5. Sur la page Add reader (Ajouter un lecteur), spécifiez les options de votre réplica Aurora. Le tableau suivant affiche les paramètres pour un réplica Aurora.

Pour cette option	Faire ceci
Zone de disponibilité	Déterminez si vous voulez spécifier une zone de disponibilité particulière. La liste n'inclut que les zones de disponibilité qui sont mappées au groupe de sous-réseaux de base de données que vous avez choisi lors de la création du cluster de bases de données. Pour plus d'informations sur les zones de disponibilité, consultez Régions et zones de disponibilité.
Accessible publiquement	Sélectionnez `Yes` pour attribuer au réplica Aurora une adresse IP publique ; sinon, sélectionnez `No`. Pour plus d'informations sur le masquage des réplicas Aurora de l'accès public, consultez Masquer un cluster de bases de données dans un VPC depuis Internet.

Pour cette option	Faire ceci
Chiffrement	Sélectionnez `Enable encryption` pour activer le chiffrement au repos pour ce réplica Aurora. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.
Classe d'instance de base de données	Sélectionnez une classe d'instance de base de données qui définit les exigences de mémoire et de traitement pour le réplica Aurora. Pour plus d'informations sur les options de classe d'instance de base de données, consultez Classes d'instance de base de données Amazon Aurora.
Source réplica Aurora	Sélectionnez l'identifiant de l'instance principale pour laquelle créer un réplica Aurora.
Identifiant d'instance de base de données	Entrez un nom pour l'instance unique pour votre compte dans la AWS région que vous avez sélectionnée. Vous pouvez choisir d'ajouter une touche d'intelligence au nom, par exemple en incluant la AWS région et le moteur de base de données que vous avez sélectionnés**`aurora-read-instance1`** .
Priorité	Choisissez une priorité de basculement pour l'instance. Si vous ne sélectionnez pas de valeur, la valeur par défaut est tier-1. Cette priorité détermine l'ordre dans lequel les réplicas Aurora sont promus lors de la reprise après une défaillance de l'instance principale. Pour plus d'informations, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.
Port de la base de données	Le port d'un réplica Aurora est le même que le port du cluster de bases de données.

Pour cette option	Faire ceci
Groupe de paramètres de base de données	Sélectionnez un groupe de paramètres. Aurora possède un groupe de paramètres par défaut que vous pouvez utiliser, ou bien vous pouvez créer votre propre groupe de paramètres. Pour plus d'informations sur les groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora.
Performance Insights	La case Turn on Performance Insights (Activer Performance Insights) est cochée par défaut. La valeur n'est pas héritée de l'instance d'enregistreur. Pour plus d'informations, consultez Surveillance de la charge de la base de données avec Performance Insights sur .
Surveillance améliorée	Choisissez Activer la surveillance améliorée pour activer la collecte de métriques en temps réel pour le système d'exploitation sur lequel votre cluster de bases de données s'exécute. Pour plus d'informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.
Monitoring Role (Rôle de surveillance)	Disponible uniquement si l'option Surveillance améliorée est définie sur Activer la surveillance améliorée. Choisissez le rôle IAM que vous avez créé pour permettre à Amazon RDS de communiquer avec Amazon CloudWatch Logs à votre place, ou choisissez Default pour que RDS crée un rôle nommé pour vous. `rds-monitoring-role` Pour de plus amples informations, veuillez consulter Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.
Granularité	Disponible uniquement si l'option Surveillance améliorée est définie sur Activer la surveillance améliorée. Définissez l'intervalle, en secondes, entre les recueils des métriques pour votre cluster DB.

Pour cette option	Faire ceci
Mise à niveau automatique de versions mineures	Sélectionnez Enable auto minor version upgrade (Activer la mise à niveau automatique des versions mineures) si vous souhaitez que votre cluster de bases de données Aurora reçoive automatiquement les mises à niveau des versions mineures dès qu'elles deviennent disponibles. Le paramètre Auto minor version upgrade (Mise à niveau automatique des versions mineures) s'applique aux clusters de bases de données Aurora PostgreSQL et Aurora MySQL. Pour les clusters Aurora MySQL 2.x, ce paramètre met à niveau les clusters vers la version maximale 2.07.2. Pour plus d'informations sur les mises à jour de moteur pour Aurora PostgreSQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora PostgreSQL. Pour plus d'informations sur les mises à jour de moteur pour Aurora MySQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL.

6. Choisissez Add reader (Ajouter un lecteur) pour créer le réplica Aurora.

AWS CLI

Pour créer une réplique Aurora dans votre cluster de base de données, exécutez la create-db-instanceAWS CLIcommande. Incluez le nom du cluster de bases de données comme valeur de l'option `--db-cluster-identifier`. Vous pouvez éventuellement spécifier une zone de disponibilité pour le réplica Aurora à l'aide du paramètre `--availability-zone`, comme dans les exemples suivants.

Par exemple, la commande suivante crée un réplica Aurora compatible avec MySQL 5.7 nommé `sample-instance-us-west-2a`.

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large \
    --availability-zone us-west-2a
```

Pour Windows :

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large ^
    --availability-zone us-west-2a
```

La commande suivante crée un nouveau réplica Aurora compatible avec MySQL 5.7 nommé `sample-instance-us-west-2a`.

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-mysql --db-instance-class
 db.r5.large \
    --availability-zone us-west-2a
```

Pour Windows :

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora --db-instance-class
 db.r5.large ^
    --availability-zone us-west-2a
```

La commande suivante crée un réplica Aurora compatible avec PostgreSQL nommé `sample-instance-us-west-2a`.

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a \
    --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r5.large \
    --availability-zone us-west-2a
```

Pour Windows :

```
aws rds create-db-instance --db-instance-identifier sample-instance-us-west-2a ^
    --db-cluster-identifier sample-cluster --engine aurora-postgresql --db-instance-
class db.r5.large ^
    --availability-zone us-west-2a
```

API RDS

Pour créer une réplique Aurora dans votre cluster de base de données, appelez l'DBInstanceopération Create. Incluez le nom du cluster de bases de données comme valeur du paramètre `DBClusterIdentifier`. Vous pouvez éventuellement spécifier une zone de disponibilité pour le réplica Aurora à l'aide du paramètre `AvailabilityZone`.

Pour plus d'informations sur l'utilisation d'Amazon Aurora Auto Scaling avec des réplicas Aurora, consultez les sections suivantes.

Rubriques

- Amazon Aurora Auto Scaling avec des réplicas Aurora
- Ajout d'une stratégie d'autoscaling à un cluster de bases de données Amazon Aurora
- Modification d'une stratégie d'autoscaling pour un cluster de bases de données Amazon Aurora
- Suppression d'une stratégie d'autoscaling d'un cluster de bases de données Amazon Aurora

Amazon Aurora Auto Scaling avec des réplicas Aurora

Pour répondre à vos exigences en matière de connectivité et de charge de travail, Aurora Auto Scaling ajuste dynamiquement le nombre de réplicas Aurora (instances de base de données en lecture) alloués pour un cluster de bases de données Aurora. Aurora Auto Scaling est disponible pour Aurora MySQL et Aurora PostgreSQL. Aurora Auto Scaling permet à votre cluster de bases de données Aurora de gérer les augmentations soudaines de connectivité ou de charge de travail. Lorsque la connectivité ou la charge de travail diminue, Aurora Auto Scaling supprime les réplicas Aurora superflus, si bien que vous ne payez pas pour les instances de base de données allouées qui ne sont pas utilisées.

Vous définissez et appliquez une stratégie de mise à l'échelle à un cluster de bases de données Aurora. La stratégie de mise à l'échelle définit le nombre minimal et maximal de réplicas Aurora qu'Aurora Auto Scaling peut gérer. Sur la base de cette politique, Aurora Auto Scaling ajuste le nombre de répliques Aurora à la hausse ou à la baisse en fonction des charges de travail réelles, déterminées à l'aide des CloudWatch métriques et des valeurs cibles d'Amazon.

> **Note**
>
> Aurora Auto Scaling ne s'applique pas à la charge de travail de l'instance de base de données d'enregistreur. Aurora Auto Scaling ne permet de gérer la charge de travail que sur les instances de lecteur.

Vous pouvez utiliser le AWS Management Console pour appliquer une politique de dimensionnement basée sur une métrique prédéfinie. Vous pouvez également utiliser l'API Aurora Auto Scaling AWS CLI ou l'API Aurora pour appliquer une politique de dimensionnement basée sur une métrique prédéfinie ou personnalisée.

Rubriques

Avant de commencer

Avant d'utiliser Aurora Auto Scaling avec un cluster de bases de données Aurora, vous devez d'abord créer un cluster de bases de données Aurora avec une instance de base de données principale (écriture). Pour plus d'informations sur la création d'un cluster de bases de données Aurora, consultez Création d'un cluster de bases de données Amazon Aurora.

Aurora Auto Scaling ne met à l'échelle un cluster de bases de données que si celui-ci est à l'état disponible.

Quand Aurora Auto Scaling ajoute un nouveau réplica Aurora, celui-ci appartient à la même classe d'instance de base de données que celle utilisée par l'instance principale. Pour plus d'informations sur les classes d'instance de base de données, consultez Classes d'instance de base de données Amazon Aurora. De même, le niveau de promotion pour les nouveaux réplicas Aurora est défini sur la dernière priorité, 15 par défaut. Cela signifie que pendant un basculement, un réplica ayant une meilleure priorité, par exemple un réplica ayant été créé manuellement, serait promu en premier. Pour plus d'informations, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.

Aurora Auto Scaling supprime uniquement les réplicas Aurora qu'il a créés.

Pour bénéficier d'Aurora Auto Scaling, vos applications doivent prendre en charge les connexions aux nouveaux réplicas Aurora. Pour cela, nous vous recommandons d'utiliser le point de terminaison de lecteur Aurora. Vous pouvez utiliser un pilote tel que le pilote AWS JDBC. Pour de plus amples informations, veuillez consulter Connexion à un cluster de bases de données Amazon Aurora.

> **Note**
>
> Les bases de données globales Aurora ne prennent actuellement pas en charge Aurora Auto Scaling pour les clusters de bases de données secondaire.

Stratégies Auto Scaling Aurora

Aurora Auto Scaling utilise une stratégie de mise à l'échelle pour ajuster le nombre de réplicas Aurora dans un cluster de bases de données Aurora. Aurora Auto Scaling comprend les éléments suivants :

- Un rôle lié à un service
- Une métrique cible

- Une capacité maximale et minimale
- Un temps de stabilisation

Rubriques

Rôle lié à un service

Aurora Auto Scaling utilise le rôle lié à un service `AWSServiceRoleForApplicationAutoScaling_RDSCluster`. Pour plus d'informations, consultez Rôles liés aux services pour Application Auto Scaling dans le Guide de l'utilisateur Application Auto Scaling.

Métrique cible

Dans ce type de stratégie, une métrique prédéfinie ou personnalisée et une valeur cible pour la métrique sont spécifiées dans une configuration de stratégie de dimensionnement Suivi de la cible. Aurora Auto Scaling crée et gère les CloudWatch alarmes qui déclenchent la politique de dimensionnement et calcule l'ajustement de dimensionnement en fonction de la métrique et de la valeur cible. La stratégie de dimensionnement ajoute ou supprime des réplicas Aurora si nécessaire pour maintenir la métrique à la valeur cible spécifiée ou proche de celle-ci. En plus de maintenir la métrique proche de la valeur cible, une stratégie de dimensionnement Suivi de la cible s'ajuste également aux fluctuations de la métrique dues à l'évolution de la charge de travail. Une stratégie de ce type minimise également les fluctuations rapides dans le nombre de réplicas Aurora disponibles pour votre cluster de bases de données.

Par exemple, examinons une stratégie de dimensionnement qui utilise la métrique d'utilisation moyenne de l'UC prédéfinie. Ce type de stratégie peut maintenir l'utilisation de l'UC au pourcentage d'utilisation indiqué, tel que 40 %, ou proche de celui-ci.

> **Note**
>
> Pour chaque cluster de bases de données Aurora, vous ne pouvez créer qu'une seule stratégie Auto Scaling pour chaque métrique cible.

Lorsque vous configurez Aurora Auto Scaling, la valeur de métrique cible est calculée comme la moyenne de toutes les instances de lecteur du cluster. Ce calcul est effectué comme suit :

- Inclut toutes les instances de lecteur du cluster Aurora, qu'elles soient gérées par Auto Scaling ou ajoutées manuellement.
- Inclut les instances associées à des points de terminaison personnalisés. Les points de terminaison personnalisés n'influencent pas le calcul des métriques cibles.
- N'inclut pas l'instance d'enregistreur du cluster.

Les mesures sont dérivées à CloudWatch l'aide des dimensions suivantes :

- `DBClusterIdentifier`
- `Role=READER`

Prenons l'exemple d'un cluster Aurora MySQL avec la configuration suivante :

- Instances manuelles (non contrôlées par Auto Scaling) :
  - Enregistreur avec 50 % d'utilisation du processeur
  - Lecteur 1 (point de terminaison personnalisé : `custom-reader-1`) avec 90 % d'utilisation du processeur
  - Lecteur 2 (point de terminaison personnalisé : `custom-reader-2`) avec 90 % d'utilisation du processeur
- Instance Auto Scaling :
  - Lecteur 3 (ajouté à l'aide d'Auto Scaling) avec 10 % d'utilisation du processeur

Dans ce scénario, la métrique cible de la stratégie Auto Scaling est calculée comme suit :

```
Target metric = (CPU utilization of reader 1 + reader 2 + reader 3) / total number of
 readers
```

```
Target metric = (90 + 90 + 10) / 3 = 63.33%
```

La stratégie Auto Scaling utilise cette valeur pour évaluer s'il convient de procéder à une réduction ou à une augmentation horizontale en fonction du seuil défini.

Éléments à prendre en compte :

- Bien que les points de terminaison personnalisés déterminent la manière dont le trafic est acheminé vers des lecteurs spécifiques, ils n'excluent pas les lecteurs du calcul des métriques.
- Les instances manuelles sont toujours comprises dans le calcul des métriques cibles.
- Pour éviter tout comportement de mise à l'échelle inattendu, assurez-vous que la configuration Auto Scaling prend en compte toutes les instances de lecteur du cluster.
- Si un cluster ne possède aucun lecteur, la métrique ne sera pas calculée et les alarmes de la stratégie Auto Scaling resteront inactives. Pour que la stratégie Auto Scaling fonctionne efficacement, au moins un lecteur doit être présent à tout moment.

Capacité minimale et maximale

Vous pouvez spécifier le nombre maximal de réplicas Aurora que doit gérer Application Auto Scaling. Cette valeur doit être comprise entre 0 et 15 et doit être supérieure ou égale à la valeur spécifiée pour le nombre minimal de réplicas Aurora.

Vous pouvez également spécifier le nombre minimal de réplicas Aurora que doit gérer Application Auto Scaling. Cette valeur doit être comprise entre 0 et 15 et doit être inférieure ou égale à la valeur spécifiée pour le nombre maximal de réplicas Aurora.

Il doit y avoir au moins une instance de base de données de lecteur pour qu'Aurora Auto Scaling fonctionne. Si le cluster de bases de données ne possède aucune instance de lecteur et que vous définissez la capacité minimale sur 0, Aurora Auto Scaling ne fonctionnera pas.

> **Note**
>
> La capacité minimale et maximale est définie pour un cluster de bases de données Aurora. Les valeurs spécifiées s'appliquent à toutes les stratégies associées à ce cluster de bases de données Aurora.

Temps de stabilisation

Vous pouvez optimiser la réactivité d'une stratégie de dimensionnement Suivi de la cible en ajoutant des temps de stabilisation qui affectent le dimensionnement de votre cluster de bases de données Aurora via l'ajout ou la suppression d'extensions matérielles. Un temps de stabilisation bloque les demandes de montée ou de diminution en charge ultérieures jusqu'à l'expiration de la période. Ces blocs ralentissent les suppressions de réplicas Aurora dans votre cluster de bases de données Aurora pour les demandes de diminution en charge, et la création de réplicas Aurora pour les demandes de montée en charge.

Vous pouvez spécifier les temps de stabilisation suivants :

- Une activité de diminution en charge réduit le nombre de réplicas Aurora dans votre cluster de bases de données Aurora. Un temps de stabilisation de diminution en charge spécifie la durée, en secondes, devant s'écouler entre la fin d'une activité de diminution et le début d'une autre.
- Une activité de montée en charge augmente le nombre de réplicas Aurora dans votre cluster de bases de données Aurora. Un temps de stabilisation de montée en charge spécifie la durée, en secondes, devant s'écouler entre la fin d'une activité de montée en charge et le début d'une autre.

> **Note**
>
> Un temps de stabilisation pour la montée en puissance est ignoré si une demande de montée en puissance suivante concerne un plus grand nombre de répliques Aurora que la première demande.

Si vous ne spécifiez pas de temps de stabilisation de mise à l'échelle horizontale ou de montée en puissance, la valeur par défaut est de 300 secondes pour chaque.

Activation ou désactivation d'activités de diminution en charge

Vous pouvez activer ou désactiver des activités de diminution en charge pour une stratégie. L'activation d'activités de diminution en charge permet à la stratégie de dimensionnement de supprimer des réplicas Aurora. Lorsque des activités de diminution en charge sont activées, le temps de stabilisation de diminution en charge figurant dans la stratégie de dimensionnement leur est appliqué. La désactivation d'activités de diminution en charge empêche la stratégie de dimensionnement de supprimer des réplicas Aurora.

> **Note**
>
> Les activités de montée en charge sont toujours activées de sorte que la stratégie de dimensionnement puisse créer des réplicas Aurora si nécessaire.

Ajout, modification ou suppression de stratégies d’autoscaling

Vous pouvez ajouter, modifier ou supprimer des politiques de dimensionnement automatique à l'AWS Management Consoleaide de l'API Application Auto Scaling. AWS CLI Pour plus d’informations sur l’ajout, la modification ou la suppression de stratégies d’autoscaling, consultez les sections suivantes.

- Ajout d’une stratégie d’autoscaling à un cluster de bases de données Amazon Aurora
- Modification d’une stratégie d’autoscaling pour un cluster de bases de données Amazon Aurora
- Suppression d’une stratégie d’autoscaling d’un cluster de bases de données Amazon Aurora

Instance de base de données IDs et balisage

Lorsqu’un réplica est ajouté par Aurora Auto Scaling, son ID d’instance de base de données est doté du préfixe `application-autoscaling-` (par exemple, `application-autoscaling-61aabbcc-4e2f-4c65-b620-ab7421abc123`).

La balise suivante est automatiquement ajoutée à l’instance de base de données. Vous pouvez l’afficher sous l’onglet Tags (Balises) de la page détaillée de l’instance de base de données.

Tag	Valeur
application-autoscaling:resourceId	cluster:mynewcluster-cluster

Pour en savoir plus sur les balises de ressource Amazon RDS, consultez Marquage des ressources Amazon Aurora et Amazon RDS.

Aurora Auto Scaling et Performance Insights

Vous pouvez utiliser Performance Insights pour surveiller les réplicas ajoutés par Aurora Auto Scaling, comme pour n’importe quelle instance de base de données de lecteur Aurora.

Pour plus d'informations sur l'utilisation de Performance Insights pour surveiller des clusters de bases de données Aurora, consultez Surveillance de la charge de la base de données avec Performance Insights sur .

Ajout d'une stratégie d'autoscaling à un cluster de bases de données Amazon Aurora

Vous pouvez ajouter une stratégie de mise à l'échelle à l'aide de la AWS Management Console, de AWS CLI ou de l'API Application Auto Scaling.

> **Note**
>
> Pour un exemple d'ajout d'une stratégie de mise à l'échelle à l'aide d'CloudFormation, consultez Déclaration d'une stratégie de mise à l'échelle pour un cluster de bases de données Aurora dans le Guide de l'utilisateur AWS CloudFormation.

Console

Vous pouvez ajouter une stratégie de mise à l'échelle à un cluster de bases de données Aurora à partir de l'AWS Management Console.

Pour ajouter une stratégie Auto Scaling à un cluster de bases de données Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données.
3. Choisissez le cluster de bases de données Aurora auquel vous souhaitez ajouter une stratégie.
4. Choisissez l'onglet Logs & events (Journaux et événements).
5. Dans la section Auto scaling policies (Stratégies Auto Scaling), choisissez Ajouter.

   La boîte de dialogue Ajouter une Stratégie Auto Scaling s'affiche.
6. Dans le champ Policy Name (Nom de stratégie), saisissez le nom de la stratégie.
7. Pour la métrique cible, choisissez l'une des actions suivantes :

   - Average CPU utilization of Aurora Replicas (Utilisation moyenne d'UC des réplicas Aurora) pour créer une stratégie basée sur l'utilisation moyenne de l'UC.

- Average connections of Aurora Replicas (Nombre moyen de connexions de réplicas Aurora) pour créer une stratégie basée sur le nombre moyen de connexions aux réplicas Aurora.

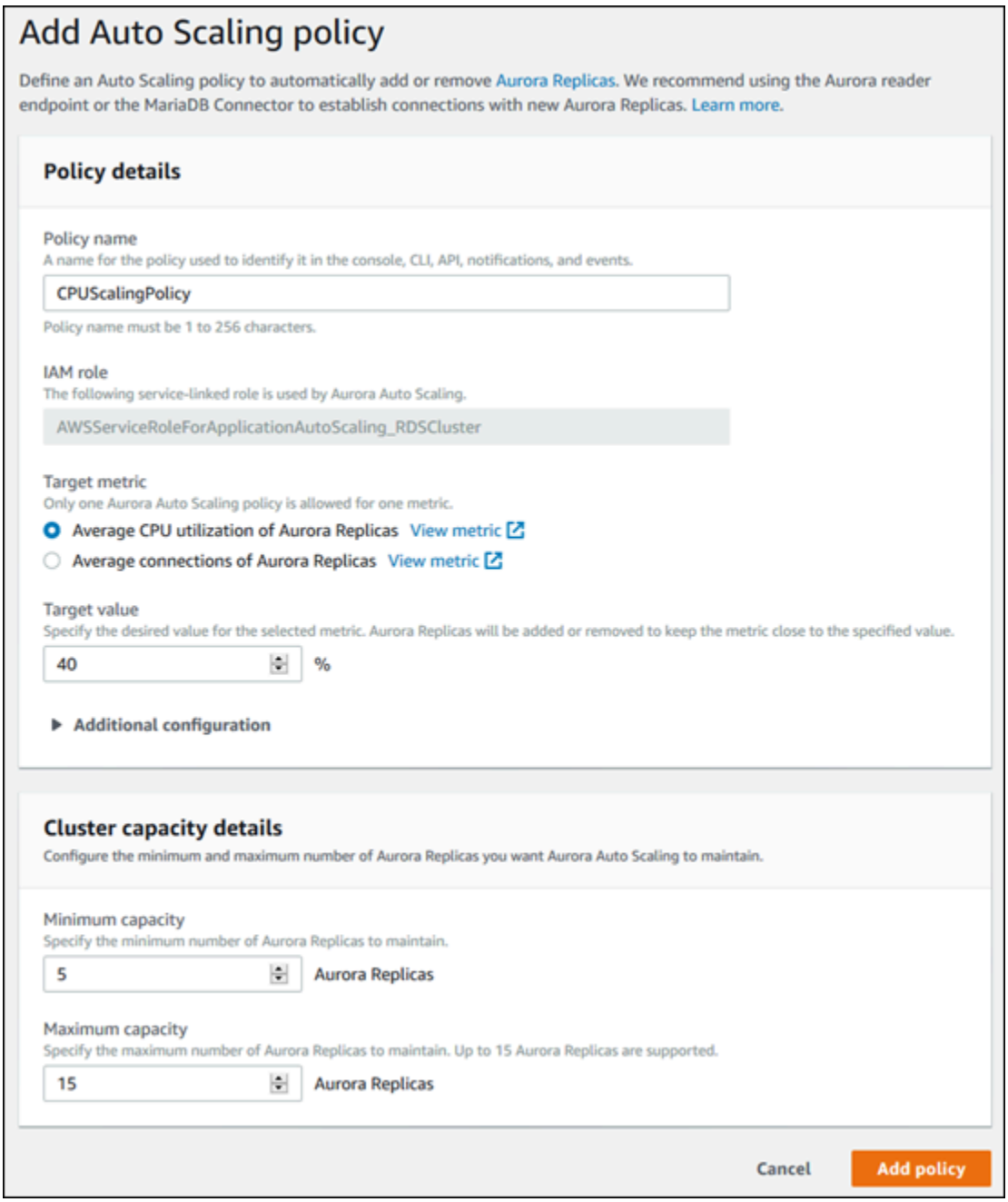
8. Pour la valeur cible, saisissez l'un des éléments suivants :

   - Si vous avez choisi Average CPU utilization of Aurora Replicas (Utilisation moyenne d'UC des réplicas Aurora) à l'étape précédente, saisissez le pourcentage d'utilisation de l'UC à maintenir sur les réplicas Aurora.
   - Si vous avez choisi Average connections of Aurora Replicas (Nombre moyen de connexions de réplicas Aurora) à l'étape précédente, saisissez le nombre de connexions à maintenir.

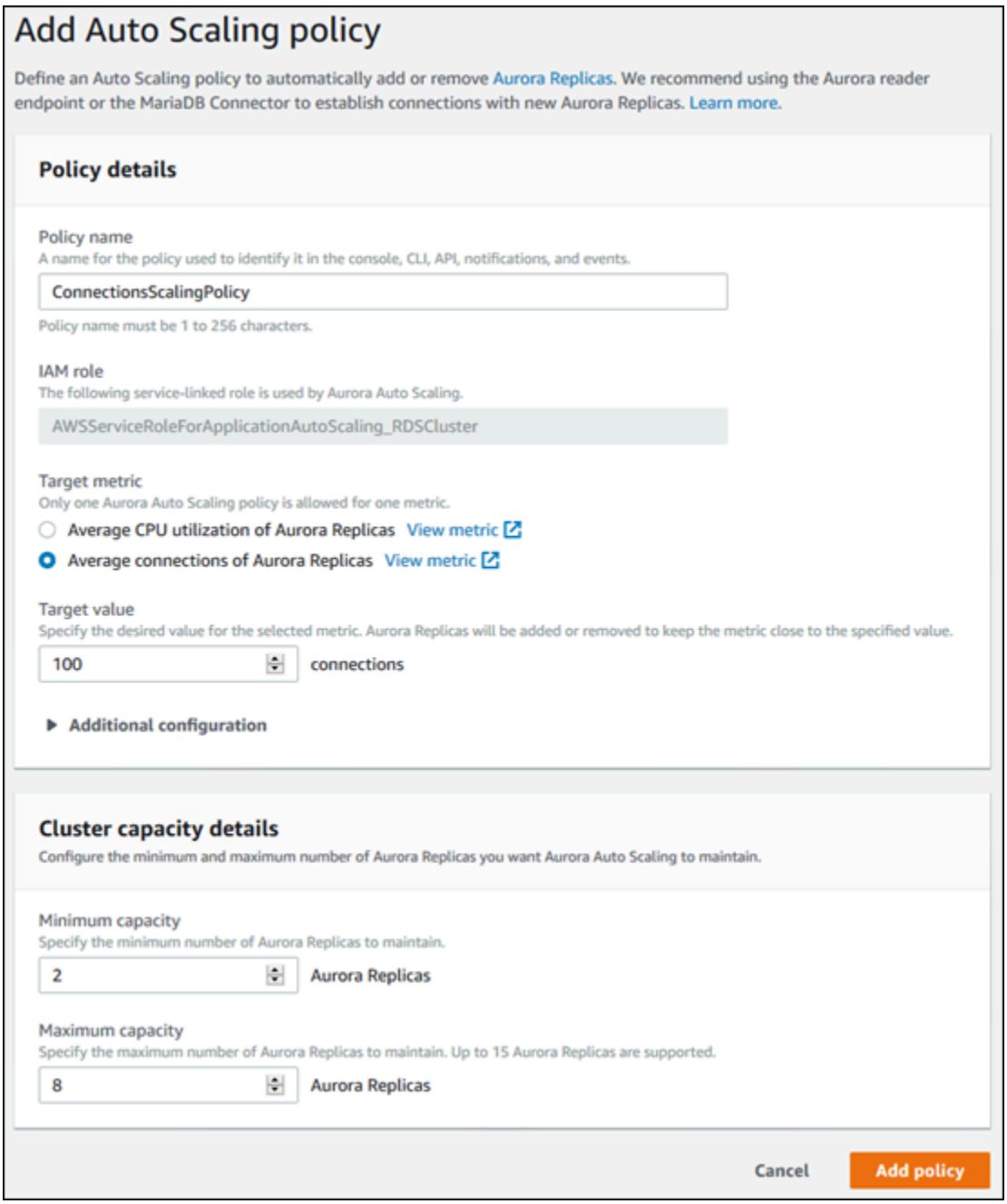
   Des réplicas Aurora sont ajoutés ou supprimés pour maintenir la métrique proche de la valeur spécifiée.

9. (Facultatif) Ouvrez Additional Configuration (Configuration supplémentaire) pour créer un temps de stabilisation de mise à l'échelle horizontale ou de montée en puissance.
10. Pour Minimum capacity (Capacité minimale), saisissez le nombre minimal de réplicas Aurora que la stratégie Aurora Auto Scaling doit maintenir.
11. Pour Maximum capacity (Capacité maximale), saisissez le nombre maximal de réplicas Aurora que la stratégie Aurora Auto Scaling doit maintenir.
12. Choisissez Add policy (Ajouter la stratégie).

La boîte de dialogue suivante crée une stratégie Auto Scaling basée sur une utilisation moyenne de l'UC de 40 %. La stratégie indique un minimum de cinq réplicas Aurora et un maximum de 15.



La boîte de dialogue suivante crée une stratégie Auto Scaling basée sur un nombre moyen de connexions égal à 100. La stratégie indique un minimum de deux réplicas Aurora et un maximum de huit.



AWS CLI ou API Application Auto Scaling

Vous pouvez appliquer une stratégie de mise à l'échelle basée sur une métrique prédéfinie ou personnalisée. Pour ce faire, vous pouvez utiliser l'AWS CLI ou l'API Application Auto Scaling. La première étape consiste à enregistrer votre cluster de bases de données Aurora dans Application Auto Scaling.

Enregistrement d'un cluster de bases de données Aurora

Avant d'utiliser Aurora Auto Scaling avec un cluster de bases de données Aurora, enregistrez votre cluster de bases de données Aurora dans Application Auto Scaling. Cette action permet de définir la dimension et les limites de la mise à l'échelle à appliquer à ce cluster. Application Auto Scaling met à l'échelle de façon dynamique le cluster de bases de données Aurora le long de la dimension évolutive `rds:cluster:ReadReplicaCount`, qui représente le nombre de réplicas Aurora.

Pour enregistrer votre cluster de bases de données Aurora, vous pouvez utiliser l'AWS CLI ou l'API Application Auto Scaling.

AWS CLI

Pour enregistrer votre cluster de bases de données Aurora, utilisez la commande de l'AWS CLI `register-scalable-target` avec les paramètres suivants :

- `--service-namespace` – Définissez cette valeur sur `rds`.
- `--resource-id` – Identifiant de la ressource du cluster de bases de données Aurora. Pour ce paramètre, le type de ressource est `cluster` et l'identifiant unique est le nom du cluster de bases de données Aurora, par exemple `cluster:myscalablecluster`.
- `--scalable-dimension` – Définissez cette valeur sur `rds:cluster:ReadReplicaCount`.
- `--min-capacity` – Nombre minimal d'instances de base de données en écriture devant être gérées par Application Auto Scaling. Pour plus d'informations sur la relation entre `--min-capacity`, `--max-capacity` et le nombre d'instances de base de données dans votre cluster, consultez Capacité minimale et maximale.
- `--max-capacity` – Nombre maximal d'instances de base de données en écriture devant être gérées par Application Auto Scaling. Pour plus d'informations sur la relation entre `--min-capacity`, `--max-capacity` et le nombre d'instances de base de données dans votre cluster, consultez Capacité minimale et maximale.

Example

Dans l'exemple suivant, vous enregistrez un cluster de bases de données Aurora nommé `myscalablecluster`. L'enregistrement indique que le cluster de bases de données doit être dimensionné de façon dynamique pour contenir de un à huit réplicas Aurora.

Pour Linux, macOS ou Unix :

```
aws application-autoscaling register-scalable-target \
    --service-namespace rds \
    --resource-id cluster:myscalablecluster \
    --scalable-dimension rds:cluster:ReadReplicaCount \
    --min-capacity 1 \
    --max-capacity 8 \
```

Pour Windows :

```
aws application-autoscaling register-scalable-target ^
    --service-namespace rds ^
    --resource-id cluster:myscalablecluster ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --min-capacity 1 ^
    --max-capacity 8 ^
```

API Application Auto Scaling

Pour enregistrer votre cluster de bases de données Aurora dans Application Auto Scaling, utilisez l'opération d'API Application Auto Scaling `RegisterScalableTarget` avec les paramètres suivants :

- `ServiceNamespace` – Définissez cette valeur sur `rds`.
- `ResourceID` – Identifiant de la ressource du cluster de bases de données Aurora. Pour ce paramètre, le type de ressource est `cluster` et l'identifiant unique est le nom du cluster de bases de données Aurora, par exemple `cluster:myscalablecluster`.
- `ScalableDimension` – Définissez cette valeur sur `rds:cluster:ReadReplicaCount`.
- `MinCapacity` – Nombre minimal d'instances de base de données en écriture devant être gérées par Application Auto Scaling. Pour plus d'informations sur la relation entre `MinCapacity`, `MaxCapacity` et le nombre d'instances de base de données dans votre cluster, consultez Capacité minimale et maximale.
- `MaxCapacity` – Nombre maximal d'instances de base de données en écriture devant être gérées par Application Auto Scaling. Pour plus d'informations sur la relation entre `MinCapacity`, `MaxCapacity` et le nombre d'instances de base de données dans votre cluster, consultez Capacité minimale et maximale.

Example

Dans l'exemple suivant, vous enregistrez un cluster de bases de données Aurora nommé `myscalablecluster` avec l'API Application Auto Scaling. Cet enregistrement indique que le cluster de bases de données doit être dimensionné de façon dynamique pour contenir de un à huit réplicas Aurora.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "MinCapacity": 1,
    "MaxCapacity": 8
}
```

Définition d'une stratégie de dimensionnement pour un cluster de bases de données Aurora

Une configuration de stratégie de mise à l'échelle Suivi de la cible est représentée par un bloc JSON dans lequel sont définies les métriques et valeurs cibles. Vous pouvez enregistrer une configuration de stratégie de mise à l'échelle sous forme de bloc JSON dans un fichier texte. Vous utilisez ce fichier texte lors de l'appel de l'AWS CLI ou de l'API Application Auto Scaling. Pour plus d'informations sur la syntaxe de la configuration d'une stratégie, consultez `TargetTrackingScalingPolicyConfiguration` dans le manuel Référence d'API Application Auto Scaling.

Les options suivantes sont disponibles pour définir une configuration de stratégie de dimensionnement Suivi de la cible.

Rubriques

- Utilisation d'une métrique prédéfinie
- Utilisation d'une métrique personnalisée

- Utilisation des temps de stabilisation
- Désactivation de l'activité de diminution en charge

Utilisation d'une métrique prédéfinie

L'utilisation de métriques prédéfinies vous permet de définir rapidement une stratégie de mise à l'échelle de suivi de la cible pour un cluster de bases de données Aurora qui fonctionne aussi bien avec la mise à l'échelle de suivi de la cible qu'avec la mise à l'échelle dynamique dans Aurora Auto Scaling.

Pour l'heure, les métriques prédéfinies d'Aurora Auto Scaling prises en charge par Aurora sont les suivantes :

- RDSReaderAverageCPUUtilization – Valeur moyenne de la métrique `CPUUtilization` dans CloudWatch sur tous les réplicas Aurora du cluster de bases de données Aurora.
- RDSReaderAverageDatabaseConnections – Valeur moyenne de la métrique `DatabaseConnections` dans CloudWatch sur tous les réplicas Aurora du cluster de bases de données Aurora.

Pour plus d'informations sur les métriques `CPUUtilization` et `DatabaseConnections`, consultez CloudWatch Métriques Amazon pour Amazon Aurora.

Pour utiliser une métrique prédéfinie dans votre stratégie de mise à l'échelle, créez une configuration de suivi de la cible pour votre politique de dimensionnement. Cette configuration doit inclure `PredefinedMetricSpecification` pour la métrique prédéfinie et `TargetValue` pour la valeur cible de cette métrique.

Example

L'exemple suivant décrit une configuration de stratégie classique pour le dimensionnement Suivi de la cible d'un cluster de bases de données Aurora. Dans cette configuration, la métrique prédéfinie `RDSReaderAverageCPUUtilization` est utilisée pour ajuster le cluster de bases de données Aurora en fonction d'une utilisation moyenne de l'UC de 40 % sur tous les réplicas Aurora.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
```

```
    }
}
```

Utilisation d’une métrique personnalisée

L’utilisation de métriques personnalisées vous permet de définir une stratégie de dimensionnement Suivi de la cible répondant à vos exigences personnelles. Vous pouvez définir une métrique personnalisée en fonction d’une métrique Aurora qui évolue proportionnellement à la mise à l’échelle.

Toutes les métriques Aurora ne fonctionnent pas pour le suivi de la cible. La métrique doit être une métrique d’utilisation valide et décrire le degré d’occupation d’une instance. La valeur de la métrique doit augmenter ou diminuer proportionnellement au nombre de réplicas Aurora dans le cluster de bases de données Aurora. Cette augmentation ou diminution proportionnelle est nécessaire pour que les données de la métrique puissent être utilisées afin d’augmenter ou de réduire proportionnellement le nombre de réplicas Aurora.

Example

L’exemple suivant décrit une configuration de suivi de la cible pour une stratégie de dimensionnement. Dans cette configuration, une métrique personnalisée ajuste un cluster de bases de données Aurora en fonction d’une utilisation moyenne de l’UC de 50 % sur tous les réplicas Aurora d’un cluster de bases de données Aurora nommé `my-db-cluster`.

```
{
    "TargetValue": 50,
    "CustomizedMetricSpecification":
    {
        "MetricName": "CPUUtilization",
        "Namespace": "AWS/RDS",
        "Dimensions": [
            {"Name": "DBClusterIdentifier","Value": "my-db-cluster"},
            {"Name": "Role","Value": "READER"}
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

Utilisation des temps de stabilisation

Vous pouvez spécifier une valeur, en secondes, pour que `ScaleOutCooldown` ajoute un temps de stabilisation pour la montée en charge de votre cluster de bases de données Aurora. De la

même manière, vous pouvez ajouter une valeur, en secondes, pour que `ScaleInCooldown` ajoute un temps de stabilisation pour la diminution en charge de votre cluster de bases de données Aurora. Pour plus d'informations sur `ScaleInCooldown` et `ScaleOutCooldown`, consultez `TargetTrackingScalingPolicyConfiguration` dans le manuel Référence d'API Application Auto Scaling.

Example

L'exemple suivant décrit une configuration de suivi de la cible pour une stratégie de dimensionnement. Dans cette configuration, la métrique prédéfinie `RDSReaderAverageCPUUtilization` est utilisée pour ajuster un cluster de bases de données Aurora en fonction d'une utilisation moyenne de l'UC de 40 % sur tous les réplicas Aurora de ce cluster de bases de données Aurora. La configuration indique un temps de stabilisation de diminution en charge de 10 minutes et un temps de stabilisation de montée en charge de 5 minutes.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
}
```

Désactivation de l'activité de diminution en charge

Vous pouvez empêcher la configuration de stratégie de dimensionnement Suivi de la cible de diminuer la taille de votre cluster de bases de données Aurora en désactivant l'activité de diminution en charge. La désactivation de l'activité de diminution en charge empêche la stratégie de dimensionnement de supprimer des réplicas Aurora, tout en autorisant encore la stratégie de dimensionnement à les créer si nécessaire.

Vous pouvez spécifier une valeur booléenne pour que `DisableScaleIn` active ou désactive l'activité de diminution en charge de votre cluster de bases de données Aurora. Pour plus d'informations sur `DisableScaleIn`, consultez `TargetTrackingScalingPolicyConfiguration` dans le manuel Référence d'API Application Auto Scaling.

Example

L'exemple suivant décrit une configuration de suivi de la cible pour une stratégie de dimensionnement. Dans cette configuration, la métrique prédéfinie `RDSReaderAverageCPUUtilization` ajuste un cluster de bases de données Aurora en fonction d'une utilisation moyenne de l'UC de 40 % sur tous les réplicas Aurora de ce cluster de bases de données Aurora. La configuration désactive l'activité de diminution en charge pour la stratégie de dimensionnement.

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
    },
    "DisableScaleIn": true
}
```

Application d'une stratégie de dimensionnement à un cluster de bases de données Aurora

Après avoir enregistré votre cluster de bases de données Aurora dans Application Auto Scaling et défini une stratégie de mise à l'échelle, appliquez cette dernière au cluster de bases de données Aurora enregistré. Pour appliquer une stratégie de mise à l'échelle à un cluster de bases de données Aurora, vous pouvez utiliser l'AWS CLI ou l'API Application Auto Scaling.

AWS CLI

Pour appliquer une stratégie de mise à l'échelle à votre cluster de bases de données Aurora, utilisez la commande de l'AWS CLI `put-scaling-policy` avec les paramètres suivants :

- `--policy-name` – Nom de la stratégie de mise à l'échelle.
- `--policy-type` – Définissez cette valeur sur `TargetTrackingScaling`.
- `--resource-id` – Identifiant de la ressource du cluster de bases de données Aurora. Pour ce paramètre, le type de ressource est `cluster` et l'identifiant unique est le nom du cluster de bases de données Aurora, par exemple `cluster:myscalablecluster`.
- `--service-namespace` – Définissez cette valeur sur `rds`.
- `--scalable-dimension` – Définissez cette valeur sur `rds:cluster:ReadReplicaCount`.
- `--target-tracking-scaling-policy-configuration` – Configuration de stratégie de mise à l'échelle de suivi de la cible à utiliser pour le cluster de bases de données Aurora.

Example

Dans l'exemple suivant, vous appliquez une stratégie de mise à l'échelle de suivi de la cible nommée `myscalablepolicy` à un cluster de bases de données Aurora nommé `myscalablecluster` à l'aide d'Application Auto Scaling. Pour ce faire, vous utilisez une configuration de stratégie enregistrée dans un fichier nommé `config.json`.

Pour Linux, macOS ou Unix :

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
    --target-tracking-scaling-policy-configuration file://config.json
```

Pour Windows :

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
    --target-tracking-scaling-policy-configuration file://config.json
```

API Application Auto Scaling

Pour appliquer une stratégie de mise à l'échelle à votre cluster de bases de données Aurora à l'aide de l'API Application Auto Scaling, utilisez l'opération d'API Application Auto Scaling `PutScalingPolicy` avec les paramètres suivants :

- `PolicyName` – Nom de la stratégie de mise à l'échelle.
- `ServiceNamespace` – Définissez cette valeur sur `rds`.
- `ResourceID` – Identifiant de la ressource du cluster de bases de données Aurora. Pour ce paramètre, le type de ressource est `cluster` et l'identifiant unique est le nom du cluster de bases de données Aurora, par exemple `cluster:myscalablecluster`.

- `ScalableDimension` – Définissez cette valeur sur `rds:cluster:ReadReplicaCount`.
- `PolicyType` – Définissez cette valeur sur `TargetTrackingScaling`.
- `TargetTrackingScalingPolicyConfiguration` – Configuration de stratégie de mise à l'échelle de suivi de la cible à utiliser pour le cluster de bases de données Aurora.

Example

Dans l'exemple suivant, vous appliquez une stratégie de mise à l'échelle de suivi de la cible nommée `myscalablepolicy` à un cluster de bases de données Aurora nommé `myscalablecluster` à l'aide d'Application Auto Scaling. Vous utilisez une configuration de stratégie basée sur la métrique prédéfinie `RDSReaderAverageCPUUtilization`.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "RDSReaderAverageCPUUtilization"
        }
    }
}
```

Modification d’une stratégie d’autoscaling pour un cluster de bases de données Amazon Aurora

Vous pouvez modifier une stratégie de mise à l’échelle à l’aide de la AWS Management Console, de AWS CLI ou de l’API Application Auto Scaling.

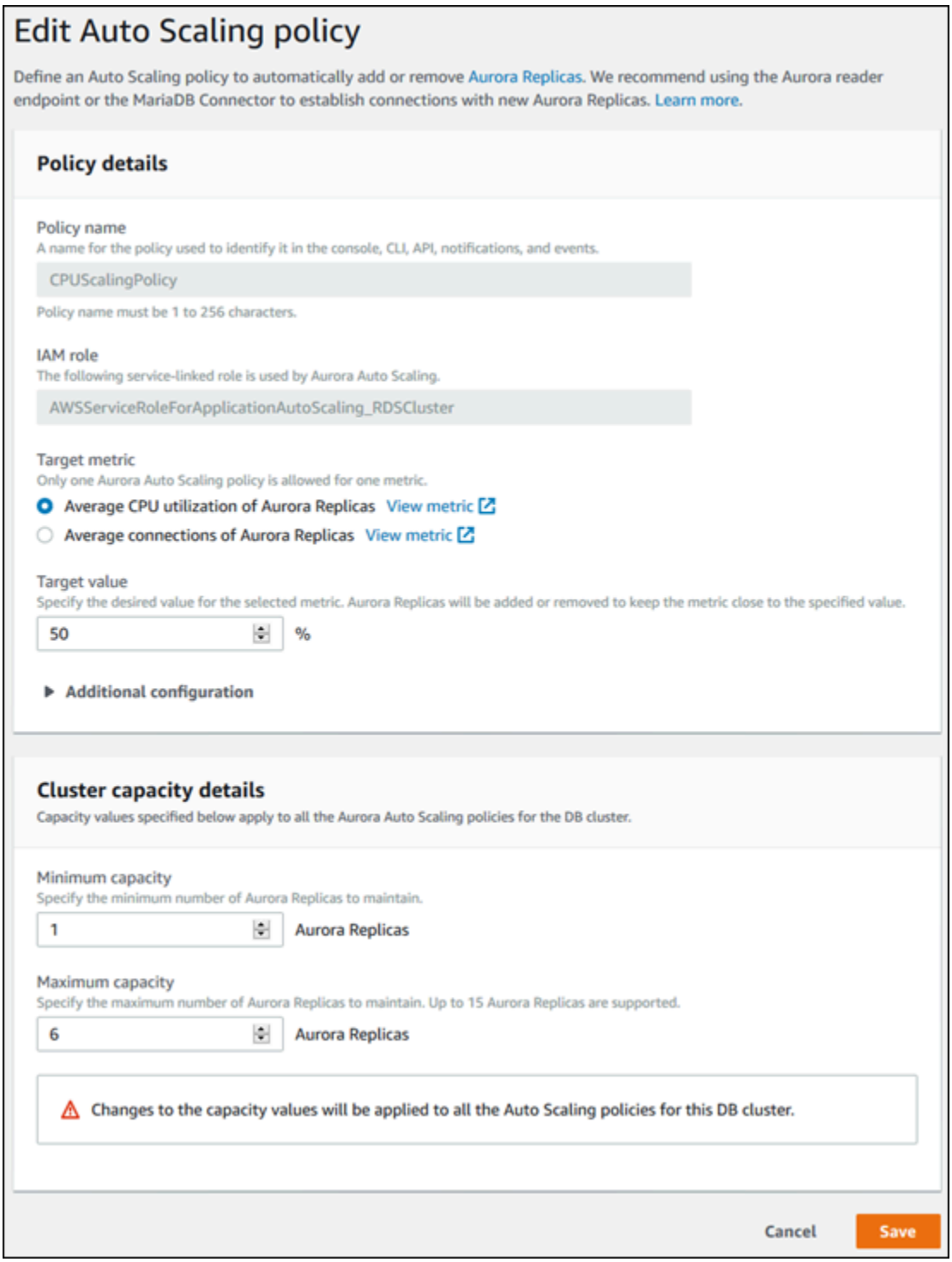
Console

Vous pouvez modifier une stratégie de mise à l’échelle à l’aide de la AWS Management Console.

Pour modifier une stratégie d’autoscaling pour un cluster de bases de données Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster de bases de données Aurora dont vous voulez modifier la stratégie d’autoscaling.
4. Choisissez l’onglet Logs & events (Journaux et événements).
5. Dans la section Stratégies Auto Scaling, choisissez la stratégie Auto Scaling, puis sélectionnez Modifier.
6. Apportez des modifications à la stratégie.
7. Choisissez Enregistrer.

Vous trouverez ci-dessous un exemple de boîte de dialogue Edit Auto Scaling policy (Modifier la stratégie Auto Scaling).



AWS CLI ou API Application Auto Scaling

Vous pouvez utiliser l'AWS CLI ou l'API Application Auto Scaling pour modifier une stratégie de mise à l'échelle de la même manière que vous appliquez une stratégie de mise à l'échelle :

- Lorsque vous utilisez l'AWS CLI, spécifiez le nom de la stratégie à modifier dans le paramètre `--policy-name`. Spécifiez de nouvelles valeurs pour les paramètres que vous souhaitez modifier.
- Lorsque vous utilisez l'API Application Auto Scaling, spécifiez le nom de la stratégie à modifier dans le paramètre `PolicyName`. Spécifiez de nouvelles valeurs pour les paramètres que vous souhaitez modifier.

Pour plus d'informations, consultez Application d'une stratégie de dimensionnement à un cluster de bases de données Aurora.

Suppression d'une stratégie d'autoscaling d'un cluster de bases de données Amazon Aurora

Vous pouvez supprimer une stratégie de mise à l'échelle à l'aide de la AWS Management Console, de la AWS CLI ou de l'API Application Auto Scaling.

Console

Vous pouvez supprimer une stratégie de dimensionnement à l'aide d AWS Management Console.

Pour supprimer une stratégie Auto Scaling pour un cluster de bases de données Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données.
3. Choisissez le cluster de bases de données Aurora dont vous voulez supprimer la stratégie Auto Scaling.
4. Choisissez l'onglet Logs & events (Journaux et événements).
5. Dans la section Auto scaling policies (Stratégies Auto Scaling), choisissez la stratégie Auto Scaling, puis Supprimer.

AWS CLI

Pour supprimer une stratégie de mise à l'échelle de votre cluster de bases de données Aurora, utilisez la commande de l'AWS CLI `delete-scaling-policy` avec les paramètres suivants :

- `--policy-name` – Nom de la stratégie de mise à l'échelle.

- `--resource-id` – Identifiant de la ressource du cluster de bases de données Aurora. Pour ce paramètre, le type de ressource est `cluster` et l'identifiant unique est le nom du cluster de bases de données Aurora, par exemple `cluster:myscalablecluster`.
- `--service-namespace` – Définissez cette valeur sur `rds`.
- `--scalable-dimension` – Définissez cette valeur sur `rds:cluster:ReadReplicaCount`.

Example

Dans l'exemple suivant, vous supprimez une stratégie de mise à l'échelle de suivi de la cible nommée `myscalablepolicy` d'un cluster de bases de données Aurora nommé `myscalablecluster`.

Pour Linux, macOS ou Unix :

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id cluster:myscalablecluster \
    --service-namespace rds \
    --scalable-dimension rds:cluster:ReadReplicaCount \
```

Pour Windows :

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id cluster:myscalablecluster ^
    --service-namespace rds ^
    --scalable-dimension rds:cluster:ReadReplicaCount ^
```

API Application Auto Scaling

Pour supprimer une stratégie de mise à l'échelle de votre cluster de bases de données Aurora, utilisez l'opération d'API Application Auto Scaling `DeleteScalingPolicy` avec les paramètres suivants :

- `PolicyName` – Nom de la stratégie de mise à l'échelle.
- `ServiceNamespace` – Définissez cette valeur sur `rds`.

- `ResourceID` – Identifiant de la ressource du cluster de bases de données Aurora. Pour ce paramètre, le type de ressource est `cluster` et l'identifiant unique est le nom du cluster de bases de données Aurora, par exemple `cluster:myscalablecluster`.
- `ScalableDimension` – Définissez cette valeur sur `rds:cluster:ReadReplicaCount`.

Example

Dans l'exemple suivant, vous supprimez une stratégie de mise à l'échelle de suivi de la cible nommée `myscalablepolicy` d'un cluster de bases de données Aurora nommé `myscalablecluster` à l'aide de l'API Application Auto Scaling.

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS

{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "rds",
    "ResourceId": "cluster:myscalablecluster",
    "ScalableDimension": "rds:cluster:ReadReplicaCount"
}
```

Gestion des performances et dimensionnement des clusters de bases de données Aurora

Vous pouvez utiliser les options suivantes pour gérer les performances et le dimensionnement des instances de base de données et des clusters de bases de données Aurora :

Rubriques

Dimensionnement du stockage

Le stockage Aurora procède à un dimensionnement automatique avec les données de votre volume de cluster. Lorsque vos données augmentent, le volume de stockage de votre cluster s'élargit en fonction de la version du moteur de base de données. Pour plus d'informations sur les tailles de volume maximales des clusters Aurora pour chaque version de moteur, consultez Limites de taille Amazon Aurora. Pour connaître le type des données incluses dans le volume de cluster, consultez Stockage Amazon Aurora. Pour plus d'informations sur la taille maximale d'une version spécifique, consultez Limites de taille Amazon Aurora.

La taille de votre volume de cluster est évaluée toutes les heures afin de déterminer vos coûts de stockage. Pour obtenir des informations sur la tarification, consultez la page de tarification Aurora.

Même si la taille d'un volume de cluster Aurora peut augmenter jusqu'à plusieurs tébioctets, vous n'êtes facturé que pour l'espace que vous utilisez dans le volume. Le mécanisme de détermination de l'espace de stockage facturé dépend de la version de votre cluster Aurora.

- Lorsque des données Aurora sont supprimées du volume de cluster, l'espace facturé global diminue d'un montant comparable. Ce comportement de redimensionnement dynamique se produit lorsque des espaces de table sous-jacents sont supprimés ou réorganisés pour nécessiter moins d'espace. Ainsi, vous pouvez réduire les frais de stockage en supprimant les tables et les bases de données dont vous n'avez plus besoin. Le redimensionnement dynamique s'applique à certaines

versions d'Aurora. Les versions suivantes sont les versions Aurora pour lesquelles le volume de cluster est redimensionné dynamiquement lorsque vous supprimez des données :

Moteur de base de données	Versions avec redimensionnement dynamique
Aurora MySQL	• Version 3 (compatible avec MySQL 8.0) : toutes versions prises en charge • Version 2 (compatible avec MySQL 5.7) : versions 2.11 et ultérieures
Aurora PostgreSQL	Toutes les versions prises en charge
Aurora Serverless v2	Toutes les versions prises en charge
Aurora Serverless v1	Non disponible

- Dans les versions Aurora antérieures à celles de la liste précédente, le volume du cluster peut réutiliser l'espace libéré lors de la suppression des données, mais le volume lui-même ne diminue jamais.

Le redimensionnement dynamique s'applique aux opérations qui suppriment ou redimensionnent physiquement des espaces de table dans le volume de cluster. Ainsi, il s'applique aux instructions SQL telles que `DROP TABLE`, `DROP DATABASE`, `TRUNCATE TABLE` et `ALTER TABLE ... DROP PARTITION`. Il ne s'applique pas à la suppression de lignes à l'aide de l'instruction `DELETE`. Si vous supprimez un grand nombre de lignes d'une table, vous pouvez exécuter l'instruction Aurora MySQL `OPTIMIZE TABLE` ou l'extension Aurora PostgreSQL `pg_repack` par la suite afin de réorganiser la table et redimensionner dynamiquement le volume de cluster.

Pour Aurora MySQL, les considérations suivantes s'appliquent :

- Une fois que vous avez mis à niveau votre cluster de bases de données vers une version du moteur de base de données qui prend en charge le redimensionnement dynamique, et lorsque cette fonctionnalité est activée dans cette Région AWS spécifique, tout espace libéré ultérieurement par certaines instructions SQL, telles que `DROP TABLE`, est récupérable.

  Si cette fonctionnalité est explicitement désactivée dans une Région AWS spécifique, l'espace pourra uniquement être réutilisable, et non récupérable, même sur les versions qui prennent en charge le redimensionnement dynamique.

Cette fonctionnalité a été activée pour des versions spécifiques du moteur de base de données (de 1.23.0 à 1.23.4, de 2.09.0 à 2.09.3 et 2.10.0) entre novembre 2020 et mars 2022, et est activée par défaut sur toutes les versions ultérieures.

- Une table est stockée en interne dans un ou plusieurs fragments contigus de différentes tailles. Lors de l'exécution d'opérations `TRUNCATE TABLE`, l'espace correspondant au premier fragment est réutilisable et non récupérable. Les autres fragments sont récupérables. Pendant les opérations `DROP TABLE`, l'espace correspondant à l'ensemble de l'espace de table est récupérable.
- Le paramètre `innodb_file_per_table` affecte la façon dont le stockage de table est organisé. Lorsque les tables font partie de l'espace de tables système, la suppression de la table ne réduit pas la taille de l'espace de tables système. Par conséquent, assurez-vous de définir `innodb_file_per_table` sur 1 pour que les clusters de bases de données Aurora MySQL tirent pleinement parti du redimensionnement dynamique.
- Dans les versions 2.11 et ultérieures, l'espace de table temporaire InnoDB est supprimé et recréé au redémarrage. Cela libère l'espace occupé par l'espace de table temporaire vers le système, puis le volume du cluster est redimensionné. Pour tirer pleinement parti de la fonction de redimensionnement dynamique, nous vous recommandons de mettre à niveau votre cluster de bases de données vers les versions 2.11 ou ultérieures.

> **Note**
>
> La fonctionnalité de redimensionnement dynamique ne permet pas de récupérer de l'espace immédiatement lorsque les tables des espaces de tables sont supprimées, mais progressivement à un rythme d'environ 10 To par jour. L'espace de l'espace de table du système n'est pas récupéré, car l'espace de table du système n'est jamais supprimé. L'espace libre non récupéré dans un espace de table est réutilisé lorsqu'une opération a besoin d'espace dans cet espace de table. La fonctionnalité de redimensionnement dynamique peut récupérer de l'espace de stockage uniquement lorsque le cluster est dans un état disponible.

Vous pouvez vérifier l'espace de stockage utilisé par un cluster en surveillant la métrique `VolumeBytesUsed` dans CloudWatch. Pour plus d'informations sur la facturation du stockage, consultez Facturation du stockage des données Aurora.

- Dans AWS Management Console, cette valeur apparaît dans un graphique sous l'onglet `Monitoring` de la page des détails du cluster.
- Avec l'AWS CLI, vous pouvez exécuter une commande similaire à l'exemple Linux suivant. Indiquez vos propres valeurs pour les heures de début et de fin, ainsi que pour le nom du cluster.

```
aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '6 hours ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" \
  --statistics Average Maximum Minimum \
  --dimensions Name=DBClusterIdentifier,Value=my_cluster_identifier
```

Le résultat de cette commande est semblable à ce qui suit.

```
{
    "Label": "VolumeBytesUsed",
    "Datapoints": [
        {
            "Timestamp": "2020-08-04T21:25:00+00:00",
            "Average": 182871982080.0,
            "Minimum": 182871982080.0,
            "Maximum": 182871982080.0,
            "Unit": "Bytes"
        }
    ]
}
```

Les exemples suivants montrent comment vous pouvez suivre l'utilisation du stockage pour un cluster Aurora au fil du temps à l'aide de commandes AWS CLI sur un système Linux. Les paramètres `--start-time` et `--end-time` définissent l'intervalle de temps global comme un jour. Le paramètre `--period` demande les métriques par intervalles d'une heure. Choisir une valeur `--period` faible n'a aucun sens, car les métriques sont collectées à intervalles réguliers, non en continu. En outre, les opérations de stockage Aurora se poursuivent parfois pendant un certain temps en arrière-plan après la fin de l'instruction SQL pertinente.

Le premier exemple renvoie la sortie au format JSON par défaut. Les points de données sont renvoyés dans un ordre arbitraire, et non triés par horodatage. Vous pouvez importer ces données JSON dans un outil de mise en forme graphique pour effectuer des opérations de tri et de visualisation.

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '1 day ago')" --end-time "$(date -d 'now')" --period 3600
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_cluster_id
{
    "Label": "VolumeBytesUsed",
    "Datapoints": [
        {
            "Timestamp": "2020-08-04T19:40:00+00:00",
            "Maximum": 182872522752.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-05T00:40:00+00:00",
            "Maximum": 198573719552.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-05T05:40:00+00:00",
            "Maximum": 206827454464.0,
            "Unit": "Bytes"
        },
        {
            "Timestamp": "2020-08-04T17:40:00+00:00",
            "Maximum": 182872522752.0,
            "Unit": "Bytes"
        },
... output omitted ...
```

Cet exemple renvoie les mêmes données que le précédent. Le paramètre `--output` représente les données au format texte brut compact. La commande `aws cloudwatch` dirige sa sortie vers la commande `sort`. Le paramètre `-k` de la commande `sort` trie la sortie sur le troisième champ, qui est l'horodatage au format UTC (Universal Coordinated Time).

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '1 day ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_cluster_id \
  --output text | sort -k 3
VolumeBytesUsed
DATAPOINTS  182872522752.0  2020-08-04T17:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T18:41:00+00:00 Bytes
```

```
DATAPOINTS  182872522752.0  2020-08-04T19:41:00+00:00 Bytes
DATAPOINTS  182872522752.0  2020-08-04T20:41:00+00:00 Bytes
DATAPOINTS  187667791872.0  2020-08-04T21:41:00+00:00 Bytes
DATAPOINTS  190981029888.0  2020-08-04T22:41:00+00:00 Bytes
DATAPOINTS  195587244032.0  2020-08-04T23:41:00+00:00 Bytes
DATAPOINTS  201048915968.0  2020-08-05T00:41:00+00:00 Bytes
DATAPOINTS  205368492032.0  2020-08-05T01:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T02:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T03:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T04:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T05:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T06:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T07:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T08:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T09:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T10:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T11:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T12:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T13:41:00+00:00 Bytes
DATAPOINTS  206827454464.0  2020-08-05T14:41:00+00:00 Bytes
DATAPOINTS  206833664000.0  2020-08-05T15:41:00+00:00 Bytes
DATAPOINTS  206833664000.0  2020-08-05T16:41:00+00:00 Bytes
```

La sortie triée indique la quantité de stockage utilisée au début et à la fin de la période de surveillance. Vous pouvez également trouver les points pendant cette période lorsqu'Aurora alloue plus de stockage pour le cluster. L'exemple suivant utilise des commandes Linux pour reformater les valeurs `VolumeBytesUsed` de début et de fin en gigaoctets (Go) et en gibioctets (Gio). Les gigaoctets représentent des unités mesurées en puissances de 10 et sont couramment utilisées dans les discussions sur le stockage pour des disques durs rotatifs. Les gibioctets représentent des unités mesurées en puissances de 2. Les mesures et limites de stockage Aurora sont généralement indiquées sous la forme d'unités de puissance de 2, telles que les gibioctets et les téraoctets.

```
$ GiB=$((1024*1024*1024))
$ GB=$((1000*1000*1000))
$ echo "Start: $((182872522752/$GiB)) GiB, End: $((206833664000/$GiB)) GiB"
Start: 170 GiB, End: 192 GiB
$ echo "Start: $((182872522752/$GB)) GB, End: $((206833664000/$GB)) GB"
Start: 182 GB, End: 206 GB
```

La métrique `VolumeBytesUsed` vous indique la quantité de stockage dans le cluster qui génère des frais. Il est donc préférable de réduire ce nombre lorsque c'est possible. Toutefois, cette métrique

n'inclut pas le stockage utilisé en interne par Aurora dans le cluster et qui n'est pas facturé. Si votre cluster approche la limite de stockage et risque de manquer d'espace, il est conseillé de surveiller la métrique `AuroraVolumeBytesLeftTotal` et d'essayer d'augmenter ce nombre. L'exemple suivant exécute un calcul similaire au précédent, mais pour `AuroraVolumeBytesLeftTotal` au lieu de `VolumeBytesUsed`.

```
$ aws cloudwatch get-metric-statistics --metric-name "AuroraVolumeBytesLeftTotal" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Maximum --dimensions
 Name=DBClusterIdentifier,Value=my_old_cluster_id \
  --output text | sort -k 3
AuroraVolumeBytesLeftTotal
DATAPOINTS      140530528288768.0       2023-02-23T19:25:00+00:00       Count
$ TiB=$((1024*1024*1024*1024))
$ TB=$((1000*1000*1000*1000))
$ echo "$((69797067915264 / $TB)) TB remaining for this cluster"
69 TB remaining for this cluster
$ echo "$((69797067915264 / $TiB)) TiB remaining for this cluster"
63 TiB remaining for this cluster
```

Pour un cluster exécutant Aurora MySQL version 2.09 ou ultérieure, ou Aurora PostgreSQL, la taille libre indiquée par `VolumeBytesUsed` augmente quand des données sont ajoutées et diminue quand des données sont supprimées. L'exemple suivant montre comment procéder. Ce rapport indique la taille de stockage maximale et minimale d'un cluster par intervalles de 15 minutes lorsque des tables contenant des données temporaires sont créées et supprimées. Le rapport indique la valeur maximale avant la valeur minimale. Ainsi, pour comprendre comment l'utilisation du stockage a changée au cours de l'intervalle de 15 minutes, interprétez les chiffres de droite à gauche.

```
$ aws cloudwatch get-metric-statistics --metric-name "VolumeBytesUsed" \
  --start-time "$(date -d '4 hours ago')" --end-time "$(date -d 'now')" --period 1800 \
  --namespace "AWS/RDS" --statistics Maximum Minimum --dimensions
 Name=DBClusterIdentifier,Value=my_new_cluster_id
  --output text | sort -k 4
VolumeBytesUsed
DATAPOINTS 14545305600.0 14545305600.0 2020-08-05T20:49:00+00:00 Bytes
DATAPOINTS 14545305600.0 14545305600.0 2020-08-05T21:19:00+00:00 Bytes
DATAPOINTS 22022176768.0 14545305600.0 2020-08-05T21:49:00+00:00 Bytes
DATAPOINTS 22022176768.0 22022176768.0 2020-08-05T22:19:00+00:00 Bytes
DATAPOINTS 22022176768.0 22022176768.0 2020-08-05T22:49:00+00:00 Bytes
DATAPOINTS 22022176768.0 15614263296.0 2020-08-05T23:19:00+00:00 Bytes
DATAPOINTS 15614263296.0 15614263296.0 2020-08-05T23:49:00+00:00 Bytes
```

```
DATAPOINTS 15614263296.0 15614263296.0 2020-08-06T00:19:00+00:00 Bytes
```

L'exemple suivant montre comment avec des clusters exécutant des versions compatibles d'Aurora MySQL ou d'Aurora PostgreSQL, la taille libre indiquée par `AuroraVolumeBytesLeftTotal` reflète la limite de taille de 256 Tio. Pour plus d'informations sur les versions compatibles, consultez Limites de taille Amazon Aurora.

```
$ aws cloudwatch get-metric-statistics --region us-east-1 --metric-name
 "AuroraVolumeBytesLeftTotal" \
  --start-time "$(date -d '4 hours ago')" --end-time "$(date -d 'now')" --period 1800 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBClusterIdentifier,Value=pq-57 \
  --output text | sort -k 3
AuroraVolumeBytesLeftTotal
DATAPOINTS 140515818864640.0 2020-08-05T20:56:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-05T21:26:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-05T21:56:00+00:00 Count
DATAPOINTS 140514866757632.0 2020-08-05T22:26:00+00:00 Count
DATAPOINTS 140511020580864.0 2020-08-05T22:56:00+00:00 Count
DATAPOINTS 140503168843776.0 2020-08-05T23:26:00+00:00 Count
DATAPOINTS 140503168843776.0 2020-08-05T23:56:00+00:00 Count
DATAPOINTS 140515818864640.0 2020-08-06T00:26:00+00:00 Count
$ TiB=$((1024*1024*1024*1024))
$ TB=$((1000*1000*1000*1000))
$ echo "$((140515818864640 / $TB)) TB remaining for this cluster"
140 TB remaining for this cluster
$ echo "$((140515818864640 / $TiB)) TiB remaining for this cluster"
256 TiB remaining for this cluster
```

Mise à l'échelle d'instances

Vous pouvez mettre à l'échelle votre cluster de bases de données Aurora en modifiant la classe d'instance de base de données pour chaque instance dans le cluster. Aurora prend en charge plusieurs classes d'instance de base de données optimisées pour Aurora, en fonction de la compatibilité du moteur de base de données.

Moteur de base de données	Mise à l'échelle d'instances
Amazon Aurora MySQL	Consultez Dimensionnement des instances de bases de données Aurora MySQL

Moteur de base de données	Mise à l'échelle d'instances
Amazon Aurora PostgreSQL	Consultez Dimensionnement des instances de base de données Aurora PostgreSQL

Dimensionnement en lecture

Vous pouvez réaliser le dimensionnement en lecture de votre cluster de bases de données Aurora en créant jusqu'à 15 réplicas Aurora dans un cluster de bases de données. Chaque réplica Aurora retourne les mêmes données du volume de cluster avec un retard de réplica minimal, généralement très inférieur à 100 ms, après que l'instance principale a écrit une mise à jour. Tandis que votre trafic en lecture augmente, vous pouvez créer des réplicas Aurora additionnels et vous y connecter directement pour répartir la charge de lecture de votre cluster de bases de données. Les réplicas Aurora n'ont pas à être de la même classe d'instance de base de données que l'instance principale.

Pour plus d'informations sur l'ajout de réplicas Aurora à un cluster de bases de données, consultez Ajout de réplicas Aurora à un cluster de bases de données.

Gestion des connexions

Le nombre maximal de connexions autorisées à une instance de base de données Aurora est déterminé par le paramètre `max_connections` du groupe de paramètres de niveau instance de l'instance de base de données. La valeur par défaut de ce paramètre varie en fonction de la classe d'instance utilisée pour l'instance de base de données et de la compatibilité du moteur de bases de données.

Moteur de base de données	Valeur par défaut de max_connections
Amazon Aurora MySQL	Consultez Nombre maximal de connexions à une instance de base de données Aurora MySQL
Amazon Aurora PostgreSQL	Consultez Nombre maximal de connexions à une instance de base de données Aurora PostgreSQL

> **Tip**
>
> Si vos applications ouvrent et ferment fréquemment des connexions, ou si elles ont ouvert un grand nombre de connexions de longue durée, nous vous recommandons d'utiliser Proxy Amazon RDS. RDS Proxy est un proxy de base de données entièrement géré et hautement disponible qui utilise le regroupement de connexions pour partager les connexions de base de données de manière sécurisée et efficace. Pour en savoir plus sur RDS Proxy, consultez Proxy Amazon RDS pour Aurora.

Gestion des plans d'exécution de requêtes

Si vous utilisez la gestion des plans d'exécution de requêtes pour Aurora PostgreSQL, vous prenez le contrôle des plans exécutés par l'optimiseur. Pour plus d'informations, consultez Gestion des plans d'exécution de requêtes pour Aurora PostgreSQL.

Clonage d’un volume pour un cluster de bases de données Amazon Aurora

Le clonage Aurora vous permet de créer un nouveau cluster qui partage les mêmes pages de données que l’original, mais qui est un volume distinct et indépendant. Le processus est conçu pour être rapide et rentable. Le nouveau cluster avec son volume de données associé est appelé clone. La création d’un clone est plus rapide et plus économe en espace que la copie physique des données à l’aide d’autres techniques telles que la restauration d’instantané.

Rubriques

Présentation du clonage Aurora

Pour créer un clone, Aurora utilise un protocole de copie sur écriture. Ce mécanisme utilise un espace supplémentaire minimal pour créer un clone initial. Lors de la création du premier clone, Aurora conserve une seule copie des données qu’utilisent le cluster de bases de données Aurora source et le nouveau cluster de bases de données Aurora (cloné). Un stockage supplémentaire n’est alloué que quand des modifications sont apportées aux données (sur le volume de stockage Aurora) par le cluster de bases de données Aurora source ou le clone du cluster de bases de données Aurora. Pour en savoir plus sur le protocole de copie sur écriture, consultez Fonctionnement du clonage Aurora.

Le clonage Aurora est particulièrement utile pour configurer rapidement des environnements de test à l’aide de vos données de production, sans risque de corruption des données. Vous pouvez utiliser des clones pour de nombreux types d’applications, telles que les suivantes :

- Expérimentez des changements potentiels (par exemple, des changements de schémas et de groupes de paramètres) pour évaluer tous les impacts.

- Exécutez des opérations imposant une charge de travail élevée, telles que l'exportation de données ou l'exécution de requêtes analytiques sur le clone.
- Créez une copie de votre cluster de bases de données de production à des fins de développement, de test ou autres.

Vous pouvez créer plusieurs clones à partir du même cluster de bases de données Aurora. Vous pouvez également créer plusieurs clones à partir d'un autre clone.

Après avoir créé un clone Aurora, vous pouvez configurer les instances de base de données Aurora différemment du cluster de bases de données Aurora source. Par exemple, il se peut que vous n'ayez pas besoin d'un clone à des fins de développement pour répondre aux mêmes exigences de haute disponibilité que le cluster de bases de données Aurora de production source. Dans ce cas, vous pouvez configurer le clone avec une seule instance de base de données Aurora plutôt que les multiples instances de base de données qu'utilise le cluster de bases de données Aurora.

Lorsque vous créez un clone à l'aide d'une configuration de déploiement différente de la source, le clone est créé à l'aide de la dernière version mineure du moteur de base de données Aurora de la source.

Lorsque vous créez des clones à partir de vos clusters de bases de données Aurora, les clones sont créés dans votre compte AWS contenant déjà le cluster de bases de données Aurora source. Toutefois, vous pouvez également partager des clusters de bases de données Aurora provisionnés et Aurora Serverless v2 et des clones avec d'autres comptes AWS. Pour plus d'informations, consultez Clonage entre comptes avec Amazon Aurora AWS RAM et Amazon.

Lorsque vous avez fini d'utiliser le clone à des fins de test, de développement ou autres, vous pouvez le supprimer.

Limites du clonage Aurora

Le clonage Aurora présente actuellement les limitations suivantes :

- Vous pouvez créer autant de clones que vous le souhaitez, jusqu'au nombre maximal de clusters de bases de données autorisés dans la Région AWS.
- Vous pouvez créer jusqu'à 15 clones avec le protocole de copie sur écriture. Lorsque vous avez créé 15 clones, le prochain clone créé est une copie intégrale. Le protocole de copie intégrale agit comme une reprise ponctuelle.

- Vous ne pouvez pas créer de clone dans une région AWS différente de celle du cluster de bases de données Aurora source.
- Vous ne pouvez pas créer un clone à partir d'un cluster de bases de données Aurora dépourvu de la fonction de requête parallèle vers un cluster utilisant la fonction de requête parallèle. Pour introduire des données dans un cluster utilisant la fonction de requête parallèle, créez un instantané du cluster d'origine et restaurez-le dans un cluster utilisant la fonction de requête parallèle.
- Vous ne pouvez pas créer de clone à partir d'un cluster de bases de données Aurora dépourvu d'instances de base de données. Vous ne pouvez cloner que des clusters de bases de données Aurora ayant au moins une instance de base de données.
- Vous pouvez créer un clone dans un cloud privé virtuel (VPC) différent de celui du cluster de bases de données Aurora. Cependant, les sous-réseaux des VPC doivent mapper aux mêmes zones de disponibilité.
- Vous pouvez créer un clone approvisionné Aurora à partir d'un cluster de bases de données Aurora approvisionné.
- Les clusters avec instances Aurora Serverless v2 suivent les mêmes règles que les clusters alloués.
- Pour Aurora Serverless v1 :
  - Vous pouvez créer un clone provisionné à partir d'un cluster de bases de données Aurora Serverless v1.
  - Vous pouvez créer un clone Aurora Serverless v1 à partir d'un cluster de bases de données Aurora Serverless v1 ou provisionné.
  - Vous ne pouvez pas créer un clone Aurora Serverless v1 à partir d'un cluster de bases de données Aurora provisionné non chiffré.
  - Actuellement, le clonage entre comptes ne prend pas en charge le clonage de clusters de bases de données Aurora Serverless v1. Pour plus d'informations, consultez Limites du clonage intercompte.
  - Un cluster de bases de données Aurora Serverless v1 cloné a le même comportement et les mêmes limitations que tout cluster de bases de données Aurora Serverless v1. Pour plus d'informations, consultez Utilisation d'Amazon Aurora Serverless v1.
  - Les clusters de bases de données Aurora Serverless v1 sont toujours chiffrés. Lorsque vous clonez un cluster de bases de données Aurora Serverless v1 dans un cluster de bases de données Aurora approvisionné, le cluster de bases de données Aurora approvisionné est chiffré. Vous pouvez choisir la clé de chiffrement, mais pas désactiver le chiffrement. Pour créer un

clone à partir d'un cluster de bases de données Aurora provisionné vers un cluster Aurora Serverless v1, vous devez commencer avec un cluster de bases de données Aurora provisionné chiffré.

Fonctionnement du clonage Aurora

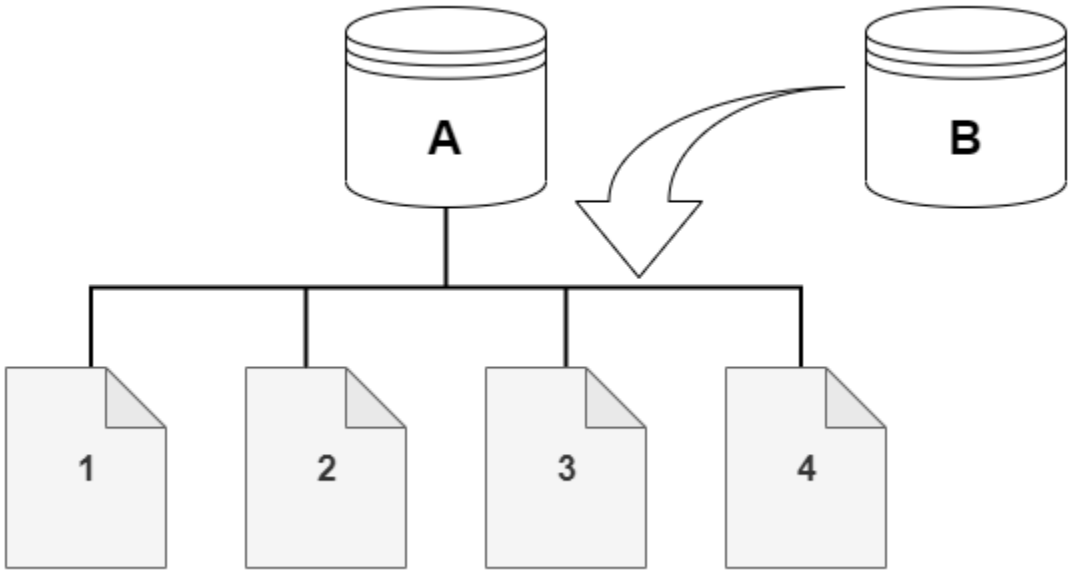
Le clonage Aurora opère au niveau de la couche de stockage d'un cluster de bases de données Aurora. Il utilise un protocole de copie sur écriture à la fois rapide et économe en espace s'agissant du support durable sous-jacent du volume de stockage Aurora. Pour plus d'informations sur les volumes de cluster Aurora, consultez Présentation du stockage Amazon Aurora.

Rubriques

Présentation du protocole de copie sur écriture

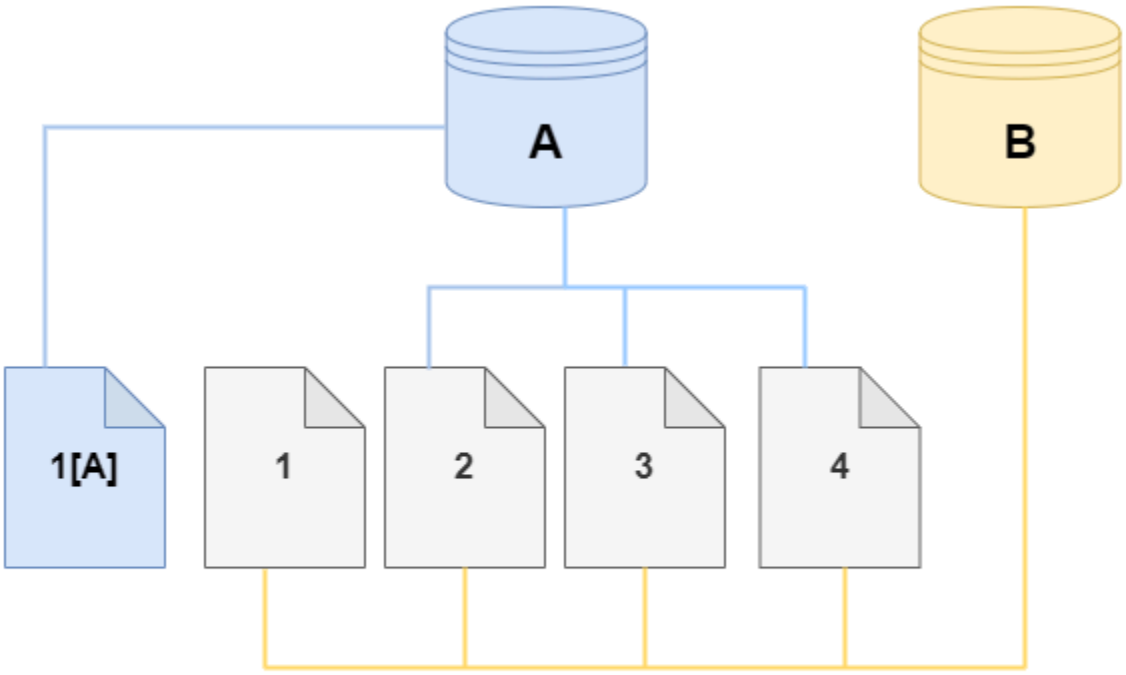
Un cluster de bases de données Aurora stocke des données dans des pages du volume de stockage Aurora sous-jacent.

Par exemple, le diagramme suivant montre un cluster de bases de données Aurora (A) comptant quatre pages de données, 1, 2, 3 et 4. Imaginez qu'un clone, B, soit créé à partir du cluster de bases de données Aurora. Lors de la création du clone, aucune donnée n'est copiée. Au contraire, le clone pointe vers le même ensemble de pages que le cluster de bases de données Aurora source.

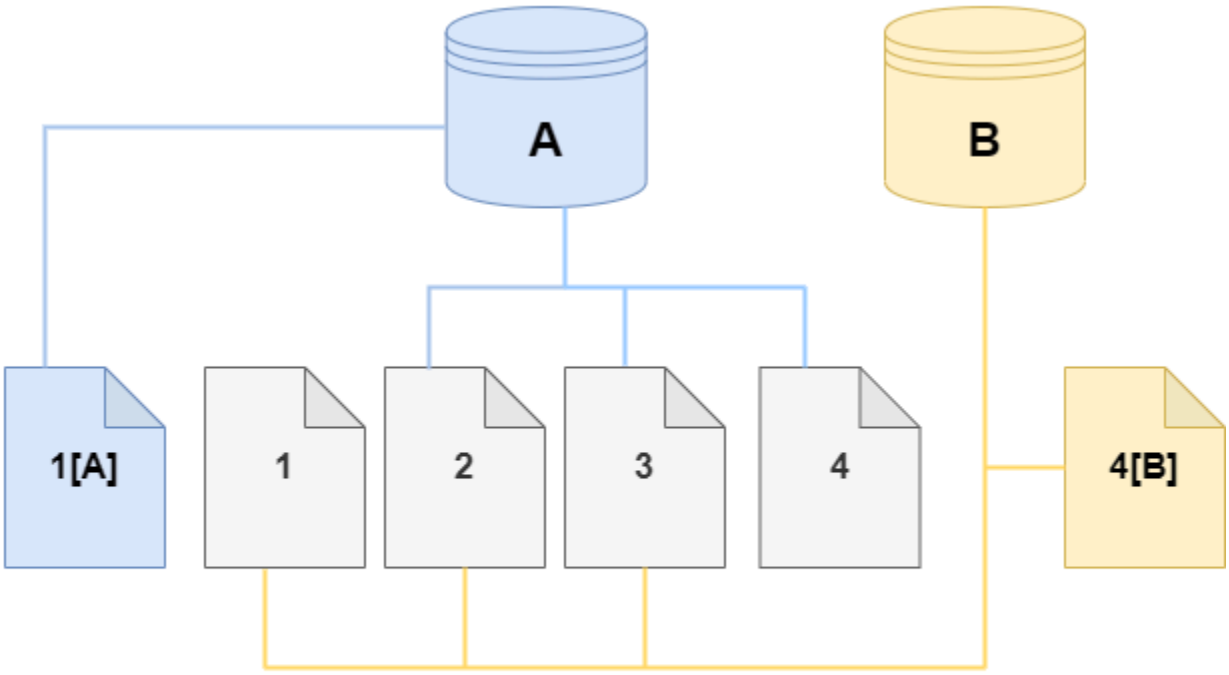


Lors de la création du clone, aucun stockage supplémentaire n’est généralement nécessaire. Le protocole de copie sur écriture utilise le même segment sur le support de stockage physique que le segment source. Un stockage supplémentaire n’est requis que si la capacité du segment source n’est pas suffisante pour le segment de clone entier. Dans ce cas, le segment source est copié sur un autre périphérique physique.

Les diagrammes suivants présentent un exemple de protocole de copie sur écriture en action utilisant les cluster A et clone B que ceux montrés précédemment. Supposons que vous apportez une modification à votre cluster de bases de données Aurora (A) qui entraîne un changement des données conservées sur la page 1. Au lieu d’écrire sur la page 1 d’origine, Aurora crée une nouvelle page 1[A]. Le volume de cluster de bases de données Aurora pour le cluster (A) pointe désormais vers les pages 1[A], 2, 3 et 4, tandis que le clone (B) continue de référencer les pages d’origine.



Sur le clone, une modification est apportée à la page 4 sur le volume de stockage. Au lieu d'écrire sur la page 4 d'origine, Aurora crée une nouvelle page 1[B]. Le clone pointe maintenant vers les pages 1, 2, 3 et 4[B], tandis que le cluster (A) continue de pointer vers les pages 1[A], 2, 3 et 4.



A mesure que des modifications supplémentaires sont apportées tant au volume de cluster Aurora source qu'au clone, plus de stockage est nécessaire pour capturer et stocker les modifications.

Suppression d'un volume de cluster source

Au départ, le volume du clone partage les mêmes pages de données que le volume d'origine à partir duquel le clone est créé. Tant que le volume d'origine existe, le volume du clone est uniquement considéré comme le propriétaire des pages créées ou modifiées par ce clone. La métrique `VolumeBytesUsed` du volume du clone est donc faible au départ et augmente uniquement à mesure que les données divergent entre le cluster d'origine et le clone. Pour les pages qui sont identiques entre le volume source et le clone, les frais de stockage s'appliquent uniquement au cluster d'origine. Pour plus d'informations sur la métrique `VolumeBytesUsed`, consultez Métriques de niveau cluster pour Amazon Aurora.

Lorsque vous supprimez un volume de cluster source auquel un ou plusieurs clones sont associés, les données des volumes de cluster des clones ne sont pas modifiées. Aurora préserve les pages qui étaient précédemment la propriété du volume de cluster source. Aurora redistribue la facturation

du stockage pour les pages qui appartenaient au cluster supprimé. Supposons, par exemple, qu'un cluster d'origine associé à deux clones soit supprimé. La moitié des pages de données qui étaient la propriété du cluster d'origine appartiennent maintenant à un clone. L'autre moitié des pages appartient à l'autre clone.

Si vous supprimez le cluster d'origine, au fur et à mesure que vous créez ou supprimez d'autres clones, Aurora continue de redistribuer la propriété des pages de données entre tous les clones partageant les mêmes pages. Il se peut donc que la valeur de la métrique `VolumeBytesUsed` change pour le volume de cluster d'un clone. La valeur de cette métrique peut diminuer lorsque de nouveaux clones sont créés et que la propriété des pages est répartie sur un plus grand nombre de clusters. La valeur de cette métrique peut également augmenter lorsque des clones sont supprimés et que la propriété des pages est attribuée à un plus petit nombre de clusters. Pour en savoir plus sur l'impact des opérations d'écriture sur les pages de données des volumes des clones, consultez Présentation du protocole de copie sur écriture.

Lorsque le cluster d'origine et les clones appartiennent au même compte AWS, tous les frais de stockage de ces clusters s'appliquent à ce même compte AWS. Si certains clusters sont des clones entre comptes, la suppression du cluster d'origine peut entraîner des frais de stockage supplémentaires pour les comptes AWS propriétaires des clones entre comptes.

Supposons, par exemple, qu'un volume de cluster compte 1 000 pages de données utilisées avant de créer des clones. Lorsque vous clonez ce cluster, le volume du clone ne contient initialement aucune page utilisée. Si le clone apporte des modifications à 100 pages de données, seules ces 100 pages sont stockées sur le volume du clone et marquées comme utilisées. Les 900 pages inchangées restantes du volume parent sont partagées par les deux clusters. Dans ce cas, le cluster parent facture des frais de stockage pour 1 000 pages et le volume du clone pour 100 pages.

Si vous supprimez le volume source, les frais de stockage pour le clone incluent les 100 pages modifiées, plus les 900 pages partagées du volume d'origine, pour un total de 1 000 pages.

Création d'un clone Amazon Aurora

Vous pouvez créer un clone dans le même compte AWS que celui du cluster de bases de données Aurora source. Pour ce faire, vous pouvez utiliser l'AWS Management Console ou l'AWS CLI et les procédures suivantes.

Pour autoriser un autre compte AWS à créer un clone, ou pour partager un clone avec un autre compte AWS, suivez les procédures décrites dans Clonage entre comptes avec Amazon Aurora AWS RAM et Amazon.

Console

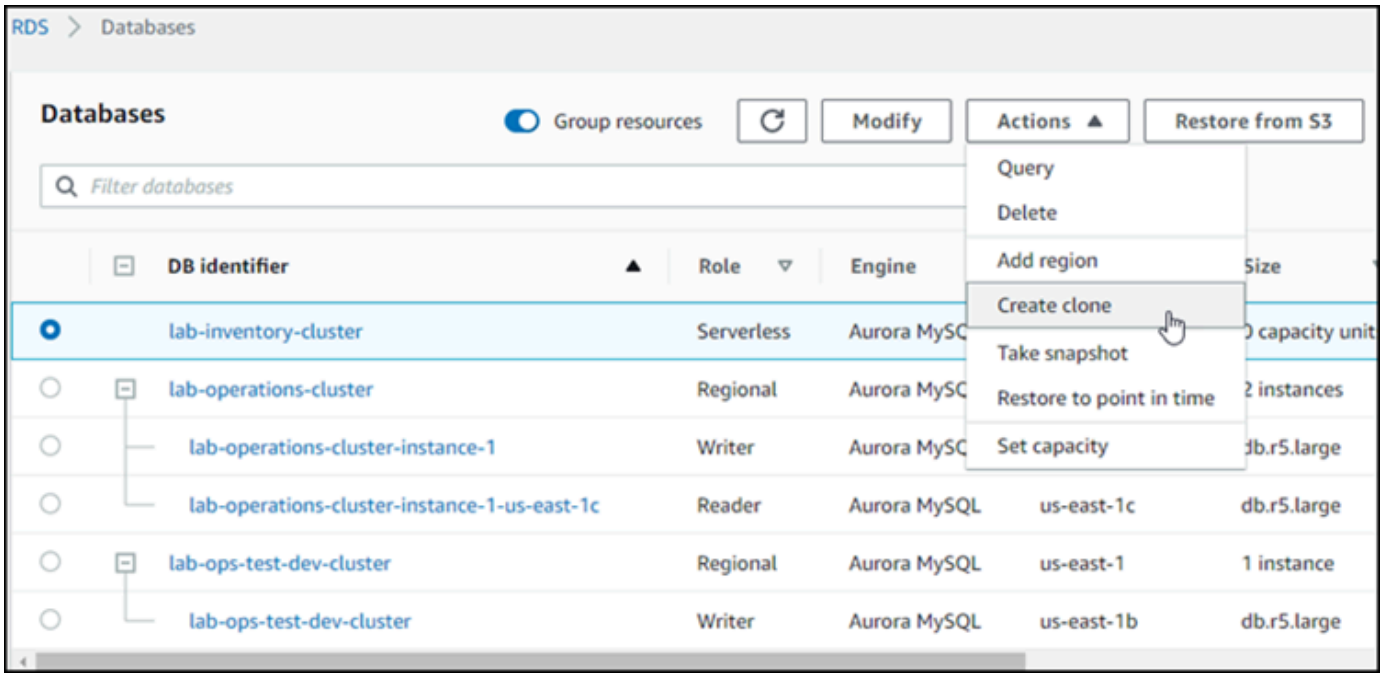
La procédure suivante explique comment cloner un cluster de bases de données Aurora à l'aide de l'AWS Management Console.

Création d'un clone à l'aide des résultat de l'AWS Management Console dans un cluster de bases de données Aurora avec une instance de base de données Aurora.

Ces instructions s'appliquent aux clusters de bases de données appartenant au même compte AWS que celui qui crée le clone. Si le cluster de bases de données appartient à un compte AWS différent, consultez plutôt Clonage entre comptes avec Amazon Aurora AWS RAM et Amazon.

Pour créer un clone d'un cluster de bases de données appartenant à votre compte AWS via AWS Management Console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez votre cluster de bases de données Aurora dans la liste, puis, pour Actions, choisissez Create clone (Créer un clone).



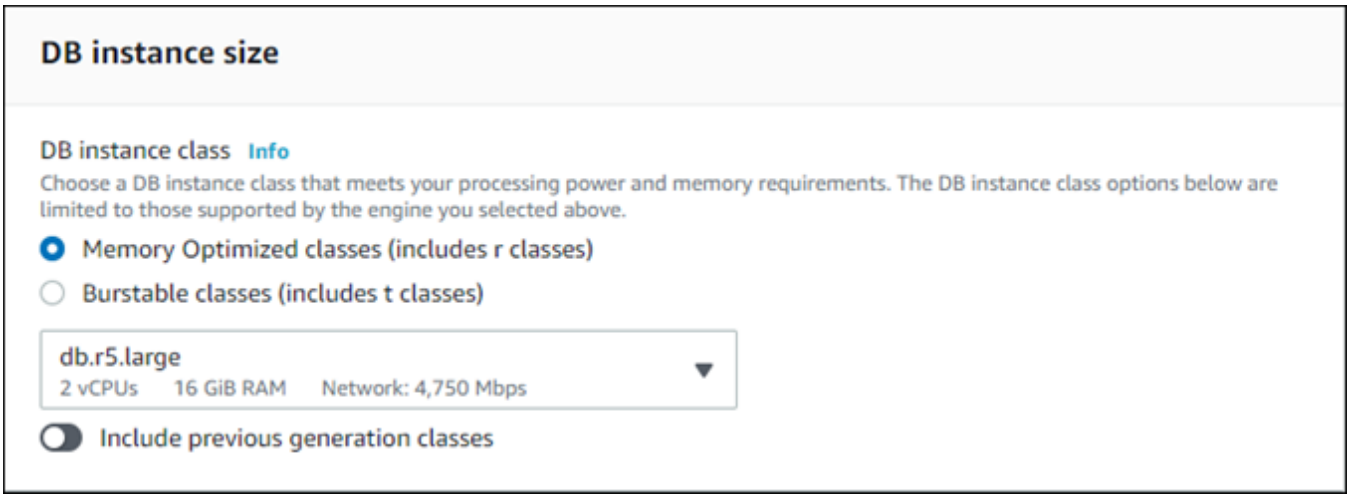
La page Créer un clone s'ouvre. Vous pouvez y configurer les options Paramètres, Connectivité et d'autres options pour le clone de cluster de bases de données Aurora.

4. Pour Identifiant d'instance de base de données, entrez le nom que vous souhaitez donner à votre cluster de bases de données Aurora cloné.
5. Pour les clusters de bases de données Aurora Serverless v1, choisissez Provisionné ou Sans serveur pour Type de capacité.

   Vous ne pouvez choisir Sans serveur que si le cluster de bases de données Aurora source est un cluster de bases de données Aurora Serverless v1 ou un cluster de bases de données Aurora approvisionné qui est chiffré.
6. Pour les clusters de bases de données Aurora Serverless v2 ou provisionnés, choisissez Aurora I/O-Optimized ou Aurora Standard pour Configuration du stockage en cluster.

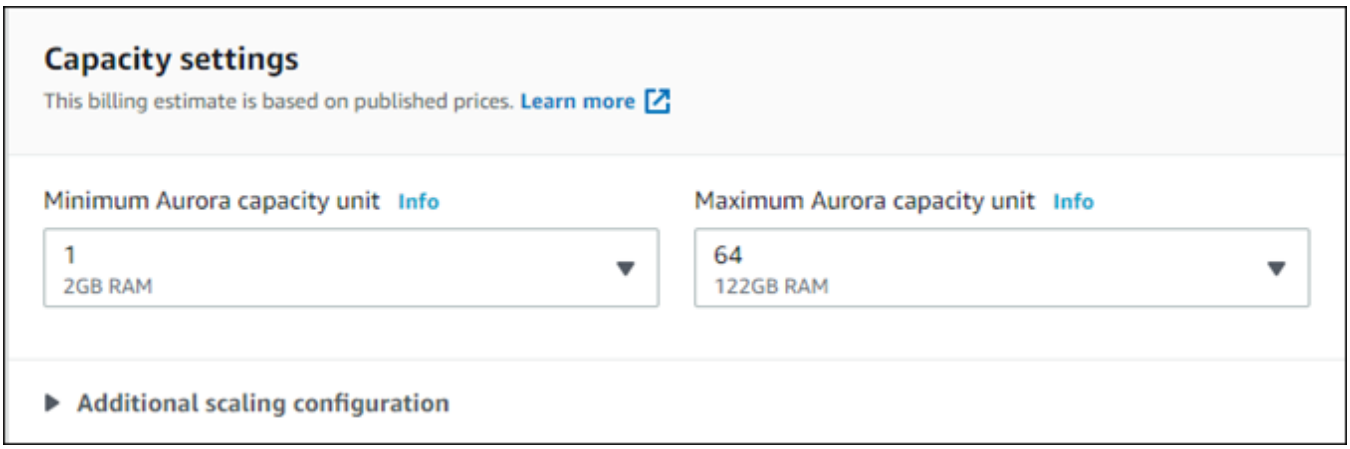
   Pour plus d'informations, consultez Configurations de stockage pour les clusters de bases de données Amazon Aurora.
7. Choisissez la taille de l'instance de base de données ou la capacité du cluster de bases de données :

   - Pour un clone provisionné, choisissez une classe d'instance de base de données.



     Vous pouvez accepter le paramètre fourni ou utiliser une autre classe d'instance de base de données différente pour votre clone.

   - Pour un clone Aurora Serverless v1 ou Aurora Serverless v2, choisissez les paramètres de capacité.



Vous pouvez accepter les paramètres fournis ou les modifier pour votre clone.

8. Choisissez les autres paramètres nécessaires pour votre clone. Pour en savoir plus sur les paramètres de cluster et d'instance de base de données Aurora, consultez Création d'un cluster de bases de données Amazon Aurora.
9. Choisissez Créer un clone.

Une fois le clone créé, il est répertorié avec vos autres clusters de bases de données Aurora dans la section Bases de données de la console, et affiche son état actuel. Votre clone est prêt à être utilisé quand son état est Disponible.

AWS CLI

L'utilisation de l'AWS CLI pour cloner votre cluster de bases de données Aurora implique des étapes distinctes pour créer le cluster du clone et y ajouter une ou plusieurs instances de base de données.

La commande `restore-db-cluster-to-point-in-time` que vous utilisez dans l'AWS CLI génère un cluster de bases de données Aurora contenant les mêmes données de stockage que le cluster d'origine, mais aucune instance de base de données Aurora. Vous créerez les instances de base de données séparément une fois que le clone sera disponible. Vous pouvez choisir le nombre d'instances de base de données et leurs classes d'instance pour donner au clone une capacité de calcul supérieure ou inférieure à celle du cluster d'origine. Les étapes de ce processus sont les suivantes :

1. Créez le clone à l'aide de la commande de la CLI restore-db-cluster-to-point-in-time.
2. Créez l'instance de base de données d'enregistreur pour le clone à l'aide de la commande create-db-instance de la CLI.

3. (Facultatif) Exécutez des commandes create-db-instance supplémentaires de la CLI pour ajouter une ou plusieurs instances de lecteur au cluster du clone. L'utilisation d'instances de lecteur permet d'améliorer les aspects du clone en matière de haute disponibilité et de capacité de mise à l'échelle en lecture. Vous pouvez ignorer cette étape si vous prévoyez d'utiliser le clone uniquement pour le développement et les tests.

Rubriques

- Création du clone
- Vérification de l'état et obtention des détails du clone
- Création de l'instance de base de données Aurora pour votre clone
- Paramètres à utiliser pour le clonage

Création du clone

Utilisez la commande `restore-db-cluster-to-point-in-time` de la CLI pour créer le cluster du clone initial.

Pour créer un clone à partir d'un cluster de bases de données Aurora source

- Utilisez la commande `restore-db-cluster-to-point-in-time` de la CLI. Spécifiez les valeurs des paramètres suivants. Dans ce cas typique, le clone utilise le même mode moteur que le cluster d'origine, qu'il soit provisionné ou Aurora Serverless v1.
  - `--db-cluster-identifier` – Choisissez un nom explicite pour votre clone. Vous nommez le clone lorsque vous utilisez la commande de la CLI restore-db-cluster-to-point-in-time. Vous passez ensuite le nom du clone dans la commande de la CLI create-db-instance.
  - `--restore-type` – Utilisez la commande `copy-on-write` pour créer un clone du cluster de bases de données source. Sans ce paramètre, la commande `restore-db-cluster-to-point-in-time` restaure le cluster de bases de données Aurora au lieu de créer un clone.
  - `--source-db-cluster-identifier` – Utilisez le nom du cluster de bases de données Aurora source que vous souhaitez cloner.
  - `--use-latest-restorable-time` : cette valeur pointe vers les données de volume restaurables les plus récentes pour le cluster de bases de données source. Utilisez-la pour créer des clones.

L'exemple suivant crée un clone nommé `my-clone` à partir d'un cluster nommé `my-source-cluster`.

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Pour Windows :

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

La commande renvoie l'objet JSON contenant les détails du clone. Vérifiez que votre cluster de bases de données cloné est disponible avant d'essayer de créer l'instance de base de données pour votre clone. Pour plus d'informations, consultez Vérification de l'état et obtention des détails du clone.

Par exemple, supposons que vous ayez un cluster nommé `tpch100g` que vous souhaitez cloner. L'exemple Linux suivant crée un cluster cloné nommé `tpch100g-clone`, une instance d'enregistreur Aurora Serverless v2 nommée `tpch100g-clone-instance` et une instance de lecteur provisionnée nommée `tpch100g-clone-instance-2` pour le nouveau cluster.

Vous n'avez pas besoin de fournir certains paramètres, tels que `--master-username` et `--master-user-password`. Aurora détermine automatiquement ceux du cluster d'origine. Vous devez spécifier le moteur de base de données à utiliser. Ainsi, l'exemple teste le nouveau cluster pour déterminer la bonne valeur à utiliser pour le paramètre `--engine`.

Cet exemple inclut également l'option `--serverless-v2-scaling-configuration` lors de la création du cluster du clone. Ainsi, vous pouvez ajouter des instances Aurora Serverless v2 au clone même si le cluster d'origine n'a pas utilisé Aurora Serverless v2.

```
$ aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier tpch100g \
  --db-cluster-identifier tpch100g-clone \
  --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16 \
```

```
  --restore-type copy-on-write \
  --use-latest-restorable-time

$ aws rds describe-db-clusters \
  --db-cluster-identifier tpch100g-clone \
    --query '*[].[Engine]' \
    --output text
aurora-mysql

$ aws rds create-db-instance \
  --db-instance-identifier tpch100g-clone-instance \
  --db-cluster-identifier tpch100g-clone \
  --db-instance-class db.serverless \
  --engine aurora-mysql

$ aws rds create-db-instance \
  --db-instance-identifier tpch100g-clone-instance-2 \
  --db-cluster-identifier tpch100g-clone \
  --db-instance-class db.r6g.2xlarge \
  --engine aurora-mysql
```

Pour créer un clone avec un mode moteur différent de celui du cluster de bases de données Aurora source

- Cette procédure s'applique uniquement aux anciennes versions du moteur compatibles avec Aurora Serverless v1. Supposons que vous disposiez d'un cluster Aurora Serverless v1 et que vous souhaitiez créer un clone qui soit provisionné. Dans ce cas, utilisez la commande `restore-db-cluster-to-point-in-time` de la CLI et spécifiez des valeurs de paramètres similaires à celles de l'exemple précédent, ainsi que les paramètres supplémentaires suivants :

  - `--engine-mode` : utilisez ce paramètre uniquement pour créer des clones dont le mode moteur est différent de celui du cluster de bases de données Aurora source. Ce paramètre s'applique uniquement aux anciennes versions du moteur compatibles avec Aurora Serverless v1. Choisissez la valeur à passer avec `--engine-mode` comme suit :

    - Utilisez `--engine-mode provisioned` pour créer un clone de cluster de bases de données Aurora approvisionné à partir d'un cluster de bases de données Aurora Serverless.

> **Note**
>
> Si vous avez prévoyez d'utiliser Aurora Serverless v2 avec un cluster cloné à partir d'Aurora Serverless v1, vous devez toujours spécifier le mode moteur du clone comme étant `provisioned`. Vous devez ensuite effectuer des étapes supplémentaires de mise à niveau et de migration.

  - Utilisez `--engine-mode serverless` pour créer un clone Aurora Serverless v1 à partir d'un cluster de bases de données Aurora provisionné. Lorsque vous spécifiez le mode moteur `serverless`, vous pouvez également choisir `--scaling-configuration`.
- `--scaling-configuration` : (facultatif) utilisez cette option avec `--engine-mode serverless` afin de configurer la capacité minimale et maximale d'un clone Aurora Serverless v1. Si vous n'utilisez pas ce paramètre, Aurora crée un clone Aurora Serverless v1 à l'aide des valeurs de capacité Aurora Serverless v1 par défaut pour le moteur de base de données.

L'exemple suivant crée un clone provisionné nommé `my-clone` à partir d'un cluster de bases de données Aurora Serverless v1 nommé `my-source-cluster`.

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-source-cluster \
    --db-cluster-identifier my-clone \
    --engine-mode provisioned \
    --restore-type copy-on-write \
    --use-latest-restorable-time
```

Pour Windows :

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-identifier my-source-cluster ^
    --db-cluster-identifier my-clone ^
    --engine-mode provisioned ^
    --restore-type copy-on-write ^
    --use-latest-restorable-time
```

Ces commandes renvoient l'objet JSON contenant les détails du clone dont vous avez besoin pour créer l'instance de base de données. Vous ne pouvez pas faire cela tant que l'état du clone (le cluster de bases de données Aurora vide) n'est pas Disponible.

> **Note**
>
> La commande CLI AWS restore-db-cluster-to-point-in-time restaure uniquement le cluster de bases de données, et non pas les instances de base de données pour ce cluster. Exécutez la commande create-db-instance pour créer des instances de base de données pour le cluster de bases de données restauré. Avec cette commande, vous spécifiez l'identifiant du cluster de bases de données restauré en tant que paramètre `--db-cluster-identifier`. Vous pouvez créer des instances de base de données uniquement après la fin de la commande `restore-db-cluster-to-point-in-time` et lorsque le cluster de bases de données est disponible.
>
> Supposons que vous commenciez par un cluster Aurora Serverless v1 et que vous ayez l'intention de le migrer vers un cluster Aurora Serverless v2. Vous créerez un clone provisionné du cluster Aurora Serverless v1 comme première étape de la migration. Pour la procédure complète, y compris les mises à niveau de version obligatoires, consultez Mise à niveau d'un cluster Aurora Serverless v1 vers Aurora Serverless v2.

Vérification de l'état et obtention des détails du clone

Vous pouvez utiliser la commande suivante pour vérifier l'état du cluster du clone que vous venez de créer.

```
$ aws rds describe-db-clusters --db-cluster-identifier my-clone --query '*[].[Status]'
 --output text
```

Ou vous pouvez obtenir l'état et les autres valeurs dont vous avez besoin pour créer l'instance de base de données pour votre clone en utilisant la requête de l'AWS CLI suivante.

Pour Linux, macOS ou Unix :

```
aws rds describe-db-clusters --db-cluster-identifier my-clone \
  --query '*[].
{Status:Status,Engine:Engine,EngineVersion:EngineVersion,EngineMode:EngineMode}'
```

Pour Windows :

```
aws rds describe-db-clusters --db-cluster-identifier my-clone ^
  --query "*[].
{Status:Status,Engine:Engine,EngineVersion:EngineVersion,EngineMode:EngineMode}"
```

Cette requête retourne une sortie similaire à la suivante.

```
[
  {
        "Status": "available",
        "Engine": "aurora-mysql",
        "EngineVersion": "8.0.mysql_aurora.3.04.1",
        "EngineMode": "provisioned"
    }
]
```

Création de l'instance de base de données Aurora pour votre clone

Utilisez la commande create-db-instance de la CLI pour créer l'instance de base de données de votre clone Aurora Serverless v2 ou provisionné. Vous ne devez pas créer d'instance de base de données pour un clone Aurora Serverless v1.

L'instance de base de données hérite des propriétés `--master-username` et `--master-user-password` du cluster de bases de données source.

L'exemple suivant créé une instance de base de données pour un clone provisionné.

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class db.r6g.2xlarge \
    --engine aurora-mysql
```

Pour Windows :

```
aws rds create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class db.r6g.2xlarge ^
    --engine aurora-mysql
```

L'exemple suivant crée une instance de base de données Aurora Serverless v2 pour un clone qui utilise une version de moteur compatible avec Aurora Serverless v2.

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance \
    --db-instance-identifier my-new-db \
    --db-cluster-identifier my-clone \
    --db-instance-class db.serverless \
  --engine aurora-postgresql
```

Pour Windows :

```
aws rds create-db-instance ^
    --db-instance-identifier my-new-db ^
    --db-cluster-identifier my-clone ^
    --db-instance-class db.serverless ^
    --engine aurora-mysql
```

Paramètres à utiliser pour le clonage

Le tableau suivant récapitule les différents paramètres utilisés avec la commande `restore-db-cluster-to-point-in-time` pour cloner des clusters de bases de données Aurora.

Paramètre	Description
`--source-db-cluster-identifier`	Utilisez le nom du cluster de bases de données Aurora source que vous souhaitez cloner.
`--db-cluster-identifier`	Choisissez un nom explicite pour votre clone lorsque vous le créez avec la commande `restore-db-cluster-to-point-in-time` . Ensuite, vous passez ce nom à la commande `create-db-instance` .
`--restore-type`	Spécifiez `copy-on-write` en tant que `--restore-type` pour créer un clone du cluster de bases de données source au lieu de restaurer le cluster de bases de données Aurora source.

Paramètre	Description
`--use-latest-restorable-time`	Cette valeur pointe vers les données de volume restaurables les plus récentes pour le cluster de bases de données source. Utilisez-la pour créer des clones.
`--serverless-v2-scaling-configuration`	(Versions récentes compatibles avec Aurora Serverless v2) Utilisez ce paramètre pour configurer la capacité minimale et maximale d'un clone Aurora Serverless v2. Si vous ne spécifiez pas ce paramètre, vous ne pourrez créer aucune instance Aurora Serverless v2 dans le cluster du clone tant que vous n'aurez pas modifié le cluster pour y ajouter cet attribut.
`--engine-mode`	(Anciennes versions compatibles avec Aurora Serverless v1 uniquement) Utilisez ce paramètre pour créer des clones d'un type différent de celui du cluster de bases de données Aurora source, avec l'une des valeurs suivantes : • Utilisez `provisioned` pour créer un clone provisionné à partir d'un cluster de bases de données Aurora Serverless v1. • Utilisez `serverless` pour créer un clone Aurora Serverless v1 à partir d'un cluster de bases de données Aurora Serverless v2 provisionné. Lorsque vous spécifiez le mode moteur `serverless`, vous pouvez également choisir `--scaling-configuration`.
`--scaling-configuration`	(Anciennes versions compatibles avec Aurora Serverless v1 uniquement) Utilisez ce paramètre pour configurer la capacité minimale et maximale d'un clone Aurora Serverless v1. Si vous ne spécifiez pas ce paramètre, Aurora crée le clone avec les valeurs de capacité par défaut correspondant au moteur de base de données.

Pour plus d'informations sur le clonage entre VPC et entre comptes, consultez les sections suivantes.

Rubriques

- Clonage entre VPC avec Amazon Aurora

- Clonage entre comptes avec Amazon Aurora AWS RAM et Amazon

Clonage entre VPC avec Amazon Aurora

Supposons que vous souhaitiez imposer des contrôles d'accès réseau différents au cluster d'origine et au clone. Par exemple, vous pouvez utiliser le clonage pour créer une copie d'un cluster Aurora de production dans un autre VPC à des fins de développement et de test. Vous pouvez également créer un clone dans le cadre d'une migration de sous-réseaux publics vers des sous-réseaux privés, afin d'améliorer la sécurité de votre base de données.

Les sections suivantes montrent comment définir la configuration réseau du clone afin que le cluster d'origine et le clone puissent accéder aux mêmes nœuds de stockage Aurora, même à partir de sous-réseaux ou de VPC différents. La vérification préalable des ressources du réseau permet d'éviter des erreurs difficiles à diagnostiquer lors du clonage.

Si vous ne connaissez pas la façon dont Aurora interagit avec les VPC, les sous-réseaux et les groupes de sous-réseaux de base de données, consultez d'abord Amazon VPC et Amazon Aurora. Vous pouvez suivre les didacticiels de cette section pour créer ce type de ressources dans la console AWS et comprendre comment elles s'intègrent.

Comme les étapes impliquent de basculer entre les services RDS et EC2, les exemples utilisent des commandes AWS de la CLI pour vous aider à comprendre comment automatiser ces opérations et enregistrer la sortie.

Rubriques

- Avant de commencer
- Collecte d'informations sur l'environnement réseau
- Création de ressources réseau pour le clone
- Création d'un clone Aurora avec de nouveaux paramètres réseau
- Remplacement des sous-réseaux publics par des sous-réseaux privés pour un cluster
- Exemple complet de création d'un clone entre VPC

Avant de commencer

Avant de commencer à configurer un clone entre VPC, assurez-vous d'avoir les ressources suivantes :

- Un cluster de bases de données Aurora à utiliser comme source pour le clonage. Si c'est la première fois que vous créez un cluster de bases de données Aurora, consultez les didacticiels sous Mise en route avec Amazon Aurora pour configurer un cluster à l'aide du moteur de base de données MySQL ou PostgreSQL.
- Un deuxième VPC, si vous avez l'intention de créer un clone entre VPC. Si vous n'avez pas de VPC à utiliser pour le clone, consultez Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement) ou Tutoriel : Créer un VPC à utiliser avec un cluster de bases de données (mode double-pile).

Collecte d'informations sur l'environnement réseau

Avec le clonage entre VPC, l'environnement réseau peut être très différent entre le cluster d'origine et son clone. Avant de créer le clone, collectez et enregistrez les informations relatives au VPC, aux sous-réseaux, au groupe de sous-réseaux de base de données et aux zones de disponibilité utilisés dans le cluster d'origine. De cette façon, vous minimiserez les risques de problèmes. En cas de problème réseau, vous n'aurez pas à interrompre les activités de dépannage pour rechercher des informations de diagnostic. Les sections suivantes présentent des exemples permettant de recueillir ce type d'informations à l'aide de la CLI. Vous pouvez enregistrer ces informations dans le format qui vous convient le mieux lors de la création du clone et de la résolution des problèmes.

- Étape 1 : vérifier les zones de disponibilité du cluster d'origine
- Étape 2 : vérifier le groupe de sous-réseaux de base de données du cluster d'origine
- Étape 3 : vérifier les sous-réseaux du cluster d'origine
- Étape 4 : vérifier les zones de disponibilité des instances de base de données dans le cluster d'origine
- Étape 5 : vérifier les VPC que vous pouvez utiliser pour le clone

Étape 1 : vérifier les zones de disponibilité du cluster d'origine

Avant de créer le clone, vérifiez les zones de disponibilité que le cluster d'origine utilise pour son stockage. Comme expliqué dans Stockage Amazon Aurora, le stockage de chaque cluster Aurora est associé à exactement trois zones de disponibilité. Dans la mesure où Clusters de bases de données Amazon Aurora tire parti de la séparation du calcul et du stockage, cette règle s'applique quel que soit le nombre d'instances présentes dans le cluster.

Par exemple, exécutez une commande d'interface de ligne de commande comme dans cet exemple, en remplaçant *my_cluster* par le nom de votre propre cluster. L'exemple suivant génère une liste de zones de disponibilité triée par ordre alphabétique.

```
aws rds describe-db-clusters \
  --db-cluster-identifier my_cluster \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' \
  --output text
```

L'exemple suivant montre un exemple de sortie de la commande `describe-db-clusters` précédente. Cela montre que le stockage du cluster Aurora utilise toujours trois zones de disponibilité.

```
us-east-1c
us-east-1d
us-east-1e
```

Pour créer un clone dans un environnement réseau ne disposant pas de toutes les ressources nécessaires pour se connecter à ces zones de disponibilité, vous devrez créer des sous-réseaux associés à au moins deux de ces zones, puis créer un groupe de sous-réseaux de base de données contenant ces deux ou trois sous-réseaux. Les exemples suivants montrent comment procéder.

Étape 2 : vérifier le groupe de sous-réseaux de base de données du cluster d'origine

Si vous souhaitez utiliser le même nombre de sous-réseaux pour le clone que dans le cluster d'origine, vous pouvez obtenir le nombre de sous-réseaux à partir du groupe de sous-réseaux de base de données du cluster d'origine. Un groupe de sous-réseaux de base de données Aurora contient au moins deux sous-réseaux, chacun étant associé à une zone de disponibilité différente. Notez à quelles zones de disponibilité les sous-réseaux sont associés.

L'exemple suivant montre comment trouver le groupe de sous-réseaux de base de données du cluster d'origine, puis comment remonter aux zones de disponibilité correspondants. Remplacez *my_cluster* par le nom de votre cluster dans la première commande. Remplacez *my_subnet* par le nom du groupe de sous-réseaux de base de données dans la deuxième commande.

```
aws rds describe-db-clusters --db-cluster-identifier my_cluster \
  --query '*[].DBSubnetGroup' --output text

aws rds describe-db-subnet-groups --db-subnet-group-name my_subnet_group \
  --query '*[].Subnets[].[SubnetAvailabilityZone.Name]' --output text
```

L'exemple de sortie peut ressembler à ce qui suit, pour un cluster avec un groupe de sous-réseaux de base de données contenant deux sous-réseaux. Dans ce cas, `two-subnets` est un nom qui a été spécifié lors de la création du groupe de sous-réseaux de base de données.

```
two-subnets

us-east-1d
us-east-1c
```

Pour un cluster où le groupe de sous-réseaux de base de données contient trois sous-réseaux, la sortie peut ressembler à ce qui suit.

```
three-subnets

us-east-1f
us-east-1d
us-east-1c
```

Étape 3 : vérifier les sous-réseaux du cluster d'origine

Si vous avez besoin de plus de détails sur les sous-réseaux du cluster d'origine, exécutez des commandes AWS similaires aux suivantes dans l'interface de ligne de commande. Vous pouvez examiner les attributs des sous-réseaux tels que les plages d'adresses IP, le propriétaire, etc. Vous pouvez ainsi déterminer s'il convient d'utiliser différents sous-réseaux dans le même VPC ou de créer des sous-réseaux présentant des caractéristiques similaires dans un VPC différent.

Identifiez les ID de tous les sous-réseaux disponibles dans votre VPC.

```
aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc \
  --query '*[].[SubnetId]' --output text
```

Identifiez les sous-réseaux exacts utilisés dans votre groupe de sous-réseaux de base de données.

```
aws rds describe-db-subnet-groups --db-subnet-group-name my_subnet_group \
  --query '*[].Subnets[].[SubnetIdentifier]' --output text
```

Spécifiez ensuite les sous-réseaux que vous souhaitez examiner dans une liste, comme dans la commande suivante. Remplacez *my_subnet_1* et ainsi de suite par le nom de vos sous-réseaux.

```
aws ec2 describe-subnets \
```

```
  --subnet-ids '["my_subnet_1","my_subnet2","my_subnet3"]'
```

L'exemple suivant montre une sortie partielle de ce type de commande `describe-subnets`. La sortie montre certains des attributs importants que vous pouvez voir pour chaque sous-réseau, tels que la zone de disponibilité associée et le VPC dont il fait partie.

```
{
    'Subnets': [
        {
            'AvailabilityZone': 'us-east-1d',
            'AvailableIpAddressCount': 54,
            'CidrBlock': '10.0.0.64/26',
            'State': 'available',
            'SubnetId': 'subnet-000a0bca00e0b0000',
            'VpcId': 'vpc-3f3c3fc3333b3ffb3',
            ...
        },
        {
            'AvailabilityZone': 'us-east-1c',
            'AvailableIpAddressCount': 55,
            'CidrBlock': '10.0.0.0/26',
            'State': 'available',
            'SubnetId': 'subnet-4b4dbfe4d4a4fd4c4',
            'VpcId': 'vpc-3f3c3fc3333b3ffb3',
            ...
```

Étape 4 : vérifier les zones de disponibilité des instances de base de données dans le cluster d'origine

Vous pouvez utiliser cette procédure pour déterminer les zones de disponibilité utilisées pour les instances de base de données du cluster d'origine. Vous pouvez ainsi configurer exactement les mêmes zones de disponibilité pour les instances de base de données du clone. Vous pouvez également utiliser plus ou moins d'instances de base de données dans le clone selon que le clone sera utilisé pour la production, le développement et les tests, etc.

Pour chaque instance du cluster d'origine, exécutez une commande comme la suivante. Assurez-vous d'abord que l'instance a terminé de se créer et que son état indique `Available`. Remplacez *`my_instance`* par l'identifiant de l'instance.

```
aws rds describe-db-instances --db-instance-identifier my_instance \
  --query '*[].AvailabilityZone' --output text
```

L’exemple suivant illustre la sortie de l’exécution de la commande `describe-db-instances` précédente. Le cluster Aurora possède quatre instances de base de données. Par conséquent, nous exécutons la commande quatre fois, en indiquant à chaque fois un identifiant d’instance de base de données différent. La sortie montre comment ces instances de base de données sont réparties sur un maximum de trois zones de disponibilité.

```
us-east-1a
us-east-1c
us-east-1d
us-east-1a
```

Une fois que le clone est créé et que vous y avez ajouté des instances de base de données, vous pouvez spécifier le nom de ces mêmes zones de disponibilité dans les commandes `create-db-instance`. Cela vous permet de configurer les instances de base de données dans le nouveau cluster exactement pour les mêmes zones de disponibilité que le cluster d’origine.

Étape 5 : vérifier les VPC que vous pouvez utiliser pour le clone

Si vous avez l’intention de créer le clone dans un VPC différent de celui d’origine, vous pouvez obtenir une liste des identifiants de VPC disponibles pour votre compte. Vous pouvez également effectuer cette étape si vous devez créer des sous-réseaux supplémentaires dans le même VPC que le cluster d’origine. Lorsque vous exécutez la commande permettant de créer un sous-réseau, vous spécifiez l’identifiant du VPC en tant que paramètre.

Pour dresser la liste de tous les VPC figurant dans votre compte, exécutez la commande suivante dans l’interface de ligne de commande :

```
aws ec2 describe-vpcs --query '*[].[VpcId]' --output text
```

L’exemple suivant montre un exemple de sortie de la commande `describe-vpcs` précédente. La sortie montre que le compte AWS actuel contient quatre VPC qui peuvent être utilisés comme source ou destination pour le clonage entre VPC.

```
vpc-fd111111
vpc-2222e2cd2a222f22e
vpc-33333333a33333d33
vpc-4ae4d4de4a4444dad
```

Vous pouvez utiliser le même VPC que la destination pour le clone, ou un VPC différent. Si le cluster d’origine et le clone se trouvent dans le même VPC, vous pouvez réutiliser le même groupe de sous-

réseaux de base de données pour le clone. Vous pouvez également créer un autre groupe de sous-réseaux de base de données. Par exemple, le nouveau groupe de sous-réseaux de base de données peut utiliser des sous-réseaux privés, tandis que le groupe de sous-réseaux de base de données du cluster d'origine peut utiliser des sous-réseaux publics. Si vous créez le clone dans un autre VPC, assurez-vous qu'il y a suffisamment de sous-réseaux dans le nouveau VPC et que les sous-réseaux sont associés aux zones de disponibilité appropriées du cluster d'origine.

Création de ressources réseau pour le clone

Si, lors de la collecte des informations réseau, vous avez découvert que des ressources réseau supplémentaires sont nécessaires pour le clone, vous pouvez les créer avant d'essayer de configurer le clone. Par exemple, vous devrez peut-être créer d'autres sous-réseaux, des sous-réseaux associés à des zones de disponibilité spécifiques ou un groupe de sous-réseaux de base de données.

- Étape 1 : créer les sous-réseaux pour le clone
- Étape 2 : créer le groupe de sous-réseaux de base de données pour le clone

Étape 1 : créer les sous-réseaux pour le clone

Si vous devez créer des sous-réseaux pour le clone, exécutez une commande similaire à ce qui suit. Vous devrez peut-être procéder de la sorte lors de la création du clone dans un autre VPC ou lors d'une autre modification du réseau, telle que l'utilisation de sous-réseaux privés au lieu de sous-réseaux publics.

AWS génère automatiquement l'identifiant du sous-réseau. Remplacez *my_vpc* par le nom du VPC du clone. Choisissez la plage d'adresses pour l'option `--cidr-block` pour y autoriser au moins 16 adresses IP. Vous pouvez inclure toutes les autres propriétés que vous souhaitez spécifier. Exécutez la commande `aws ec2 create-subnet help` pour voir tous les choix.

```
aws ec2 create-subnet --vpc-id my_vpc \
  --availability-zone AZ_name --cidr-block IP_range
```

L'exemple suivant montre certains attributs importants d'un sous-réseau nouvellement créé.

```
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1b',
        'AvailableIpAddressCount': 59,
        'CidrBlock': '10.0.0.64/26',
```

```
        'State': 'available',
        'SubnetId': 'subnet-44b4a44f4e44db444',
        'VpcId': 'vpc-555fc5df555e555dc',
        ...
        }
}
```

Étape 2 : créer le groupe de sous-réseaux de base de données pour le clone

Si vous créez le clone dans un autre VPC ou dans un ensemble différent de sous-réseaux au sein du même VPC, vous devez créer un nouveau groupe de sous-réseaux de base de données et le spécifier lors de la création du clone.

Veillez à connaître tous les détails suivants. Ils sont tous visibles dans la sortie des exemples précédents.

1. VPC du cluster d'origine. Pour obtenir des instructions, consultez Étape 3 : vérifier les sous-réseaux du cluster d'origine.
2. VPC du clone, si vous le créez dans un autre VPC. Pour obtenir des instructions, consultez Étape 5 : vérifier les VPC que vous pouvez utiliser pour le clone.
3. Trois zones de disponibilité associées au stockage Aurora pour le cluster d'origine. Pour obtenir des instructions, consultez Étape 1 : vérifier les zones de disponibilité du cluster d'origine.
4. Deux ou trois zones de disponibilité associées au groupe de sous-réseaux de base de données pour le cluster d'origine. Pour obtenir des instructions, consultez Étape 2 : vérifier le groupe de sous-réseaux de base de données du cluster d'origine.
5. ID de sous-réseau et zones de disponibilité associées de tous les sous-réseaux du VPC que vous souhaitez utiliser pour le clone. Utilisez la même commande `describe-subnets` que dans Étape 3 : vérifier les sous-réseaux du cluster d'origine, en remplaçant l'ID du VPC de destination.

Vérifiez combien de zones de disponibilité sont à la fois associées au stockage du cluster d'origine et aux sous-réseaux du VPC de destination. Pour que le clone soit créé, deux ou trois zones de disponibilité en commun sont nécessaires. Si vous avez moins de deux zones de disponibilité en commun, revenez à Étape 1 : créer les sous-réseaux pour le clone. Créez un, deux ou trois sous-réseaux liés aux zones de disponibilité associés au stockage du cluster d'origine.

Choisissez dans le VPC de destination des sous-réseaux associés aux mêmes zones de disponibilité que le stockage Aurora dans le cluster d'origine. Idéalement, choisissez trois zones de disponibilité. Cela vous laissera la liberté de répartir les instances de base de données du clone sur plusieurs zones de disponibilité pour assurer une haute disponibilité des ressources de calcul.

Exécutez une commande similaire à ce qui suit afin de créer le groupe de sous-réseaux de base de données. Remplacez les identifiants de vos sous-réseaux dans la liste. Si vous spécifiez les identifiants de sous-réseau à l'aide de variables d'environnement, veillez à citer la liste des paramètres `--subnet-ids` de manière à conserver les guillemets doubles autour des identifiants.

```
aws rds create-db-subnet-group --db-subnet-group-name my_subnet_group \
  --subnet-ids '["my_subnet_1","my_subnet_2","my_subnet3"]' \
  --db-subnet-group-description 'DB subnet group with 3 subnets for clone'
```

L'exemple suivant illustre la sortie partielle de la commande `create-db-subnet-group`.

```
{
    'DBSubnetGroup': {
        'DBSubnetGroupName': 'my_subnet_group',
        'DBSubnetGroupDescription': 'DB subnet group with 3 subnets for clone',
        'VpcId': 'vpc-555fc5df555e555dc',
        'SubnetGroupStatus': 'Complete',
        'Subnets': [
            {
                'SubnetIdentifier': 'my_subnet_1',
                'SubnetAvailabilityZone': {
                    'Name': 'us-east-1c'
                },
                'SubnetStatus': 'Active'
            },
            {
                'SubnetIdentifier': 'my_subnet_2',
                'SubnetAvailabilityZone': {
                    'Name': 'us-east-1d'
                },
                'SubnetStatus': 'Active'
            }
            ...
        ],
        'SupportedNetworkTypes': [
            'IPV4'
        ]
    }
}
```

À ce stade, vous n'avez pas encore créé le clone. Vous avez créé toutes les ressources de VPC et de sous-réseau pertinentes afin de pouvoir spécifier les paramètres appropriés pour les commandes

`restore-db-cluster-to-point-in-time` et `create-db-instance` lors de la création du clone.

Création d'un clone Aurora avec de nouveaux paramètres réseau

Une fois que vous vous êtes assuré que vous avez bien configuré les VPC, les sous-réseaux, les zones de disponibilité et les groupes de sous-réseaux pour le nouveau cluster, vous pouvez effectuer l'opération de clonage proprement dite. Les exemples suivants de la CLI mettent en évidence les options telles que `--db-subnet-group-name`, `--availability-zone` et `--vpc-security-group-ids` que vous spécifiez dans les commandes pour configurer le clone et ses instances de base de données.

Étape 1 : spécifier le groupe de sous-réseaux de base de données pour le clone

Lorsque vous créez le clone, vous pouvez configurer tous les paramètres appropriés pour les VPC, les sous-réseaux et les zones de disponibilité en spécifiant un groupe de sous-réseaux de base de données. Utilisez les commandes des exemples précédents pour vérifier toutes les relations et tous les mappages inhérents au groupe de sous-réseaux de base de données.

Par exemple, les commandes suivantes illustrent le clonage d'un cluster d'origine vers un clone. Dans le premier exemple, le cluster source est associé à deux sous-réseaux et le clone est associé à trois sous-réseaux. Le deuxième exemple montre le cas contraire, à savoir le clonage d'un cluster de trois sous-réseaux vers un cluster de deux sous-réseaux.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier cluster-with-3-subnets \
  --db-cluster-identifier cluster-cloned-to-2-subnets \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name two-subnets
```

Si vous avez l'intention d'utiliser des instances Aurora sans serveur v2 dans le clone, incluez une option `--serverless-v2-scaling-configuration` lors de la création du clone, comme indiqué. Cela vous permet d'utiliser la classe `db.serverless` lors de la création d'instances de base de données dans le clone. Vous pourrez également modifier le clone ultérieurement pour ajouter cet attribut de configuration de mise à l'échelle. Les valeurs de capacité indiquées dans cet

exemple permettent à chaque instance sans serveur v2 du cluster de se mettre à l'échelle avec des unités de capacité Aurora (ACU) comprises entre 2 et 32. Pour en savoir plus sur la fonctionnalité Aurora sans serveur v2 et sur le choix de la plage de capacité, consultez Utiliser Aurora Serverless v2.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier cluster-with-2-subnets \
  --db-cluster-identifier cluster-cloned-to-3-subnets \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name three-subnets \
  --serverless-v2-scaling-configuration 'MinCapacity=2,MaxCapacity=32'
```

Quel que soit le nombre de sous-réseaux utilisés par les instances de base de données, le stockage Aurora pour le cluster source et le clone est associé à trois zones de disponibilité. L'exemple suivant répertorie les zones de disponibilité associées au cluster d'origine et au clone, pour les deux commandes `restore-db-cluster-to-point-in-time` des exemples précédents.

```
aws rds describe-db-clusters --db-cluster-identifier cluster-with-3-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f

aws rds describe-db-clusters --db-cluster-identifier cluster-cloned-to-2-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f

aws rds describe-db-clusters --db-cluster-identifier cluster-with-2-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1a
us-east-1c
us-east-1d

aws rds describe-db-clusters --db-cluster-identifier cluster-cloned-to-3-subnets \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1a
```

```
us-east-1c
us-east-1d
```

Comme au moins deux des zones de disponibilité sont les mêmes entre chaque paire de clusters d'origine et de clone, les deux clusters peuvent accéder au même stockage Aurora sous-jacent.

Étape 2 : spécifier les paramètres réseau des instances du clone

Lorsque vous créez des instances de base de données dans le clone, elles héritent par défaut du groupe de sous-réseaux de base de données du cluster lui-même. De cette façon, Aurora assigne automatiquement chaque instance à un sous-réseau particulier et la crée dans la zone de disponibilité associée au sous-réseau. Ce choix est pratique, en particulier pour les systèmes de développement et de test, car vous n'avez pas à suivre les identifiants de sous-réseau ni les zones de disponibilité lors de l'ajout de nouvelles instances au clone.

Vous pouvez également spécifier la zone de disponibilité lorsque vous créez une instance de base de données Aurora pour le clone. La zone de disponibilité que vous spécifiez doit faire partie de l'ensemble de zones de disponibilité associées au clone. Si le groupe de sous-réseaux de base de données que vous utilisez pour le clone ne contient que deux sous-réseaux, vous ne pouvez choisir que parmi les zones de disponibilité associées à ces deux sous-réseaux. Ce choix offre flexibilité et résilience pour des systèmes à haute disponibilité, car vous pouvez vous assurer que l'instance d'enregistreur et l'instance de lecteur de secours se trouvent dans des zones de disponibilité différentes. Ou si vous ajoutez des lecteurs supplémentaires au cluster, vous pouvez vous assurer qu'ils sont répartis sur trois zones de disponibilité. Ainsi, même dans les rares cas d'échec d'une zone de disponibilité, vous disposez toujours d'une instance d'enregistreur et d'une autre instance de lecteur dans deux autres zones de disponibilité.

L'exemple suivant ajoute une instance de base de données provisionnée à un cluster Aurora PostgreSQL cloné qui utilise un groupe de sous-réseaux de base de données personnalisé.

```
aws rds create-db-instance --db-cluster-identifier my_aurora_postgresql_clone \
  --db-instance-identifier my_postgres_instance \
  --db-subnet-group-name my_new_subnet \
  --engine aurora-postgresql \
  --db-instance-class db.t4g.medium
```

L'exemple suivant montre une sortie partielle de ce type de commande.

```
{
  'DBInstanceIdentifier': 'my_postgres_instance',
```

```
  'DBClusterIdentifier': 'my_aurora_postgresql_clone',
  'DBInstanceClass': 'db.t4g.medium',
  'DBInstanceStatus': 'creating'
  ...
}
```

L'exemple suivant ajoute une instance de base de données Aurora sans serveur v2 à un clone Aurora MySQL qui utilise un groupe de sous-réseaux de base de données personnalisé. Pour pouvoir utiliser les instances sans serveur v2, assurez-vous de spécifier l'option `--serverless-v2-scaling-configuration` pour la commande `restore-db-cluster-to-point-in-time`, comme indiqué dans les exemples précédents.

```
aws rds create-db-instance --db-cluster-identifier my_aurora_mysql_clone \
  --db-instance-identifier my_mysql_instance \
  --db-subnet-group-name my_other_new_subnet \
  --engine aurora-mysql \
  --db-instance-class db.serverless
```

L'exemple suivant montre une sortie partielle de ce type de commande.

```
{
  'DBInstanceIdentifier': 'my_mysql_instance',
  'DBClusterIdentifier': 'my_aurora_mysql_clone',
  'DBInstanceClass': 'db.serverless',
  'DBInstanceStatus': 'creating'
  ...
}
```

Étape 3 : établir la connectivité entre un système client et un clone

Si vous vous connectez déjà à un cluster Aurora à partir d'un système client, vous souhaiterez peut-être autoriser le même type de connectivité à un nouveau clone. Par exemple, vous pouvez vous connecter au cluster d'origine à partir d'une instance Amazon Cloud9 ou d'une instance EC2. Pour autoriser les connexions à partir des mêmes systèmes clients ou de nouveaux systèmes que vous créerez dans le VPC de destination, configurez des groupes de sous-réseaux de base de données et des groupes de sécurité VPC équivalents à ceux du VPC. Spécifiez ensuite le groupe de sous-réseaux et les groupes de sécurité lorsque vous créez le clone.

Les exemples suivants configurent un clone Aurora sans serveur v2. Cette configuration est basée sur la combinaison de `--engine-mode provisioned` et `--serverless-v2-scaling-`

`configuration` lors de la création du cluster de bases de données, puis `--db-instance-class db.serverless` lors de la création de chaque instance de base de données du cluster. Le mode moteur `provisioned` étant le mode par défaut, vous pouvez omettre cette option si vous le souhaitez.

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier serverless-sql-postgres\
  --db-cluster-identifier serverless-sql-postgres-clone \
  --db-subnet-group-name 'default-vpc-1234' \
  --vpc-security-group-ids 'sg-4567' \
  --serverless-v2-scaling-configuration 'MinCapacity=0.5,MaxCapacity=16' \
  --restore-type copy-on-write \
  --use-latest-restorable-time
```

Ensuite, lors de la création des instances de base de données dans le clone, spécifiez la même option `--db-subnet-group-name`. Vous pouvez éventuellement inclure l'option `--availability-zone` et spécifier l'une des zones de disponibilité associées aux sous-réseaux de ce groupe de sous-réseaux. Cette zone de disponibilité doit également être l'une des zones de disponibilité associées au cluster d'origine.

```
aws rds create-db-instance \
  --db-cluster-identifier serverless-sql-postgres-clone \
  --db-instance-identifier serverless-sql-postgres-clone-instance \
  --db-instance-class db.serverless \
  --db-subnet-group-name 'default-vpc-987zyx654' \
  --availability-zone 'us-east-1c' \
  --engine aurora-postgresql
```

Remplacement des sous-réseaux publics par des sous-réseaux privés pour un cluster

Vous pouvez utiliser le clonage pour remplacer les sous-réseaux publics d'un cluster par des sous-réseaux privés. Cette opération peut être utile lorsque vous ajoutez des couches de sécurité supplémentaires à votre application avant de la déployer en production. Pour cet exemple, les sous-réseaux privés et la passerelle NAT devraient déjà être configurés avant que vous ne démarriez le processus de clonage avec Aurora.

Pour les étapes impliquant Aurora, vous pouvez suivre les mêmes étapes générales que dans les exemples précédents pour Collecte d'informations sur l'environnement réseau et Création d'un clone Aurora avec de nouveaux paramètres réseau. La principale différence est que même si vous avez des sous-réseaux publics mappés à toutes les zones de disponibilité du cluster d'origine, vous devez

maintenant vérifier que vous disposez de suffisamment de sous-réseaux privés pour un cluster Aurora et que ces sous-réseaux sont associés aux mêmes zones de disponibilité que celles utilisées pour le stockage Aurora dans le cluster d'origine. Comme dans d'autres cas d'utilisation du clonage, vous pouvez créer le groupe de sous-réseaux de base de données pour le clone avec trois ou deux sous-réseaux privés associés aux zones de disponibilité requises. Toutefois, si vous utilisez deux sous-réseaux privés dans le groupe de sous-réseaux de base de données, vous devrez disposer d'un troisième sous-réseau privé associé à la troisième zone de disponibilité utilisée pour le stockage Aurora dans le cluster d'origine.

Vous pouvez consulter cette liste de contrôle pour vérifier que toutes les exigences sont satisfaites pour effectuer ce type d'opération de clonage.

- Enregistrez les trois zones de disponibilité associées au cluster d'origine. Pour obtenir des instructions, consultez Étape 1 : vérifier les zones de disponibilité du cluster d'origine.
- Enregistrez les trois ou deux zones de disponibilité associées aux sous-réseaux publics dans le groupe de sous-réseaux de base de données du cluster d'origine. Pour obtenir des instructions, consultez Étape 3 : vérifier les sous-réseaux du cluster d'origine.
- Créez des sous-réseaux privés mappés aux trois zones de disponibilité associées au cluster d'origine. Effectuez également toute autre configuration réseau, telle que la création d'une passerelle NAT, pour pouvoir communiquer avec les sous-réseaux privés. Pour obtenir des instructions détaillées, consultez Création d'un sous-réseau dans le Guide de l'utilisateur Amazon Virtual Private Cloud.
- Créez un groupe de sous-réseaux de base de données contenant trois ou deux des sous-réseaux privés associés aux zones de disponibilité du premier point. Pour obtenir des instructions, consultez Étape 2 : créer le groupe de sous-réseaux de base de données pour le clone.

Lorsque toutes les conditions requises sont réunies, vous pouvez suspendre l'activité de la base de données sur le cluster d'origine pendant que vous créez le clone et que vous changez d'application pour l'utiliser. Une fois que le clone est créé et que vous avez vérifié que vous pouvez vous y connecter, exécuter le code de votre application, etc., vous pouvez cesser d'utiliser le cluster d'origine.

Exemple complet de création d'un clone entre VPC

La création d'un clone dans un VPC différent de celui d'origine suit les mêmes étapes générales que dans les exemples précédents. L'ID du VPC étant une propriété des sous-réseaux, vous ne spécifiez pas cet ID en tant que paramètre lorsque vous exécutez l'une des commandes de la CLI RDS. La principale différence est que vous aurez probablement besoin de créer des sous-réseaux, des sous-

réseaux mappés à des zones de disponibilité spécifiques, un groupe de sécurité VPC et un groupe de sous-réseaux de base de données. Cela est particulièrement vrai s'il s'agit du premier cluster Aurora que vous créez dans ce VPC.

Vous pouvez consulter cette liste de contrôle pour vérifier que toutes les exigences sont satisfaites pour effectuer ce type d'opération de clonage.

- Enregistrez les trois zones de disponibilité associées au cluster d'origine. Pour obtenir des instructions, consultez Étape 1 : vérifier les zones de disponibilité du cluster d'origine.
- Enregistrez les trois ou deux zones de disponibilité associées aux sous-réseaux dans le groupe de sous-réseaux de base de données du cluster d'origine. Pour obtenir des instructions, consultez Étape 2 : vérifier le groupe de sous-réseaux de base de données du cluster d'origine.
- Créez des sous-réseaux mappés aux trois zones de disponibilité associées au cluster d'origine. Pour obtenir des instructions, consultez Étape 1 : créer les sous-réseaux pour le clone.
- Effectuez toute autre configuration réseau, telle que la configuration d'un groupe de sécurité VPC, pour les systèmes clients, les serveurs d'applications, etc. afin de pouvoir communiquer avec les instances de base de données du clone. Pour obtenir des instructions, consultez Contrôle d'accès par groupe de sécurité.
- Créez un groupe de sous-réseaux de base de données contenant trois ou deux des sous-réseaux associés aux zones de disponibilité du premier point. Pour obtenir des instructions, consultez Étape 2 : créer le groupe de sous-réseaux de base de données pour le clone.

Lorsque toutes les conditions requises sont réunies, vous pouvez suspendre l'activité de la base de données sur le cluster d'origine pendant que vous créez le clone et que vous changez d'application pour l'utiliser. Une fois que le clone est créé et que vous avez vérifié que vous pouvez vous y connecter, exécuter le code de votre application, etc., vous pouvez décider de continuer à exécuter le cluster d'origine ainsi que les clones ou de cesser d'utiliser le cluster d'origine.

Les exemples Linux suivants montrent la séquence d'opérations de l'AWS CLI permettant de cloner un cluster de bases de données Aurora d'un VPC à un autre. Certains champs qui ne sont pas pertinents pour ces exemples ne sont pas affichés dans la sortie de la commande.

Tout d'abord, vérifions les identifiants des VPC source et de destination. Le nom descriptif que vous attribuez à un VPC lorsque vous le créez est représenté sous forme de balise dans les métadonnées du VPC.

```
$ aws ec2 describe-vpcs --query '*[].[VpcId,Tags]'
[
```

```
    [
        'vpc-0f0c0fc0000b0ffb0',
        [
            {
                'Key': 'Name',
                'Value': 'clone-vpc-source'
            }
        ]
    ],
    [
        'vpc-9e99d9f99a999bd99',
        [
            {
                'Key': 'Name',
                'Value': 'clone-vpc-dest'
            }
        ]
    ]
]
```

Le cluster d'origine existe déjà dans le VPC source. Pour configurer le clone en utilisant le même ensemble de zones de disponibilité pour le stockage Aurora, nous devons vérifier les zones de disponibilité utilisées par le cluster d'origine.

```
$ aws rds describe-db-clusters --db-cluster-identifier original-cluster \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f
```

Nous nous assurons que certains sous-réseaux correspondent aux zones de disponibilité utilisées par le cluster d'origine : `us-east-1c`, `us-east-1d` et `us-east-1f`.

```
$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
  --availability-zone us-east-1c --cidr-block 10.0.0.128/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1c',
        'SubnetId': 'subnet-3333a33be3ef3e333',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
```

```
}

$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
--availability-zone us-east-1d --cidr-block 10.0.0.160/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1d',
        'SubnetId': 'subnet-4eeb444cd44b4d444',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}

$ aws ec2 create-subnet --vpc-id vpc-9e99d9f99a999bd99 \
--availability-zone us-east-1f --cidr-block 10.0.0.224/28
{
    'Subnet': {
        'AvailabilityZone': 'us-east-1f',
        'SubnetId': 'subnet-66eea6666fb66d66c',
        'VpcId': 'vpc-9e99d9f99a999bd99',
    }
}
```

Cet exemple confirme que des sous-réseaux sont mappés aux zones de disponibilité nécessaires dans le VPC de destination.

```
aws ec2 describe-subnets --query 'sort_by(*[] | [?VpcId == `vpc-9e99d9f99a999bd99`] |
[].{SubnetId:SubnetId,VpcId:VpcId,AvailabilityZone:AvailabilityZone},
 &AvailabilityZone)' --output table

-------------------------------------------------------------------------
|                           DescribeSubnets                             |
+------------------+---------------------------+-------------------------+
| AvailabilityZone |         SubnetId          |          VpcId          |
+------------------+---------------------------+-------------------------+
|  us-east-1a      |  subnet-000ff0e00000c0aea  |  vpc-9e99d9f99a999bd99  |
|  us-east-1b      |  subnet-1111d111111ca11b1  |  vpc-9e99d9f99a999bd99  |
|  us-east-1c      | subnet-3333a33be3ef3e333   |  vpc-9e99d9f99a999bd99  |
|  us-east-1d      | subnet-4eeb444cd44b4d444   |  vpc-9e99d9f99a999bd99  |
|  us-east-1f      | subnet-66eea6666fb66d66c   |  vpc-9e99d9f99a999bd99  |
+------------------+---------------------------+-------------------------+
```

Avant de créer un cluster de bases de données Aurora dans le VPC, vous devez disposer d'un groupe de sous-réseaux de base de données avec des sous-réseaux mappés aux zones de

disponibilité utilisées pour le stockage Aurora. Lorsque vous créez un cluster standard, vous pouvez utiliser n'importe quel ensemble de trois zones de disponibilité. Lorsque vous clonez un cluster, le groupe de sous-réseaux doit correspondre à au moins deux des trois zones de disponibilité qu'il utilise pour le stockage Aurora.

```
$ aws rds create-db-subnet-group \
  --db-subnet-group-name subnet-group-in-other-vpc \
  --subnet-ids
 '["subnet-3333a33be3ef3e333","subnet-4eeb444cd44b4d444","subnet-66eea6666fb66d66c"]' \
  --db-subnet-group-description 'DB subnet group with 3 subnets:
 subnet-3333a33be3ef3e333,subnet-4eeb444cd44b4d444,subnet-66eea6666fb66d66c'

{
    'DBSubnetGroup': {
        'DBSubnetGroupName': 'subnet-group-in-other-vpc',
        'DBSubnetGroupDescription': 'DB subnet group with 3 subnets:
 subnet-3333a33be3ef3e333,subnet-4eeb444cd44b4d444,subnet-66eea6666fb66d66c',
        'VpcId': 'vpc-9e99d9f99a999bd99',
        'SubnetGroupStatus': 'Complete',
        'Subnets': [
            {
                'SubnetIdentifier': 'subnet-4eeb444cd44b4d444',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1d' }
            },
            {
                'SubnetIdentifier': 'subnet-3333a33be3ef3e333',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1c' }
            },
            {
                'SubnetIdentifier': 'subnet-66eea6666fb66d66c',
                'SubnetAvailabilityZone': { 'Name': 'us-east-1f' }
            }
        ]
    }
}
```

Les sous-réseaux et le groupe de sous-réseaux de base de données sont maintenant en place. L'exemple suivant montre l'opération `restore-db-cluster-to-point-in-time` qui clone le cluster. L'option `--db-subnet-group-name` associe le clone à l'ensemble approprié de sous-réseaux mappés à l'ensemble approprié de zones de disponibilité du cluster d'origine.

```
$ aws rds restore-db-cluster-to-point-in-time \
```

```
  --source-db-cluster-identifier original-cluster \
  --db-cluster-identifier clone-in-other-vpc \
  --restore-type copy-on-write --use-latest-restorable-time \
  --db-subnet-group-name subnet-group-in-other-vpc

{
  'DBClusterIdentifier': 'clone-in-other-vpc',
  'DBSubnetGroup': 'subnet-group-in-other-vpc',
  'Engine': 'aurora-postgresql',
  'EngineVersion': '15.4',
  'Status': 'creating',
  'Endpoint': 'clone-in-other-vpc.cluster-c0abcdef.us-east-1.rds.amazonaws.com'
}
```

L'exemple suivant confirme que le stockage Aurora du clone utilise le même ensemble de zones de disponibilité que dans le cluster d'origine.

```
$ aws rds describe-db-clusters --db-cluster-identifier clone-in-other-vpc \
  --query 'sort_by(*[].AvailabilityZones[].{Zone:@},&Zone)' --output text

us-east-1c
us-east-1d
us-east-1f
```

À ce stade, vous pouvez créer des instances de base de données pour le clone. Assurez-vous que le groupe de sécurité VPC associé à chaque instance autorise les connexions à partir des plages d'adresses IP que vous utilisez pour les instances EC2, les serveurs d'applications, etc. qui se trouvent dans le VPC de destination.

Clonage entre comptes avec Amazon Aurora AWS RAM et Amazon

En utilisant AWS Resource Access Manager (AWS RAM) avec Amazon Aurora, vous pouvez partager des clusters et des clones de base de données Aurora appartenant à votre AWS compte avec un autre AWS compte ou une autre organisation. Un tel clonage entre comptes est beaucoup plus rapide que la création et la restauration d'un instantané de base de données. Vous pouvez créer un clone d'un de vos clusters de bases de données Aurora et partager le clone. Vous pouvez également partager votre cluster de base de données Aurora avec un autre AWS compte et laisser le titulaire du compte créer le clone. L'approche que vous choisissez dépend de votre cas d'utilisation.

Par exemple, il se peut que vous deviez partager régulièrement un clone de votre base de données financière avec l'équipe d'audit interne de votre organisation. Dans ce cas, votre équipe d'audit

dispose de son propre compte AWS pour les applications qu'elle utilise. Vous pouvez autoriser le AWS compte de l'équipe d'audit à accéder à votre cluster de base de données Aurora et à le cloner selon les besoins.

D'un autre côté, si un fournisseur externe audite vos données financières, vous pouvez créer vous-même le clone. Vous donnez ensuite au fournisseur externe l'accès au clone uniquement.

Vous pouvez également utiliser le clonage entre comptes pour prendre en charge de nombreux cas d'utilisation identiques pour le clonage au sein d'un même AWS compte, tels que le développement et les tests. Par exemple, votre organisation peut utiliser différents AWS comptes pour la production, le développement, les tests, etc. Pour de plus amples informations, veuillez consulter Présentation du clonage Aurora.

Ainsi, vous souhaiterez peut-être partager un clone avec un autre AWS compte ou autoriser un autre AWS compte à créer des clones de vos clusters de base de données Aurora. Dans les deux cas, commencez par utiliser AWS RAM pour créer un objet de partage. Pour obtenir des informations complètes sur le partage de AWS ressources entre AWS comptes, consultez le guide de AWS RAM l'utilisateur.

La création d'un clone entre comptes nécessite des actions de la part du AWS compte propriétaire du cluster d'origine et du AWS compte qui crée le clone. D'abord, le propriétaire du cluster d'origine modifie le cluster pour autoriser un ou plusieurs autres comptes à le cloner. Si l'un des comptes appartient à une autre AWS organisation, AWS génère une invitation de partage. L'autre compte doit accepter l'invitation avant de continuer. Ensuite, chaque compte autorisé peut cloner le cluster. À travers ce processus, le cluster est identifié par son unique ARN.

Comme pour le clonage au sein du même AWS compte, l'espace de stockage supplémentaire n'est utilisé que si des modifications sont apportées aux données par la source ou le clone. Des frais de stockage sont alors appliqués. Si le cluster source est supprimé, les coûts de stockage sont répartis également entre les clusters clonés restants.

Rubriques

Limites du clonage intercompte

Le clonage intercompte Aurora présente les limitations suivantes :

- Vous ne pouvez pas cloner un Aurora Serverless v1 cluster sur plusieurs AWS comptes.
- Vous ne pouvez ni afficher ni accepter d'invitations à des ressources partagées avec leAWS Management Console. Utilisez l'AWS CLIAPI Amazon RDS ou la AWS RAM console pour consulter et accepter des invitations à des ressources partagées.
- Vous ne pouvez créer qu'un seul nouveau clone à partir d'une ressource partagée avec votreCompte AWS. Cela s’applique que la ressource partagée soit un cluster de bases de données Aurora original ou un clone créé précédemment.
- Vous ne pouvez créer qu'un seul nouveau clone à partir d'un clone partagé avec votre AWS compte.
- Vous ne pouvez pas partager les ressources (clones ou clusters de bases de données Aurora) qui ont été partagées avec votre AWS compte.
- Vous pouvez créer un maximum de 15 clones entre comptes à partir d’un seul cluster de bases de données Aurora.
- Chacun des 15 clones entre comptes doit appartenir à un compte différentAWS. En d'autres termes, vous ne pouvez créer qu'un seul clone multicompte d'un cluster au sein d'un AWS compte.
- Après avoir cloné un cluster, le cluster d’origine et son clone sont considérés comme étant les mêmes dans le but d’appliquer les limites sur les clones entre comptes. Vous ne pouvez pas créer de clones entre comptes à la fois du cluster d'origine et du cluster cloné au sein du même compte. AWS Le nombre total de clones entre comptes pour le cluster original et n’importe lequel de ses clones ne peut pas dépasser 15.
- Vous ne pouvez pas partager un cluster de base de données Aurora avec d'autres AWS comptes, sauf si le cluster est dans un `ACTIVE` état.
- Vous ne pouvez pas renommer un cluster de base de données Aurora qui a été partagé avec d'autres AWS comptes.
- Vous ne pouvez pas créer le clone intercompte d’un cluster chiffré avec la clé RDS par défaut.
- Vous ne pouvez pas créer de clones non chiffrés dans un AWS compte à partir de clusters de base de données Aurora chiffrés partagés par un autre AWS compte. Le propriétaire du cluster doit accorder l’autorisation d’accéder à la AWS KMS key du cluster source. Toutefois, vous pouvez utiliser une autre clé lorsque vous créez le clone.

Autoriser d'autres AWS comptes à cloner votre cluster

Pour autoriser d'autres AWS comptes à cloner un cluster dont vous êtes propriétaire, utilisez AWS RAM pour définir l'autorisation de partage. Cela envoie également une invitation à chacun des autres comptes appartenant à une AWS organisation différente.

Pour connaître les procédures de partage des ressources vous appartenant dans la AWS RAM console, consultez la section Partage des ressources vous appartenant dans le Guide de AWS RAM l'utilisateur.

Rubriques

Autoriser d'autres AWS comptes à cloner votre cluster

Si le cluster que vous partagez est chiffré, vous partagez également la AWS KMS key pour le cluster. Vous pouvez autoriser les utilisateurs ou les rôles Gestion des identités et des accès AWS (IAM) d'un AWS compte à utiliser une clé KMS dans un autre compte.

Pour ce faire, vous devez d'abord ajouter le compte externe (utilisateur root) à la politique clé de la clé KMS viaAWS KMS. Vous n’ajoutez pas les utilisateurs ou les rôles à la politique de clé, mais seulement le compte externe qui les possède. Vous ne pouvez partager qu’une clé KMS que vous créez, pas la clé de service RDS par défaut. Pour plus d’informations sur le contrôle d’accès pour les clés KMS, consultez Authentification et contrôle d’accès pour AWS KMS.

Console

Pour accorder l’autorisation de cloner votre cluster

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster de bases de données que vous voulez partager pour afficher sa page Details (Détails) et choisissez l’onglet Connectivity & security (Connexion e sécurité.
4. Dans la section Partager le cluster de base de données avec d'autres AWS comptes, entrez l'ID de compte numérique du AWS compte que vous souhaitez autoriser à cloner ce cluster. Pour

un compte IDs appartenant à la même organisation, vous pouvez commencer à taper dans le champ, puis choisir dans le menu.

> **Important**
>
> Dans certains cas, il se peut que vous souhaitiez qu'un compte qui n'est pas dans la même organisation AWS que votre compte puisse cloner un cluster. Dans ces cas-là, pour des raisons de sécurité, la console ne signale pas qui est le propriétaire de l'ID de compte ou si le compte existe.
> Soyez prudent lorsque vous saisissez des numéros de compte qui ne font pas partie de la même AWS organisation que votre AWS compte. Vérifiez immédiatement que vous avez partagé avec le compte prévu.

5. Sur la page de confirmation, vérifiez que l'ID de compte spécifié est correct. Entrez `share` dans la zone de confirmation pour confirmer.

   Sur la page Détails, une entrée apparaît qui indique l'ID de AWS compte spécifié sous Comptes avec lesquels ce cluster de base de données est partagé. La colone Status (État) affiche initialement l'état Pending.

6. Contactez le propriétaire de l'autre AWS compte ou connectez-vous à ce compte si vous possédez les deux. Demandez au propriétaire de l'autre compte d'accepter l'invitation de partage et clonez le cluster de bases de données, comme décrit ci-après.

AWS CLI

Pour accorder l'autorisation de cloner votre cluster

1. Collectez les informations pour les paramètres requis. Vous avez besoin de l'ARN de votre cluster et de l'identifiant numérique de l'autre AWS compte.
2. Exécutez la commande AWS RAM CLI `create-resource-share`.

   Pour Linux, macOS ou Unix :

```
aws ram create-resource-share --name descriptive_name \
  --region region \
  --resource-arns cluster_arn \
  --principals other_account_ids
```

Pour Windows :

```
aws ram create-resource-share --name descriptive_name ^
  --region region ^
  --resource-arns cluster_arn ^
  --principals other_account_ids
```

Pour inclure plusieurs comptes IDs pour le `--principals` paramètre, séparez-les les IDs uns des autres par des espaces. Pour spécifier si le compte autorisé IDs peut se trouver en dehors de votre AWS organisation, incluez le `--no-allow-external-principals` paramètre `--allow-external-principals` or pour`create-resource-share`.

AWS RAMAPI

Pour accorder l’autorisation de cloner votre cluster

1. Collectez les informations pour les paramètres requis. Vous avez besoin de l'ARN de votre cluster et de l'identifiant numérique de l'autre AWS compte.
2. Appelez l'opération AWS RAM API CreateResourceShareet spécifiez les valeurs suivantes :
   - Spécifiez l'ID de compte pour un ou plusieurs AWS comptes en tant que `principals` paramètre.
   - Spécifiez l’ARN d’un ou plusieurs clusters de bases de données Aurora comme paramètre `resourceArns`.
   - Spécifiez si le compte autorisé IDs peut se trouver en dehors de votre AWS organisation en incluant une valeur booléenne pour le `allowExternalPrincipals` paramètre.

Recréation d’un cluster qui utilise la clé RDS par défaut

Si le cluster chiffré que vous prévoyez de partager utilise la clé RDS par défaut, veillez à recréer le cluster. Pour ce faire, créez un instantané manuel de votre cluster de bases de données, utilisez une AWS KMS key, puis restaurez le cluster dans un nouveau cluster. Partagez ensuite le nouveau cluster. Pour effectuer ce processus, procédez comme suit.

Pour recréer un cluster chiffré qui utilise la clé RDS par défaut

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Snapshots (Instantanés).
3. Choisissez votre instantané.
4. Pour Actions (Actions), choisissez Copy Snapshot (Copier un instantané), puis choisissez Enable encryption (Activer le chiffrement).
5. Pour AWS KMS key, choisissez la nouvelle clé de chiffrement que vous souhaitez utiliser.
6. Restaurez l'instantané copié. Pour ce faire, suivez la procédure décrite dans Restauration à partir d'un instantané de cluster de bases de données. La nouvelle instance de base de données utilise votre nouvelle clé de chiffrement.
7. (Facultatif) Supprimez l'ancien cluster de bases de données si vous n'en n'avez plus besoin. Pour ce faire, suivez la procédure décrite dans Suppression d'un instantané de cluster de bases de données. Auparavant, confirmez que votre nouveau cluster a toutes les données nécessaires et que votre application peut y accéder avec succès.

Vérifier si un cluster dont vous êtes propriétaire est partagé avec d'autres AWS comptes

Vous pouvez vérifier si d'autres utilisateurs ont l'autorisation de partager un cluster. Procéder ainsi peut vous aider à comprendre si le cluster approche de la limite du nombre maximal de clones intercomptes.

Pour les procédures de partage de ressources à l'aide de la AWS RAM console, consultez la section Partage des ressources dont vous êtes propriétaire dans le Guide de AWS RAM l'utilisateur.

AWS CLI

Pour savoir si un cluster dont vous êtes propriétaire est partagé avec d'autres AWS comptes

- Appelez la commande AWS RAM `list-principals`CLI en utilisant votre identifiant de compte comme propriétaire de la ressource et l'ARN de votre cluster comme ARN de ressource. Vous pouvez voir tous les partages à l'aide de la commande suivante. Les résultats indiquent quels AWS comptes sont autorisés à cloner le cluster.

```
aws ram list-principals \
    --resource-arns your_cluster_arn \
```

```
    --principals your_aws_id
```

AWS RAMAPI

Pour savoir si un cluster dont vous êtes propriétaire est partagé avec d'autres AWS comptes

- Appelez l'opération AWS RAM API ListPrincipals. Utilisez votre ID de compte comme propriétaire de la ressource et l'ARN de votre cluster comme ARN de la ressource.

Clonage d'un cluster appartenant à un autre compte AWS

Pour cloner un cluster appartenant à un autre AWS compte, utilisez AWS RAM pour obtenir l'autorisation de créer le clone. Après que vous avez obtenu l'autorisation requise, utilisez la procédure standard pour cloner un cluster Aurora.

Vous pouvez également vérifier si un cluster que vous possédez est un clone d'un cluster appartenant à un autre AWS compte.

Pour connaître les procédures relatives à l'utilisation de ressources appartenant à d'autres utilisateurs dans la AWS RAM console, consultez la section Accès aux ressources partagées avec vous dans le Guide de AWS RAM l'utilisateur.

Rubriques

- Afficher les invitations à cloner des clusters appartenant à d'autres AWS comptes
- Accepter les invitations à partager des clusters appartenant à d'autres AWS comptes
- Clonage d'un cluster Aurora appartenant à un autre compte AWS
- Vérification pour savoir si un cluster de bases de données est un clone intercompte

Afficher les invitations à cloner des clusters appartenant à d'autres AWS comptes

Pour utiliser des invitations à cloner des clusters détenus par des AWS comptes d'autres AWS organisationsAWS CLI, utilisez la AWS RAM console ou l'AWS RAMAPI. Actuellement, vous ne pouvez pas exécuter cette procédure à l'aide de la console Amazon RDS.

Pour connaître les procédures relatives aux invitations dans la AWS RAM console, consultez la section Accès aux ressources partagées avec vous dans le Guide de AWS RAM l'utilisateur.

AWS CLI

Pour voir les invitations à cloner des clusters appartenant à d'autres AWS comptes

1. Exécutez la commande AWS RAM CLI `get-resource-share-invitations`.

   ```
   aws ram get-resource-share-invitations --region region_name
   ```

   Les résultats de la commande précédente affichent toutes les invitations pour cloner les clusters, y compris celles que vous avez déjà acceptées ou refusées.
2. (Facultatif) Filtrez la liste afin que vous ne puissiez voir que les invitations qui nécessitent une action de votre part. Pour ce faire, ajoutez le paramètre `--query 'resourceShareInvitations[?status==`PENDING`]'`.

AWS RAMAPI

Pour voir les invitations à cloner des clusters appartenant à d'autres AWS comptes

1. Appelez l'opération AWS RAM API `GetResourceShareInvitations`. Cette opération retourne toutes les invitations, y compris celles que vous avez déjà acceptées ou refusées.
2. (Facultatif) Recherchez uniquement les invitations qui nécessitent une action de votre part en vérifiant le champ de retour `resourceShareAssociations` pour une valeur `status` de `PENDING`.

Accepter les invitations à partager des clusters appartenant à d'autres AWS comptes

Vous pouvez accepter des invitations à partager des clusters appartenant à d'autres AWS comptes appartenant à différentes AWS organisations. Pour utiliser ces invitations, utilisez leAWS CLI, le AWS RAM et RDS APIs, ou la AWS RAM console. Actuellement, vous ne pouvez pas exécuter cette procédure avec la console RDS.

Pour connaître les procédures relatives aux invitations dans la AWS RAM console, consultez la section Accès aux ressources partagées avec vous dans le Guide de AWS RAM l'utilisateur.

AWS CLI

Pour accepter une invitation à partager un cluster depuis un autre AWS compte

1. Trouvez l'ARN de l'invitation en exécutant la commande AWS RAM CLI `get-resource-share-invitations`, comme indiqué ci-dessus.
2. Acceptez l'invitation en appelant la commande AWS RAM CLI `accept-resource-share-invitation`, comme indiqué ci-dessous.

   Pour Linux, macOS ou Unix :

   ```
   aws ram accept-resource-share-invitation \
     --resource-share-invitation-arn invitation_arn \
     --region region
   ```

   Pour Windows :

   ```
   aws ram accept-resource-share-invitation ^
     --resource-share-invitation-arn invitation_arn ^
     --region region
   ```

AWS RAMet API RDS

Pour accepter des invitations à partager le cluster de quelqu'un

1. Trouvez l'ARN de l'invitation en appelant l'opération AWS RAM API `GetResourceShareInvitations`, comme indiqué ci-dessus.
2. Transmettez cet ARN en tant que `resourceShareInvitationArn` paramètre à l'opération AcceptResourceShareInvitationd'API RDS.

Clonage d'un cluster Aurora appartenant à un autre compte AWS

Après avoir accepté l'invitation du AWS compte propriétaire du cluster de base de données, comme indiqué ci-dessus, vous pouvez cloner le cluster.

Console

Pour cloner un cluster Aurora appartenant à un autre AWS compte

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).

   En haut de la liste des bases de données, vous devez voir un ou plusieurs éléments avec une valeur Role (Rôle) égale à `Shared from account #account_id`. Pour des raisons de sécurité, vous pouvez uniquement afficher des informations limitées sur les clusters d'origine. Les propriétés que vous pouvez voir sont celles telles que le moteur de base de données et la version qui doivent être identiques dans votre cluster cloné.
3. Choisissez le cluster que vous prévoyez de cloner.
4. Pour Actions, choisissez Create clone (Créer un clone).
5. Suivez la procédure dans Console pour terminer la configuration du cluster cloné.
6. Si nécessaire, activez le chiffrement du cluster cloné. Si le cluster que vous clonez est chiffré, vous devez activer le chiffrement pour le cluster cloné. Le compte AWS avec lequel vous avez partagé le cluster doit aussi partager la clé KMS utilisée pour chiffrer le cluster. Vous pouvez utiliser la même clé KMS pour chiffrer le clone, ou utiliser votre propre clé CMK. Vous ne pouvez pas créer de clone intercompte pour un cluster chiffré avec la clé KMS par défaut.

   Le compte propriétaire de la clé de chiffrement doit accorder au compte de destination l'autorisation d'utiliser la clé à l'aide d'une politique de clé. Ce processus est similaire à la façon dont les instantanés chiffrés sont partagés, à l'aide d'une politique de clé qui accorde au compte d destination l'autorisation d'utiliser la clé.

AWS CLI

Pour cloner un cluster Aurora appartenant à un autre AWS compte

1. Acceptez l'invitation du AWS compte propriétaire du cluster de base de données, comme indiqué ci-dessus.
2. Clonez le cluster en spécifiant l'ARN complet du cluster source dans le paramètre `source-db-cluster-identifier` de la commande `restore-db-cluster-to-point-in-time` de CLI RDS, comme illustré ci-après.

Si l'ARN transmis comme `source-db-cluster-identifier` n'a pas été partagé, la même erreur est retourné comme si le cluster spécifié n'existait pas.

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-to-point-in-time \
  --source-db-cluster-identifier=arn:aws:rds:arn_details \
  --db-cluster-identifier=new_cluster_id \
  --restore-type=copy-on-write \
  --use-latest-restorable-time
```

Pour Windows :

```
aws rds restore-db-cluster-to-point-in-time ^
  --source-db-cluster-identifier=arn:aws:rds:arn_details ^
  --db-cluster-identifier=new_cluster_id ^
  --restore-type=copy-on-write ^
  --use-latest-restorable-time
```

3. Si le cluster que vous clonez est chiffré, chiffrez-le en incluant un paramètre `kms-key-id`. La valeur `kms-key-id` peut être la même que celle utilisée pour chiffrer le cluster de bases de données d'origine ou votre propre clé KMS. Votre compte doit avoir l'autorisation d'utiliser cette clé de chiffrement.

   Pour Linux, macOS ou Unix :

   ```
   aws rds restore-db-cluster-to-point-in-time \
     --source-db-cluster-identifier=arn:aws:rds:arn_details \
     --db-cluster-identifier=new_cluster_id \
     --restore-type=copy-on-write \
     --use-latest-restorable-time \
     --kms-key-id=arn:aws:kms:arn_details
   ```

   Pour Windows :

   ```
   aws rds restore-db-cluster-to-point-in-time ^
     --source-db-cluster-identifier=arn:aws:rds:arn_details ^
     --db-cluster-identifier=new_cluster_id ^
     --restore-type=copy-on-write ^
     --use-latest-restorable-time ^
   ```

```
  --kms-key-id=arn:aws:kms:arn_details
```

Le compte propriétaire de la clé de chiffrement doit accorder au compte de destination l'autorisation d'utiliser la clé à l'aide d'une politique de clé. Ce processus est similaire à la façon dont les instantanés chiffrés sont partagés, à l'aide d'une politique de clé qui accorde au compte d destination l'autorisation d'utiliser la clé. Un exemple de politique de clé suit.

JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
```

```
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

> **Note**
>
> La commande restore-db-cluster-to- point-in-time AWS CLI restaure uniquement le cluster de base de données, pas les instances de base de données pour ce cluster de base de données. Pour créer des instances de base de données pour le cluster de base de données restauré, appelez la create-db-instancecommande. Spécifiez l'identifiant du cluster de bases de données restauré dans `--db-cluster-identifier`.
> Vous pouvez créer des instances de base de données uniquement après la fin de la commande `restore-db-cluster-to-point-in-time` et lorsque le cluster de bases de données est disponible.

API RDS

Pour cloner un cluster Aurora appartenant à un autre AWS compte

1. Acceptez l'invitation du AWS compte propriétaire du cluster de base de données, comme indiqué ci-dessus.
2. Clonez le cluster en spécifiant l'ARN complet du cluster source dans le paramètre `SourceDBClusterIdentifier` de l'opération `RestoreDBClusterToPointInTime` de l'API RDS.

   Si l'ARN transmis comme `SourceDBClusterIdentifier` n'a pas été partagé, la même erreur est retournée comme si le cluster spécifié n'existait pas.

3. Si le cluster que vous clonez est chiffré, incluez un paramètre `KmsKeyId` pour chiffrer le cluster cloné. La valeur `kms-key-id` peut être la même que celle utilisée pour chiffrer le cluster de bases de données d'origine ou votre propre clé KMS. Votre compte doit avoir l'autorisation d'utiliser cette clé de chiffrement.

   Lors du clonage d'un volume, le compte de destination doit avoir l'autorisation d'utiliser la clé de chiffrement utilisée pour chiffrer le cluster source. Aurora chiffre le nouveau cluster cloné avec la clé de chiffrement spécifiée dans `KmsKeyId`.

   Le compte propriétaire de la clé de chiffrement doit accorder au compte de destination l'autorisation d'utiliser la clé à l'aide d'une politique de clé. Ce processus est similaire à la façon dont les instantanés chiffrés sont partagés, à l'aide d'une politique de clé qui accorde au compte d destination l'autorisation d'utiliser la clé. Un exemple de politique de clé suit.

   JSON

```
{
    "Id": "key-policy-1",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/KeyUser",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                }
            }
        }
    ]
}
```

> **Note**
>
> L'opération Restore DBCluster ToPointInTime RDS API restaure uniquement le cluster de base de données, pas les instances de base de données pour ce cluster de base de données. Pour créer des instances de base de données pour le cluster de base de données restauré, appelez l'opération Create DBInstance RDS API. Spécifiez l'identifiant du cluster de bases de données restauré dans `DBClusterIdentifier`. Vous pouvez créer des instances de base de données une fois seulement l'opération `RestoreDBClusterToPointInTime` terminée et le cluster de bases de données disponible.

Vérification pour savoir si un cluster de bases de données est un clone intercompte

L'objet `DBClusters` identifie si chaque cluster est un clone intercompte. Vous pouvez voir les clusters que vous avez l'autorisation de cloner en utilisant l'option `include-shared` lorsque vous exécutez la commande `describe-db-clusters` de CLI RDS. Cependant, vous ne pouvez pas voir la plupart des détails de configuration de tels clusters.

AWS CLI

Pour vérifier si un cluster de bases de données est un clone entre comptes

- Appelez la commande de CLI RDS `describe-db-clusters`.

  L'exemple suivant montre comment les clusters de bases de données de clone intercompte actuel ou potentiel apparaissent dans la sortie `describe-db-clusters`. Pour les clusters existants appartenant à votre AWS compte, le `CrossAccountClone` champ indique s'il s'agit d'un clone d'un cluster de base de données appartenant à un autre AWS compte.

  Dans certains cas, le numéro de AWS compte d'une entrée peut être différent du vôtre dans le `DBClusterArn` champ. Dans ce cas, cette entrée représente un cluster qui appartient à un autre AWS compte et que vous pouvez cloner. De telles entrées ont quelques champs autres que `DBClusterArn`. Lors de la création du cluster cloné, spécifiez les mêmes valeurs `StorageEncrypted`, `Engine` et `EngineVersion` que dans le cluster d'origine.

```
$aws rds describe-db-clusters --include-shared --region us-east-1
{
  "DBClusters": [
      {
          "EarliestRestorableTime": "2023-02-01T21:17:54.106Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": false,
...
      },
      {
          "EarliestRestorableTime": "2023-02-09T16:01:07.398Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": true,
...
      },
      {
          "StorageEncrypted": false,
          "DBClusterArn": "arn:aws:rds:us-east-1:12345678:cluster:cluster-
abcdefgh",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0
  ]
```

```
}
```

API RDS

Pour vérifier si un cluster de bases de données est un clone entre comptes

- Appelez l'opération DBClustersDescribe de l'API RDS.

  Pour les clusters existants appartenant à votre AWS compte, le `CrossAccountClone` champ indique s'il s'agit d'un clone d'un cluster de base de données appartenant à un autre AWS compte. Les entrées avec un numéro de AWS compte différent dans le `DBClusterArn` champ représentent des clusters que vous pouvez cloner et qui appartiennent à d'autres AWS comptes. Ces entrées ont quelques champs autres que `DBClusterArn`. Lors de la création du cluster cloné, spécifiez les mêmes valeurs `StorageEncrypted`, `Engine` et `EngineVersion` que dans le cluster d'origine.

  L'exemple suivant montre une valeur de retour qui illustre les clusters clonés réels et potentiels.

```
{
  "DBClusters": [
      {
          "EarliestRestorableTime": "2023-02-01T21:17:54.106Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": false,
...
      },
      {
          "EarliestRestorableTime": "2023-02-09T16:01:07.398Z",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0",
          "CrossAccountClone": true,
...
      },
      {
          "StorageEncrypted": false,
          "DBClusterArn": "arn:aws:rds:us-east-1:12345678:cluster:cluster-
abcdefgh",
          "Engine": "aurora-mysql",
          "EngineVersion": "8.0.mysql_aurora.3.02.0"
      }
```

```
    ]
}
```

Intégration d'Aurora à d'autres AWS services

Intégrez Amazon Aurora à d'autres AWS services afin de pouvoir étendre votre cluster de base de données Aurora afin d'utiliser des fonctionnalités supplémentaires dans le AWS cloud.

Rubriques

Intégration de AWS services avec Amazon Aurora MySQL

Amazon Aurora MySQL s'intègre à d'autres AWS services afin que vous puissiez étendre votre cluster de bases de données Aurora MySQL afin d'utiliser des fonctionnalités supplémentaires dans le AWS cloud. Votre cluster de base de données Aurora MySQL peut utiliser AWS des services pour effectuer les opérations suivantes :

- Invoquez une AWS Lambda fonction de manière synchrone ou asynchrone à l'aide des fonctions natives ou. `lambda_sync` `lambda_async` Ou appeler de façon asynchrone une fonction AWS Lambda en utilisant la procédure `mysql.lambda_async`.
- Chargez dans votre cluster de bases de données les données de fichiers texte ou XML stockés dans un compartiment Amazon S3 à l'aide de la commande `LOAD DATA FROM S3` ou `LOAD XML FROM S3`.
- Enregistrer des données dans des fichiers texte stockés dans un compartiment Amazon S3 à partir de votre cluster de bases de données à l'aide de la commande `SELECT INTO OUTFILE S3`.
- Ajouter ou supprimer automatiquement des réplicas Aurora avec Application Auto Scaling. Pour de plus amples informations, veuillez consulter Amazon Aurora Auto Scaling avec des réplicas Aurora.

Pour plus d'informations sur l'intégration d'Aurora MySQL à d'autres AWS services, consultezIntégration d'Amazon Aurora MySQL avec d'autres services AWS.

Intégration de AWS services avec Amazon Aurora PostgreSQL

Amazon Aurora PostgreSQL s'intègre à AWS d'autres services afin que vous puissiez étendre votre cluster de bases de données Aurora PostgreSQL afin d'utiliser des fonctionnalités supplémentaires dans le cloud. AWS Votre cluster de base de données Aurora PostgreSQL peut AWS utiliser les services pour effectuer les opérations suivantes :

- Recueillez, affichez et évaluez rapidement les performances des charges de travail de votre base de données avec Performance Insights.
- Ajouter ou supprimer automatiquement des réplicas Aurora avec Aurora Auto Scaling. Pour de plus amples informations, veuillez consulter Amazon Aurora Auto Scaling avec des réplicas Aurora.

Pour plus d'informations sur l'intégration d'Aurora PostgreSQL à AWS d'autres services, consultez. Intégration d’Amazon Aurora PostgreSQL avec d’autres services AWS

Entretien d’un cluster de bases de données Amazon Aurora

Amazon RDS effectue régulièrement la maintenance des ressources Amazon RDS. Les rubriques suivantes décrivent ces actions de maintenance et expliquent comment les appliquer.

Présentation des mises à jour relatives au cluster de bases de données

La maintenance implique le plus souvent la mise à jour des ressources suivantes dans votre cluster de bases de données :

- Matériel sous-jacent
- Système d’exploitation (SE) sous-jacent
- Version du moteur de base de données

Les mises à jour du système d’exploitation se produisent le plus souvent pour des raisons de sécurité. Nous vous recommandons de les effectuer dès que possible. Pour plus d’informations sur les mise à jour des systèmes d’exploitation, consultez the section called “Mises à jour du système d’exploitation”.

Rubriques

Ressources hors ligne lors des mises à jour de maintenance

Certains éléments de maintenance exigent qu’Amazon RDS mette votre cluster de bases de données hors ligne pendant un court moment. Parmi les éléments de maintenance qui nécessitent qu’une ressource soit hors ligne figure l’application obligatoire de correctifs au système d’exploitation ou à la base de données. Les mises à jour correctives obligatoires sont planifiées automatiquement uniquement pour les correctifs associés à la sécurité et à la fiabilité de l’instance. Ce type d’application de correctifs est peu fréquent, généralement une fois tous les quelques mois. Cela nécessite rarement plus d’une fraction de votre fenêtre de maintenance.

Modifications différées de l’instance de base de données et du cluster de bases de données

Les modifications différées des clusters et des instances de base de données que vous avez choisi de ne pas appliquer immédiatement le sont pendant la fenêtre de maintenance. Par exemple, vous pouvez choisir de modifier les classes des instances de base de données, un cluster de bases de données ou des groupes de paramètres de base de données pendant le créneau de maintenance. Les modifications que vous spécifiez à l’aide du paramètre de redémarrage en attente n’apparaissent pas dans la liste Maintenance en attente. Pour plus d’informations sur la modification d’un cluster de bases de données, consultez Modification d’un cluster de bases de données Amazon Aurora.

Pour voir les modifications en attente pour la prochaine fenêtre de maintenance, utilisez la describe-db-clusters AWS CLI commande et cochez le `PendingModifiedValues` champ.

Cohérence éventuelle de l' DescribePendingMaintenanceActions API

L’API `DescribePendingMaintenanceActions` Amazon RDS suit un modèle de cohérence à terme. Cela signifie que le résultat de la commande `DescribePendingMaintenanceActions` peut ne pas être immédiatement visible pour toutes les commandes RDS suivantes. Gardez cela à l’esprit lorsque vous utilisez `DescribePendingMaintenanceActions` immédiatement après avoir utilisé une commande API précédente.

La cohérence à terme peut affecter la façon dont vous avez géré vos mises à jour de maintenance. Par exemple, si vous exécutez la commande `ApplyPendingMaintenanceActions` pour mettre à jour la version du moteur de base de données pour un cluster de bases de données, elle sera finalement visible par `DescribePendingMaintenanceActions`. Dans ce scénario, `DescribePendingMaintenanceActions` peut indiquer que l’action de maintenance n’a pas été appliquée alors qu’elle l’était.

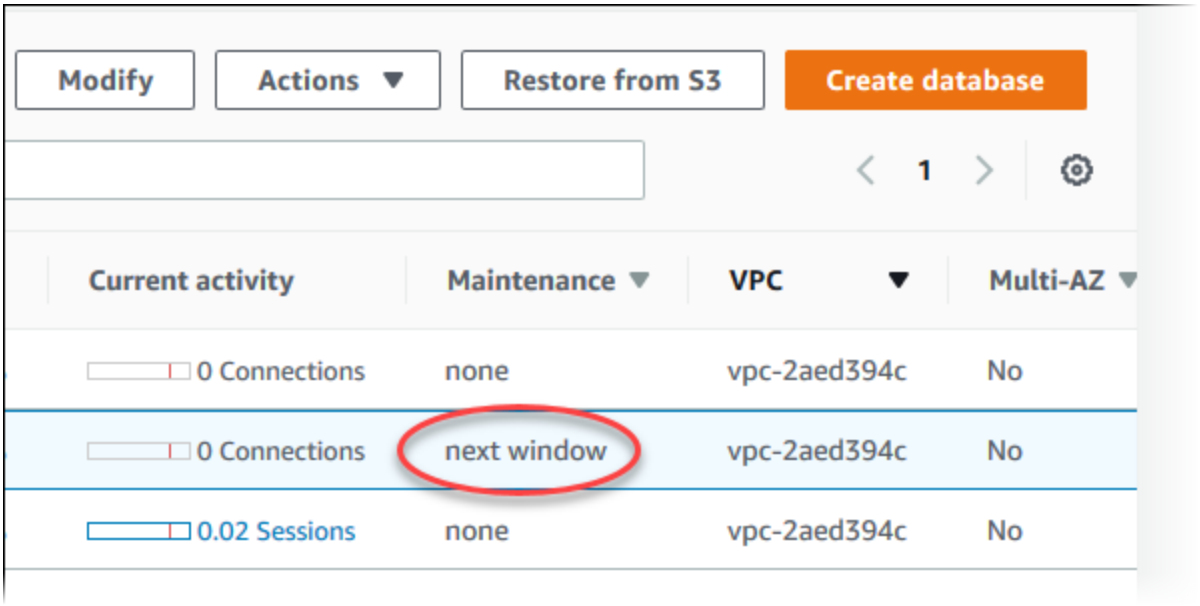
Pour gérer la cohérence à terme, vous pouvez procéder comme suit :

- Vérifiez l’état du cluster de bases de données avant d’exécuter une commande pour le modifier. Exécutez la commande `DescribePendingMaintenanceActions` appropriée à l’aide d’un algorithme de backoff exponentiel afin de laisser suffisamment de temps à la commande précédente pour se propager dans le système. Pour ce faire, exécutez la commande `DescribePendingMaintenanceActions` à plusieurs reprises, en commençant par quelques secondes d’attente, puis en augmentant progressivement jusqu’à cinq minutes.
- Ajoutez un temps d’attente entre les commandes suivantes, même si une commande `DescribePendingMaintenanceActions` renvoie une réponse précise. Appliquez un algorithme

de backoff exponentiel en commençant par quelques secondes d'attente, puis augmentez progressivement jusqu'à environ cinq minutes.

Affichage des mise à jour de maintenance en attente

Vérifiez si une mise à jour de maintenance est disponible pour votre cluster d' de base de données à l'aide de la console RDS, de l' AWS CLI API RDS ou de l'API RDS. Si une mise à jour est disponible, elle est indiquée dans la colonne Maintenance pour le cluster de bases de données sur la console Amazon RDS, comme illustré dans cette figure.



Si aucune mise à jour de maintenance n'est disponible pour un cluster de bases de données, la valeur de la colonne est none.

Si une mise à jour de maintenance est disponible pour un cluster de bases de données, les valeurs de colonne suivantes sont possibles :

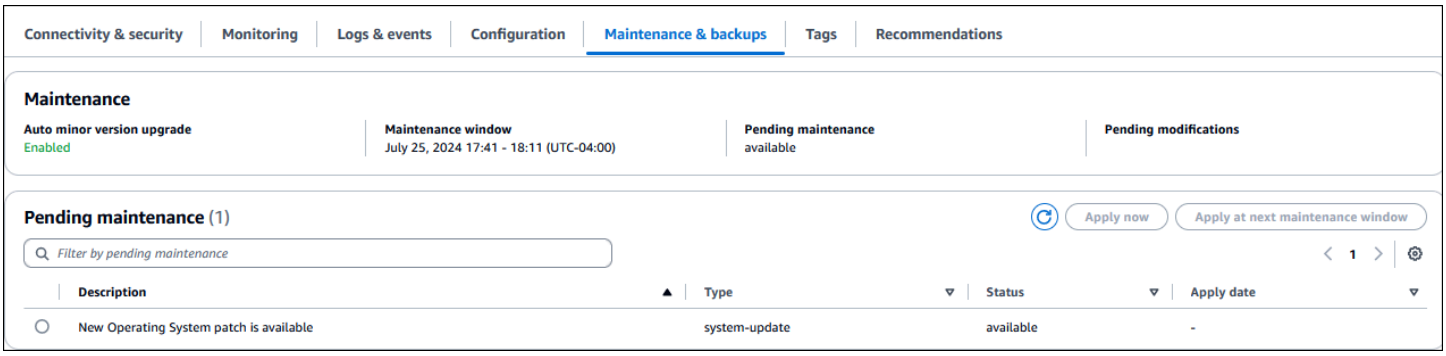
- required (obligatoire) : l'action de maintenance sera appliquée à la ressource et ne peut pas être reportée indéfiniment.
- available : l'action de maintenance est disponible, mais ne sera pas appliquée automatiquement à la ressource. Vous pouvez l'appliquer manuellement.
- next window : l'action de maintenance sera appliquée à la ressource lors de la prochaine fenêtre de maintenance.
- En cours : l'action de maintenance est en cours d'application à la ressource.

Si une mise à jour est disponible, vous pouvez procéder de l’une des manières suivantes :

- Si la valeur de maintenance est fenêtre suivante, reportez les éléments de maintenance en choisissant Reporter la mise à niveau dans Actions. Vous ne pouvez pas reporter une action de maintenance en cours.
- Appliquer immédiatement les actions de maintenance.
- Appliquez les actions de maintenance pendant la fenêtre de maintenance suivante.
- Ne rien faire.

Pour effectuer une action à l'aide du AWS Management Console

1. Sélectionnez le nom de l’instance de base de donnéesou du cluster de bases de données pour afficher ses détails.
2. Choisissez Maintenance et sauvegardes. Les actions de maintenance en attente s’affichent.
3. Choisissez l’action à effectuer, puis choisissez quand l’appliquer.



La fenêtre de maintenance détermine quand les opérations en attente démarrent et ne limite pas la durée d’exécution totale de ces opérations. Il n’est pas garanti que les opérations de maintenance seront terminées avant la fin de la fenêtre de maintenance ; elles peuvent continuer au-delà de l’heure de fin spécifiée. Pour de plus amples informations, veuillez consulter Fenêtre de maintenance Amazon RDS.

Vous pouvez également voir si une mise à jour de maintenance est disponible pour votre cluster d' de base de données en exécutant la describe-pending-maintenance-actions AWS CLI commande.

Pour en savoir plus sur l’application de mises à jour, consultez Application des mises à jour à un cluster de bases de données.

Actions de maintenance pour Amazon Aurora

Les actions de maintenance suivantes s'appliquent aux clusters de bases de données Aurora :

- `os-upgrade` : mettez à jour les systèmes d'exploitation de toutes les instances de base de données du cluster de bases de données, en utilisant des mises à niveau progressives. Pour plus d'informations, consultez the section called "Mises à jour du système d'exploitation".
- `system-update` : appliquez des correctifs du moteur de base de données pour Aurora PostgreSQL.

Les actions de maintenance suivantes s'appliquent aux instances de base de données Aurora :

- `ca-certificate-rotation` : mettez à jour le certificat de l'autorité de certification Amazon RDS pour l'instance de base de données.
- `hardware-maintenance` : effectuez la maintenance du matériel sous-jacent pour l'instance de base de données.
- `system-update` : mettez à jour le système d'exploitation de l'instance de base de données.

Choix de la fréquence des mises à jour de maintenance d'Aurora MySQL

Vous pouvez contrôler la fréquence des mises à niveau d'Aurora MySQL pour chaque cluster de bases de données. Le meilleur choix dépend de votre utilisation d'Aurora MySQL et des priorités pour vos applications s'exécutant sur Aurora. Pour plus d'informations sur les versions LTS (long-term stability) d'Aurora MySQL qui requièrent des mises à niveau moins fréquentes, consultez Versions à long terme (LTS) d'Aurora MySQL.

Vous pouvez choisir de n'effectuer que de rares mises à niveau d'un cluster Aurora MySQL si certaines ou toutes les conditions suivantes s'appliquent :

- Le cycle de test de votre application dure très longtemps pour chaque mise à niveau du moteur de base de données Aurora MySQL.
- Vous avez de nombreux clusters de bases de données ou de nombreuses applications s'exécutant tous ou toutes sur la même version d'Aurora MySQL. Vous préférez mettre à niveau tous vos clusters de bases de données et toutes vos applications associées en même temps.
- Vous utilisez à la fois Aurora MySQL et RDS for MySQL.. Vous préférez que les clusters Aurora MySQL et les instances de base de données RDS for MySQL restent compatibles avec le même niveau de MySQL.

- Votre application Aurora MySQL est en production ou est très importante pour votre entreprise. Vous ne pouvez pas vous permettre d'avoir une durée d'indisponibilité pour les mises à niveau, en dehors des rares occurrences de correctifs critiques.
- Votre application Aurora MySQL n'est pas limitée par les problèmes de performances ou les manques de fonctionnalités résolus dans les suivantes d'Aurora MySQL.

Si les facteurs précédemment indiqués reflètent votre situation, vous pouvez limiter le nombre de mises à niveau forcées pour un cluster de bases de données Aurora MySQL. Pour cela, vous devez choisir une version d'Aurora MySQL spécifique connue sous le nom de version « Long-Term Support » (LTS) lorsque vous créez ou mettez à niveau ce cluster de bases de données. Cela permet de réduire le nombre de cycles de mises à niveau, de cycles de test et d'interruptions liées aux mises à niveau pour ce cluster de bases de données.

Vous pouvez choisir d'effectuer des mises à niveau fréquentes d'un cluster Aurora MySQL si certaines ou toutes les conditions suivantes s'appliquent :

- Le cycle de test de votre application est simple et bref.
- Votre application est toujours au stade du développement.
- Votre environnement de base de données utilise plusieurs versions d'Aurora MySQL ou des versions d'Aurora MySQL et de RDS for MySQL. Chaque cluster Aurora MySQL possède son propre cycle de mise à niveau.
- Vous attendez des améliorations de performances ou de fonctionnalités spécifiques avant d'accroître votre utilisation d'Aurora MySQL.

Si les facteurs précédemment indiqués reflètent votre situation, vous pouvez autoriser Aurora à appliquer d'importantes mises à niveau plus fréquemment. Pour ce faire, mettez à niveau un cluster de bases de données Aurora MySQL vers une version d'Aurora MySQL plus récente que la version LTS. Cela vous permettra de bénéficier plus rapidement des dernières améliorations de performances, des derniers correctifs de bogues et des dernières fonctionnalités disponibles.

Fenêtre de maintenance Amazon RDS

La fenêtre de maintenance est un intervalle de temps hebdomadaire au cours duquel toutes les modifications système sont appliquées. Chaque cluster d' de base de données dispose d'une fenêtre de maintenance hebdomadaire. Considérez la fenêtre de maintenance comme une occasion de contrôler le moment où les modifications et les correctifs logiciels sont appliqués. Pour plus

d’informations sur l’ajustement des fenêtres de maintenance, consultez Ajustement du créneau de maintenance préféré pour un cluster de bases de données.

RDS consomme certaines des ressources de votre cluster de bases de données pendant les opérations de maintenance. Vous remarquerez peut-être un effet minimal sur les performances. Dans le cas d’une instance de base de données, en de rares occasions, un basculement Multi-AZ peut être requis pour terminer une mise à jour de maintenance.

Si un événement de maintenance est planifié pour une semaine donnée, il est déclenché pendant le créneau de maintenance de 30 minutes que vous identifiez. La plupart des événements de maintenance se terminent également au cours du créneau de maintenance de 30 minutes, mais des événements de maintenance plus importants peuvent prendre plus de 30 minutes. La fenêtre de maintenance est suspendue lors de l’arrêt du cluster de bases de données.

Ce créneau de maintenance de 30 minutes est sélectionné de manière aléatoire sur un bloc horaire de 8 heures par région. Si vous ne spécifiez pas de créneau de maintenance lors de la création du cluster de bases de données, RDS attribue un créneau de maintenance de 30 minutes un jour de semaine aléatoire.

Le tableau suivant indique les intervalles de temps pour chacun Région AWS desquels les fenêtres de maintenance par défaut sont attribuées.

Nom de la région	Région	Bloc chronologique
USA Est (Virginie du Nord)	us-east-1	03:00–11:00 UTC
USA Est (Ohio)	us-east-2	03:00–11:00 UTC
USA Ouest (Californie du Nord)	us-west-1	06:00–14:00 UTC
US West (Oregon)	us-west-2	06:00–14:00 UTC
Africa (Cape Town)	af-south-1	03:00–11:00 UTC
Asie-Pacifique (Hong Kong)	ap us-east-1	06:00–14:00 UTC

Nom de la région	Région	Bloc chronologique
Asie-Pacifique (Hyderabad)	ap-south-2	06:30–14:30 UTC
Asie-Pacifique (Jakarta)	ap-southeast-3	08:00–16:00 UTC
Asie-Pacifique (Malaisie)	ap-southeast-5	09:00–17:00 UTC
Asie-Pacifique (Melbourne)	ap-southeast-4	11:00–19:00 UTC
Asie-Pacifique (Mumbai)	ap-south-1	06:00–14:00 UTC
Asie-Pacifique (Nouvelle Zélande)	ap-southeast-6	13:00–21:00 UTC
Asie-Pacifique (Osaka)	ap-northeast-3	22:00–23:59 UTC
Asia Pacific (Seoul)	ap-northeast-2	13:00–21:00 UTC
Asia Pacific (Singapore)	ap-southeast-1	14:00–22:00 UTC
Asia Pacific (Sydney)	ap-southeast-2	12:00–20:00 UTC
Asie-Pacifique (Taipei)	ap-east-2	9:00–17:00 UTC
Asie-Pacifique (Thaïlande)	ap-southeast-7	8:00–16:00 UTC
Asie-Pacifique (Tokyo)	ap-northeast-1	13:00–21:00 UTC
Canada (Centre)	ca-central-1	03:00–11:00 UTC

Nom de la région	Région	Bloc chronologique
Canada-Ouest (Calgary)	ca-west-1	18:00–02:00 UTC
Chine (Pékin)	cn-north-1	06:00–14:00 UTC
China (Ningxia)	cn-northwest-1	06:00–14:00 UTC
Europe (Frankfurt)	eu-central-1	21:00–05:00 UTC
Europe (Ireland)	eu-west-1	22:00–06:00 UTC
Europe (London)	eu-west-2	22:00–06:00 UTC
Europe (Milan)	eu-south-1	02:00–10:00 UTC
Europe (Paris)	eu-west-3	23:59–07:29 UTC
Europe (Espagne)	eu-south-2	02:00–10:00 UTC
Europe (Stockholm)	eu-north-1	23:00–07:00 UTC
Europe (Zurich)	eu-central-2	02:00–10:00 UTC
Israël (Tel Aviv)	il-central-1	03:00–11:00 UTC
Mexique (Centre)	mx-central-1	19:00–03:00 UTC
Moyen-Orient (Bahreïn)	me-south-1	06:00–14:00 UTC
Moyen-Orient (EAU)	me-central-1	05:00–13:00 UTC
Amérique du Sud (São Paulo)	sa-east-1	00:00–08:00 UTC
AWS GovCloud (USA Est)	us-gov-east-1	17:00–01:00 UTC

Nom de la région	Région	Bloc chronologique
AWS GovCloud (US-Ouest)	us-gov-west-1	06:00–14:00 UTC

Rubriques

- Ajustement du créneau de maintenance préféré pour un cluster de bases de données

Ajustement du créneau de maintenance préféré pour un cluster de bases de données

Le créneau de maintenance de cluster de bases de données Aurora doit intervenir au moment où l'utilisation est la plus faible et peut donc nécessiter d'être modifié de temps en temps. Votre cluster de bases de données est indisponible pendant ce délai uniquement si les mises à jour appliquées nécessitent une interruption de service. L'interruption de service ne dure que le délai minimal requis pour effectuer les mises à jour nécessaires.

> **Note**
>
> Pour les mises à niveau du moteur de base de données, Amazon Aurora gère le créneau de maintenance préféré pour un cluster de bases de données et non pour chaque instance.

Console

Pour ajuster le créneau de maintenance préféré d'un cluster de bases de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster de bases de données pour lequel vous souhaitez modifier la fenêtre de maintenance.
4. Sélectionnez Modify.
5. Dans la section Maintenance, mettez à jour la fenêtre de maintenance.
6. Choisissez Continuer.

   Sur la page de confirmation, examinez vos modifications.

7. Pour appliquer immédiatement les modifications à la fenêtre de maintenance, choisissez Immédiatement dans la section Planification des modifications.
8. Choisissez Modifier le cluster pour enregistrer vos modifications.

   Sinon, choisissez Retour pour modifier vos modifications, ou choisissez Annuler pour les annuler.

AWS CLI

Pour ajuster la fenêtre de maintenance du cluster de base de données préférée, utilisez la AWS CLI `modify-db-cluster`commande avec les paramètres suivants :

- `--db-cluster-identifier`
- `--preferred-maintenance-window`

Example

L'exemple de code suivant définit la fenêtre de maintenance sur les mardis entre 04h00–04h30 UTC.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
--db-cluster-identifier my-cluster \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

Pour Windows :

```
aws rds modify-db-cluster ^
--db-cluster-identifier my-cluster ^
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

API RDS

Pour ajuster la fenêtre de maintenance préférée d'un cluster de bases de données, utilisez l'opération `ModifyDBCluster` de l'API Amazon RDS avec les paramètres suivants :

- `DBClusterIdentifier`
- `PreferredMaintenanceWindow`

Application des mises à jour à un cluster de bases de données

Amazon RDS vous permet de choisir le moment d'application des opérations de maintenance. Vous pouvez décider quand Amazon RDS applique les mises à jour en utilisant l' AWS Management Console API AWS CLI, ou RDS.

Console

Pour gérer une mise à jour du système d'exploitation pour un cluster de bases de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster de bases de données pour lequel une mise à jour obligatoire est disponible.
4. Sous Actions, choisissez une des options suivantes :

   - Appliquer le correctif maintenant
   - Appliquer le correctif au créneau suivant

   > **Note**
   >
   > Si vous choisissez Appliquer le correctif au créneau suivant et que vous souhaitez ensuite retarder la mise à jour, vous pouvez choisir Reporter la mise à niveau. Vous ne pouvez pas reporter une action de maintenance en cours.
   > Pour annuler une action de maintenance, modifiez l'instance de base de données et désactivez Mise à niveau automatique des versions mineures.

AWS CLI

Pour appliquer une mise à jour en attente à un cluster d' de base de données, utilisez la apply-pending-maintenance-action AWS CLI commande.

Example

Pour Linux, macOS ou Unix :

```
aws rds apply-pending-maintenance-action \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db \
```

```
    --apply-action system-update \
    --opt-in-type immediate
```

Pour Windows :

```
aws rds apply-pending-maintenance-action ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db ^
    --apply-action system-update ^
    --opt-in-type immediate
```

> **Note**
>
> Pour différer une action de maintenance, spécifiez `undo-opt-in` pour `--opt-in-type`. Vous ne pouvez pas indiquer`undo-opt-in` pour `--opt-in-type` si l'action de maintenance est en cours.
> Pour annuler une action de maintenance, exécutez la modify-db-instance AWS CLI commande et spécifiez`--no-auto-minor-version-upgrade`.

Pour renvoyer une liste des ressources dont au moins une mise à jour est en attente, utilisez la describe-pending-maintenance-actions AWS CLI commande.

Example

Pour Linux, macOS ou Unix :

```
aws rds describe-pending-maintenance-actions \
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

Pour Windows :

```
aws rds describe-pending-maintenance-actions ^
    --resource-identifier arn:aws:rds:us-west-2:001234567890:db:mysql-db
```

Vous pouvez également renvoyer une liste de ressources pour un cluster d' de base de données en spécifiant le `--filters` paramètre de la `describe-pending-maintenance-actions` AWS CLI commande. Le format de la commande `--filters` est `Name=`*`filter-name`*`,Value=`*`resource-id`*`,...`.

Les valeurs suivantes sont acceptées pour le paramètre `Name` d'un filtre :

- `db-instance-id`— Accepte une liste d'identifiants d'instance de base de données ou de noms de ressources Amazon (ARNs). La liste renvoyée inclut uniquement les actions de maintenance en attente pour les instances de base de données identifiées par ces identifiants ou ARNs.
- `db-cluster-id`— Accepte une liste d'identifiants de clusters de bases de données ou ARNs pour Amazon Aurora. La liste renvoyée inclut uniquement les actions de maintenance en attente pour les clusters de base de données identifiés par ces identifiants ou ARNs.

Par exemple, l'exemple suivant renvoie les actions de maintenance en attente pour les clusters de bases de données `sample-cluster1` et `sample-cluster2`.

Example

Pour Linux, macOS ou Unix :

```
aws rds describe-pending-maintenance-actions \
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

Pour Windows :

```
aws rds describe-pending-maintenance-actions ^
 --filters Name=db-cluster-id,Values=sample-cluster1,sample-cluster2
```

API RDS

Pour appliquer une mise à jour à un cluster de bases de données, appelez l'opération `ApplyPendingMaintenanceAction` de l'API Amazon RDS.

Pour renvoyer la liste des ressources qui possèdent au moins une mise à jour en attente, appelez l'opération d'API Amazon RDS `DescribePendingMaintenanceActions`.

Mises à niveau automatiques des versions mineures pour les clusters de bases de données Aurora

Le paramètre Mise à niveau automatique des versions mineures spécifie si Aurora applique automatiquement les mises à niveau à votre cluster de bases de données. Ces mises à niveau incluent de nouvelles versions mineures contenant des fonctionnalités supplémentaires et des correctifs de bogues.

Les mises à niveau automatiques des versions mineures mettent régulièrement à jour votre base de données avec les versions récentes du moteur de base de données. Toutefois, la mise à niveau n'inclut pas toujours la dernière version du moteur de base de données. Si vous devez conserver des versions spécifiques de vos bases de données à un moment donné, nous vous recommandons de procéder à une mise à niveau manuelle vers les versions de base de données dont vous avez besoin conformément au calendrier requis. En cas de problèmes de sécurité critiques ou lorsqu'une version atteint sa end-of-support date limite, Aurora peut appliquer une mise à niveau de version mineure même si vous n'avez pas activé l'option de mise à niveau automatique de la version mineure. Pour plus d'informations, consultez la documentation de la mise à niveau de votre moteur de base de données spécifique.

Consultez Mise à niveau de la version mineure ou du niveau de correctif d'un cluster de bases de données Aurora MySQL et Réalisation d'une mise à niveau de version mineure.

> **Note**
>
> Aurora Global Database ne prend pas en charge les mises à niveau automatiques des versions mineures.

Ce paramètre est activé par défaut. Pour chaque nouveau cluster de bases de données, choisissez la valeur appropriée de ce paramètre. Cette valeur repose sur son importance, sa durée de vie prévue et le nombre de tests de vérification effectués après chaque mise à niveau.

Pour obtenir des instructions sur l'activation et la désactivation du paramètre Mise à niveau automatique des versions mineures, consultez les références suivantes :

- Activation des mises à niveau automatiques des versions mineures pour un cluster de bases de données Aurora
- Activation des mises à niveau automatiques des versions mineures pour les instances de base de données individuelles dans un cluster de bases de données Aurora

> **Important**
>
> Pour les clusters de bases de données nouveaux et existants, nous vous recommandons vivement d'appliquer ce paramètre au cluster de bases de données et non aux instances de base de données du cluster individuellement. Si ce paramètre est désactivé dans une

instance de base de données quelconque de votre cluster, le cluster de bases de données n'est pas automatiquement mis à niveau.

Le tableau suivant montre comment fonctionne le paramètre Mise à niveau automatique des versions mineures lorsqu'il est appliqué aux niveaux du cluster et de l'instance.

Action	Paramètre du cluster	Paramètres des instances	Le cluster est-il mis à niveau automatiquement ?
Vous le définissez sur True sur le cluster de bases de données.	True	True pour toutes les instances nouvelles et existantes	Oui
Vous le définissez sur False sur le cluster de bases de données.	False	False pour toutes les instances nouvelles et existantes	Non
Il était précédemment défini sur True sur le cluster de bases de données. Vous le définissez sur False sur au moins une instance de base de données.	Devient False	False pour une ou plusieurs instances	Non
Il était précédemment défini sur False sur le cluster de bases de données. Vous le définissez sur True sur au moins une instance de base de données, mais	False	True pour une ou plusieurs instances, mais pas pour toutes les instances	Non

Action	Paramètre du cluster	Paramètres des instances	Le cluster est-il mis à niveau automatiquement ?
pas sur toutes les instances.			
Il était précédemment défini sur False sur le cluster de bases de données. Vous le définissez sur True sur toutes les instances de base de données.	Devient True	True pour toutes les instances	Oui

Les mises à niveau automatiques des versions mineures sont communiquées à l'avance via un événement de cluster de bases de données Amazon RDS avec une catégorie `maintenance` et un ID `RDS-EVENT-0156`. Pour plus d'informations, consultez Catégories d'événements et messages d'événements pour Aurora.

Des mises à niveau automatiques se produisent dans la fenêtre de maintenance. Si les différentes instances de base de données du cluster de bases de données ont des fenêtres de maintenance différentes de la fenêtre de maintenance du cluster, cette dernière est prioritaire.

Pour plus d'informations sur les mises à jour de moteur pour Aurora PostgreSQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora PostgreSQL.

Pour plus d'informations sur le paramètre Mise à niveau automatique de versions mineures pour Aurora MySQL, consultez Activation des mises à niveau automatiques entre versions mineures Aurora MySQL. Pour obtenir des informations générales sur les mises à jour de moteur pour Aurora MySQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL.

Activation des mises à niveau automatiques des versions mineures pour un cluster de bases de données Aurora

Suivez la procédure générale de la rubrique Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API.

Console

Sur la page Modifier le cluster de bases de données, dans la section Maintenance, cochez la case Activer la mise à niveau automatique des versions mineures.

AWS CLI

Appelez la commande modify-db-cluster AWS CLI . Spécifiez le nom de votre cluster de bases de données pour l'option `--db-cluster-identifier` et `true` pour l'option `--auto-minor-version-upgrade`. Si vous le souhaitez, spécifiez l'option `--apply-immediately` pour activer immédiatement ce paramètre pour votre cluster de bases de données.

API RDS

Appelez l'opération DBClusterModify API et spécifiez le nom de votre cluster de base de données pour le `DBClusterIdentifier` paramètre et `true` pour le `AutoMinorVersionUpgrade` paramètre. Si vous le souhaitez, définissez le paramètre `ApplyImmediately` sur `true` pour l'activer immédiatement pour votre cluster de bases de données.

Activation des mises à niveau automatiques des versions mineures pour les instances de base de données individuelles dans un cluster de bases de données Aurora

Suivez la procédure générale de la rubrique Modification d'une instance de base de données dans un cluster de bases de données.

Console

Sur la page Modifier l'instance de base de données, dans la section Maintenance, cochez la case Activer la mise à niveau automatique des versions mineures.

AWS CLI

Appelez la commande modify-db-instance AWS CLI . Spécifiez le nom de votre instance de base de données pour l'option `--db-instance-identifier` et `true` pour l'option `--auto-minor-version-upgrade`. Si vous le souhaitez, spécifiez l'option `--apply-immediately` pour activer immédiatement ce paramètre pour votre instance de base de données. Exécutez une commande `modify-db-instance` distincte pour chaque instance de base de données dans le cluster.

API RDS

Appelez l'opération DBInstanceModify API et spécifiez le nom de votre cluster de base de données pour le `DBInstanceIdentifier` paramètre et `true` pour le

`AutoMinorVersionUpgrade` paramètre. Le cas échéant, définissez le paramètre `ApplyImmediately` sur `true` pour l'activer immédiatement pour votre instance de base de données. Appelez une action `ModifyDBInstance` distincte pour chaque instance de base de données du cluster.

Vous pouvez utiliser une commande CLI telle que la suivante pour vérifier le statut du paramètre `AutoMinorVersionUpgrade` pour toutes les instances de base de données figurant dans vos clusters Aurora MySQL.

```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Le résultat de cette commande est semblable à ce qui suit :

```
[
  {
      "DBInstanceIdentifier": "db-writer-instance",
      "DBClusterIdentifier": "my-db-cluster-57",
      "AutoMinorVersionUpgrade": true
  },
  {
      "DBInstanceIdentifier": "db-reader-instance1",
      "DBClusterIdentifier": "my-db-cluster-57",
      "AutoMinorVersionUpgrade": false
  },
  {
      "DBInstanceIdentifier": "db-writer-instance2",
      "DBClusterIdentifier": "my-db-cluster-80",
      "AutoMinorVersionUpgrade": true
  },
... output omitted ...
```

Dans cet exemple, la paramètre Activer la mise à niveau automatique des versions mineures est désactivé pour le cluster de bases de données `my-db-cluster-57`, car il est désactivé pour l'une des instances de base de données du cluster.

Mises à jour du système d’exploitation pour les clusters de bases de données Aurora

Les instances de base de données des clusters de bases de données Aurora MySQL et Aurora PostgreSQL nécessitent parfois des mises à jour du système d’exploitation. Amazon RDS met à niveau le système d’exploitation vers une version plus récente afin d’améliorer les performances de la base de données et la posture de sécurité globale des clients. En général, les mises à jour prennent environ 10 minutes. Les mises à jour du système d’exploitation ne modifient pas la version du moteur de base de données ou la classe d’instance de base de données d’une instance de base de données.

Il existe deux types de mises à jour du système d’exploitation, différenciées par la description visible dans l’action de maintenance en attente sur l’instance de base de données :

- Mise à niveau de la distribution du système d’exploitation : utilisée pour migrer vers la dernière version majeure prise en charge d’Amazon Linux. Sa description est `New Operating System upgrade is available`.
- Correctif du système d’exploitation : utilisé pour appliquer divers correctifs de sécurité et parfois pour améliorer les performances de la base de données. Sa description est `New Operating System patch is available`.

Les mises à jour du système d’exploitation peuvent être facultatives ou obligatoires :

- Une mise à jour facultative peut être appliquée à tout moment. Bien que ces mises à jour soient facultatives, nous vous recommandons de les appliquer régulièrement pour que votre flotte RDS reste à jour. RDS n’applique pas ces mises à jour automatiquement.

  Pour être averti de la disponibilité d’un nouveau correctif du système d’exploitation facultatif, vous pouvez vous inscrire à RDS-EVENT-0230 dans la catégorie des événements d’application de correctifs de sécurité. Pour obtenir des informations sur l’abonnement à des événements RDS, consultez Abonnement à la notification d’événement Amazon RDS.

  > **Note**
  >
  > `RDS-EVENT-0230` ne s’applique pas aux mises à niveau de distribution du système d’exploitation.

- Une mise à jour obligatoire est requise et nous envoyons une notification avant celle-ci. La notification peut contenir une date d'échéance. Prévoyez de planifier la mise à jour avant cette date. Après la date d'échéance spécifiée, Amazon RDS met automatiquement à niveau le système d'exploitation de l'instance de base de données vers la dernière version au cours de l'une de vos fenêtres de maintenance attribuées.

  Les mises à niveau de la distribution du système d'exploitation sont obligatoires.

> **Note**
>
> Vous devrez peut-être appliquer toutes les mises à jour facultatives et obligatoires afin de respecter diverses obligations de conformité. Nous vous recommandons d'appliquer systématiquement toutes les mises à jour qui sont mises à disposition par RDS pendant vos fenêtres de maintenance.

Pour les clusters de bases de données Aurora, vous pouvez utiliser l'option de maintenance au niveau du cluster pour effectuer des mises à jour du système d'exploitation (SE). Trouvez l'option permettant d'effectuer des mises à jour au niveau du cluster dans l'onglet Maintenance et sauvegardes lorsque vous sélectionnez le nom de votre cluster de bases de données dans la console ou en utilisant la commande `os-upgrade` dans l' AWS CLI. Cette méthode préserve la disponibilité des lectures grâce à des mises à niveau progressives qui appliquent automatiquement les mises à jour à quelques instances de base de données de lecteur à la fois. Pour éviter les multiples basculements et réduire les durées d'indisponibilité inutiles, Aurora met à niveau l'instance de base de données d'enregistreur en dernier.

Les mises à jour du système d'exploitation au niveau du cluster ont lieu pendant la fenêtre de maintenance que vous avez spécifiée pour le cluster. Cela garantit des mises à jour coordonnées sur l'ensemble du cluster.

Pour des raisons de rétrocompatibilité, Aurora conserve également l'option de maintenance au niveau de l'instance. Toutefois, nous vous recommandons plutôt d'utiliser les mises à jour au niveau du cluster. Si vous devez utiliser des mises à jour au niveau de l'instance, mettez d'abord à jour les instances de base de données de lecteur dans un cluster de bases de données, puis mettez à jour l'instance de base de données d'enregistreur. Si vous mettez à jour les instances du lecteur et de l'enregistreur simultanément, vous augmentez le risque de durée d'indisponibilité liée au basculement. Trouvez l'option permettant d'effectuer des mises à jour au niveau de l'instance dans

l’onglet Maintenance et sauvegardes lorsque vous sélectionnez le nom de votre instance de base de données dans la console ou en utilisant la commande `system-update` dans l’ AWS CLI.

Les mises à jour du système d’exploitation au niveau de l’instance ont lieu pendant la fenêtre de maintenance que vous avez spécifiée pour chaque instance respective. Par exemple, si un cluster et deux instances de lecteur ont des fenêtres de maintenance différentes, une mise à jour du système d’exploitation au niveau du cluster s’aligne sur la fenêtre de maintenance du cluster.

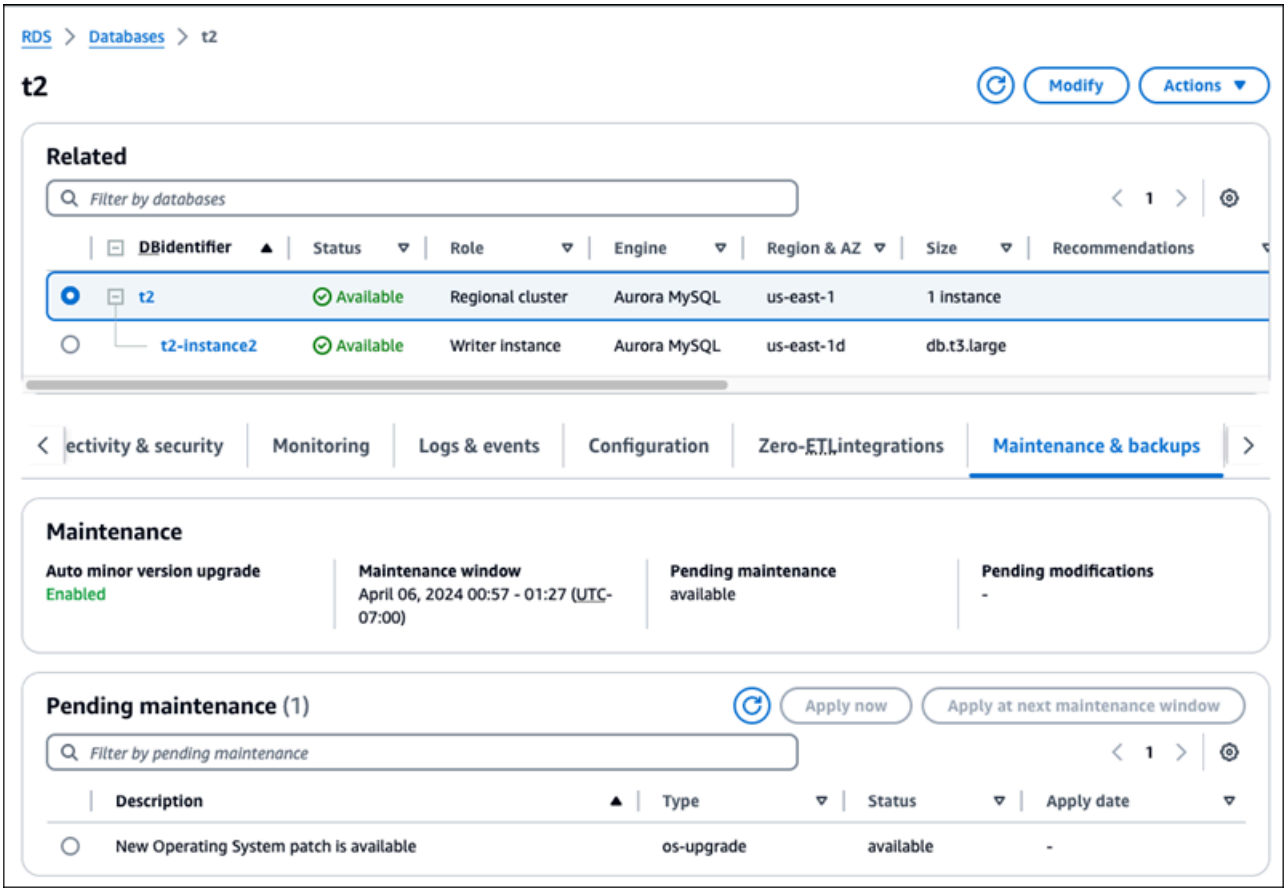
Vous pouvez utiliser le AWS Management Console ou le AWS CLI pour obtenir des informations sur le type de mise à niveau du système d'exploitation.

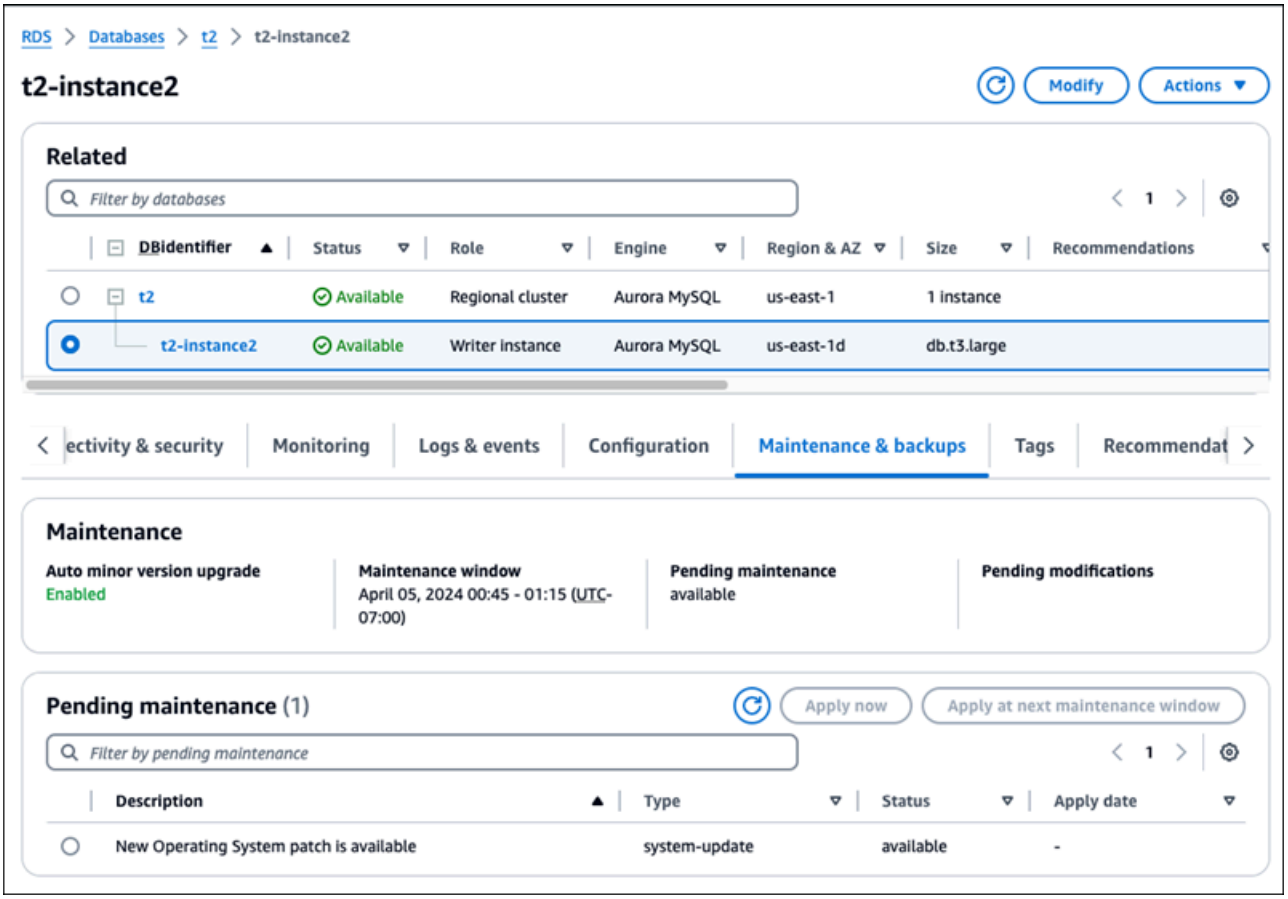
Console

Pour obtenir des informations de mise à jour à l'aide du AWS Management Console

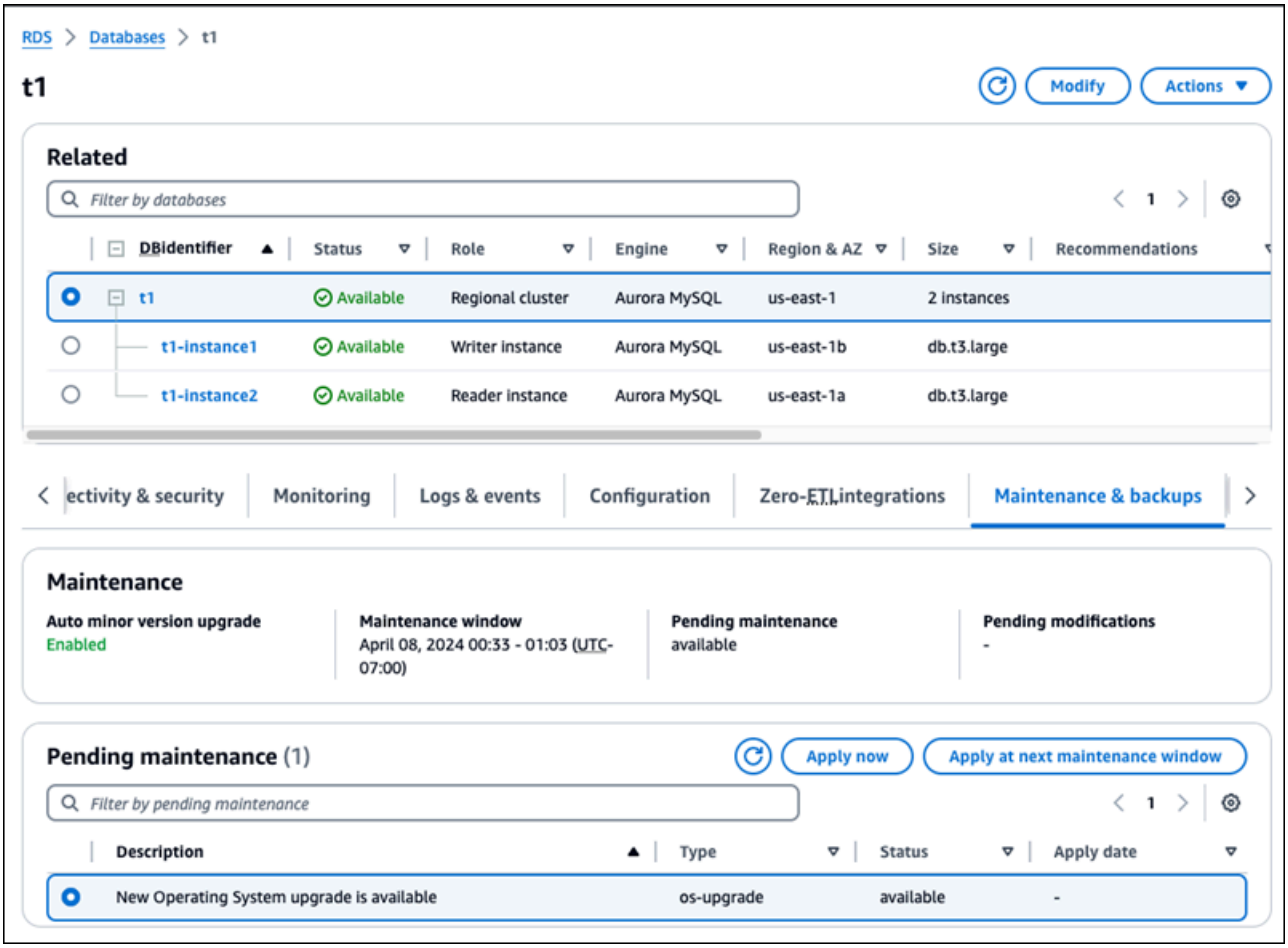
1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Bases de données, puis sélectionnez l’instance de base de données.
3. Choisissez Maintenance et sauvegardes.
4. Dans la section Maintenance en attente, recherchez la mise à jour du système d’exploitation et sélectionnez la valeur Description.

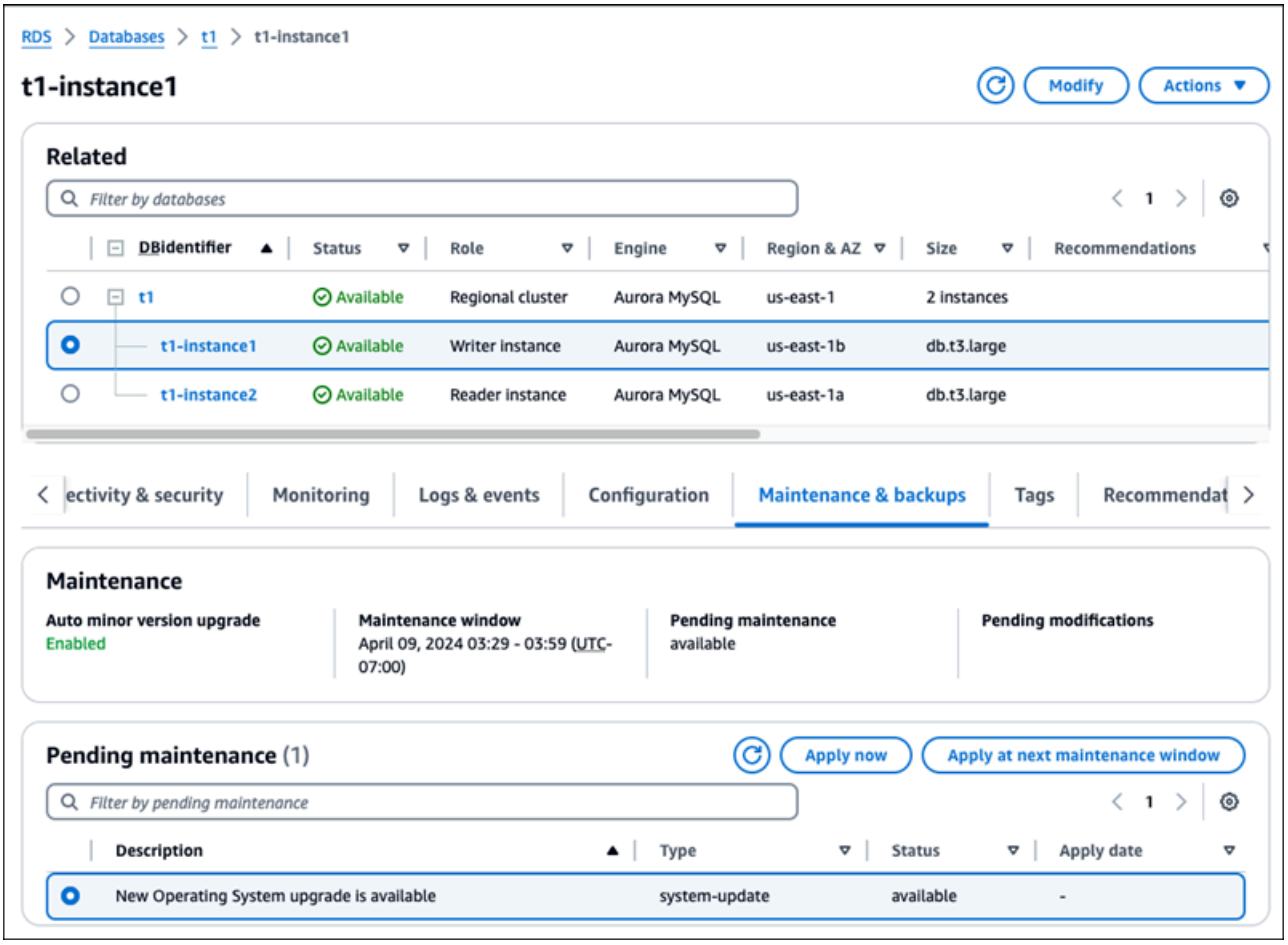
Les images suivantes montrent un cluster de bases de données avec une instance de base de données d’enregistreur pour laquelle un correctif de système d’exploitation est disponible.

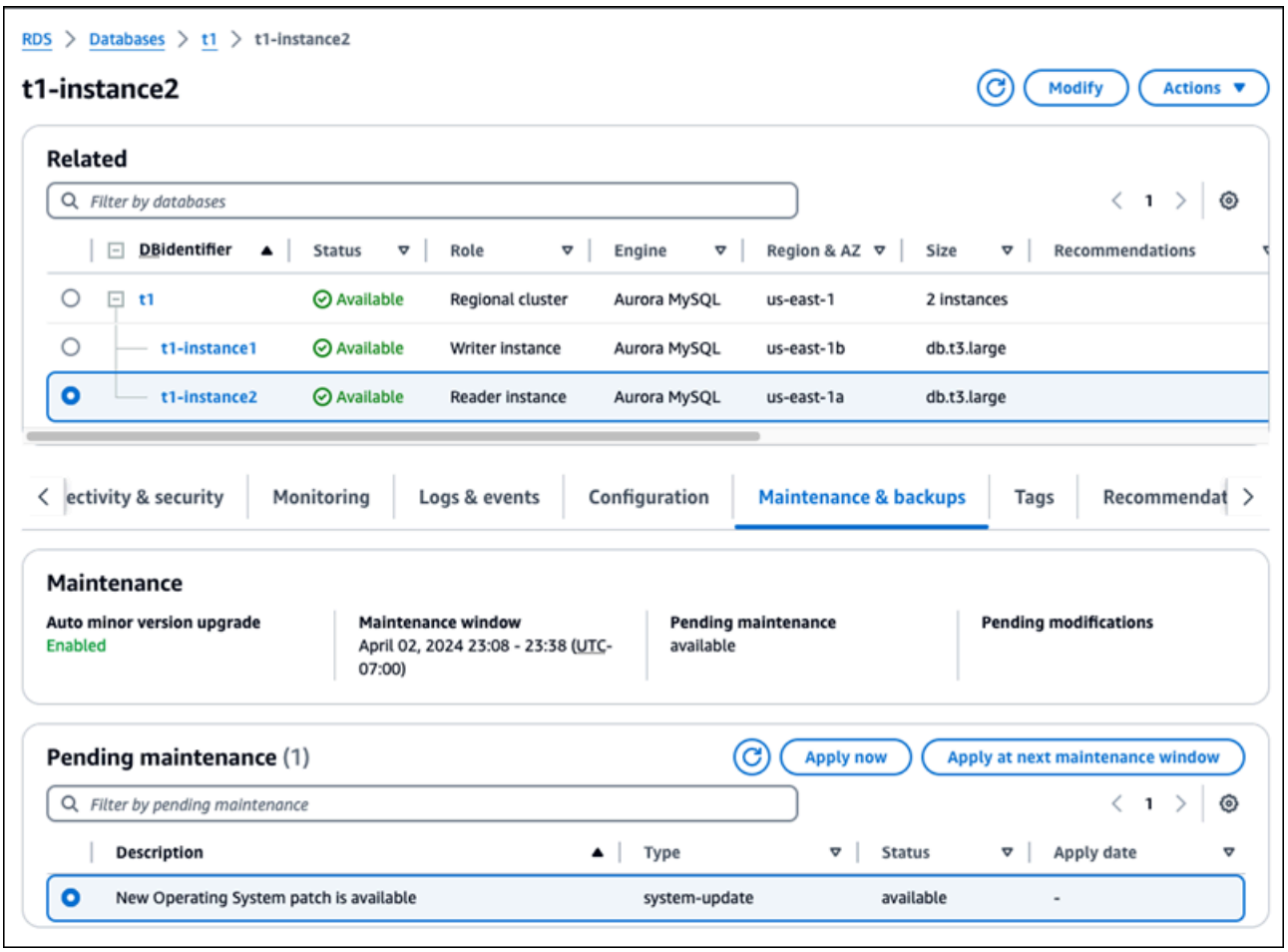

RDS > Databases > t2
t2
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t2
Available
Regional cluster
Aurora MySQL
us-east-1
1 instance
t2-instance2
Available
Writer instance
Aurora MySQL
us-east-1d
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 06, 2024 00:57 - 01:27 (UTC-07:00)
Pending maintenance
available
Pending modifications
-
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System patch is available
os-upgrade
available
-



Les images suivantes montrent un cluster de bases de données avec une instance de base de données d'enregistreur et une instance de base de données de lecteur. Une mise à niveau obligatoire du système d'exploitation est disponible pour l'instance d'enregistreur. Un correctif de système d'exploitation est disponible pour l'instance du lecteur.


RDS > Databases > t1
t1
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t1
Available
Regional cluster
Aurora MySQL
us-east-1
2 instances
t1-instance1
Available
Writer instance
Aurora MySQL
us-east-1b
db.t3.large
t1-instance2
Available
Reader instance
Aurora MySQL
us-east-1a
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Zero-ETL integrations
Maintenance & backups
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 08, 2024 00:33 - 01:03 (UTC-07:00)
Pending maintenance
available
Pending modifications
-
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System upgrade is available
os-upgrade
available
-


RDS > Databases > t1 > t1-instance1
t1-instance1
Modify
Actions
Related
Filter by databases
1
DBidentifier
Status
Role
Engine
Region & AZ
Size
Recommendations
t1
Available
Regional cluster
Aurora MySQL
us-east-1
2 instances
t1-instance1
Available
Writer instance
Aurora MySQL
us-east-1b
db.t3.large
t1-instance2
Available
Reader instance
Aurora MySQL
us-east-1a
db.t3.large
ectivity & security
Monitoring
Logs & events
Configuration
Maintenance & backups
Tags
Recommendat
Maintenance
Auto minor version upgrade
Enabled
Maintenance window
April 09, 2024 03:29 - 03:59 (UTC-07:00)
Pending maintenance
available
Pending modifications
Pending maintenance (1)
Apply now
Apply at next maintenance window
Filter by pending maintenance
1
Description
Type
Status
Apply date
New Operating System upgrade is available
system-update
available
-



AWS CLI

Pour obtenir des informations de mise à jour à partir du AWS CLI, utilisez la [describe-pending-maintenance-actions](#)commande.

```
aws rds describe-pending-maintenance-actions
```

La sortie suivante indique une mise à niveau de la distribution du système d'exploitation pour un cluster de bases de données et une instance de base de données.

```
{
  "PendingMaintenanceActions": [
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:cluster:t3",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "os-upgrade",
          "Description": "New Operating System upgrade is available"
```

```
        }
      ]
    },
    {
      "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:t3-instance1",
      "PendingMaintenanceActionDetails": [
        {
          "Action": "system-update",
          "Description": "New Operating System upgrade is available"
        }
      ]
    },
  ]
}
```

La sortie suivante indique un correctif du système d'exploitation pour une instance de base de données.

```
{
  "ResourceIdentifier": "arn:aws:rds:us-east-1:123456789012:db:mydb2",
  "PendingMaintenanceActionDetails": [
    {
      "Action": "system-update",
      "Description": "New Operating System patch is available"
    }
  ]
}
```

Disponibilité des mises à jour du système d'exploitation

Les mises à jour du système d'exploitation sont spécifiques à la version du moteur de base de données et à la classe d'instance de base de données. Par conséquent, les instances de base de données reçoivent ou requièrent des mises à jour à différents moments. Lorsqu'une mise à jour du système d'exploitation est disponible pour votre instance de base de données en fonction de sa version de moteur et de sa classe d'instance, la mise à jour apparaît dans la console. Il peut également être consulté en exécutant la describe-pending-maintenance-actions AWS CLI commande ou en appelant l'opération de l'API DescribePendingMaintenanceActionsRDS. Si une mise à jour est disponible pour votre instance, vous pouvez mettre à jour le système d'exploitation en suivant les instructions de la section Application des mises à jour à un cluster de bases de données.

Utilisation de la politique de déploiement des mises à AWS Organizations niveau pour les mises à niveau automatiques des versions mineures

Aurora prend en charge la politique de déploiement des mises à AWS Organizations niveau afin de gérer les mises à niveau automatiques des versions mineures sur plusieurs ressources de base de données et Comptes AWS. Cette politique vous aide à mettre en œuvre une stratégie de mise à niveau contrôlée pour vos clusters en :

Comment fonctionne la politique de déploiement des mises à niveau

Lorsqu'une nouvelle version mineure du moteur devient éligible à la mise à niveau automatique, la politique contrôle la séquence de mise à niveau en fonction des ordres définis :

- Les ressources marquées comme [premier] (généralement des environnements de développement) deviennent éligibles aux mises à niveau pendant leurs fenêtres de maintenance.
- Après un temps de cuisson défini, les ressources marquées comme [deuxième] deviennent éligibles.
- Après un autre temps de cuisson défini, les ressources marquées comme [last] (généralement des environnements de production) deviennent éligibles.
- Surveillance de la progression de la mise à niveau par le biais AWS de notifications Health.

Vous pouvez définir vos ordres de surclassement en :

- Politiques au niveau du compte : s'appliquent à toutes les ressources éligibles des comptes spécifiés.
- Balises de ressources : s'appliquent à des ressources spécifiques en fonction des balises.

> **Note**
>
> Les ressources non configurées avec une politique de mise à niveau ou exclues de cette politique reçoivent automatiquement un ordre de mise à niveau de [seconde].

Prérequis

- Vous Compte AWS devez faire partie d'une organisation appartenant à Organizations pour laquelle la politique de déploiement des mises à niveau est activée

- Activez les mises à niveau automatiques des versions mineures pour vos clusters
- Les balises ne sont pas strictement requises pour la politique de déploiement des mises à niveau. Si vous souhaitez définir des ordres de mise à niveau spécifiques pour différents environnements (par exemple, développement, test, assurance qualité, production), vous pouvez utiliser des balises. Si vous n'incluez pas de paramètres de balise dans votre politique, toutes les ressources soumises à cette politique suivent l'ordre de mise à niveau par défaut. Pour les ressources Aurora, seules les balises au niveau du cluster sont utilisées pour la politique de déploiement des mises à niveau, même si vous avez défini des balises au niveau de l'instance.

Pour étiqueter vos ressources

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster que vous souhaitez étiqueter.
4. Choisissez Actions, puis sélectionnez Gérer les balises.
5. Choisissez Ajouter une balise.
6. Entrez votre clé de balise (par exemple, « Environnement ») et votre valeur (par exemple, « Développement »)
7. Choisissez Ajouter une étiquette, puis Enregistrer.

Vous pouvez également ajouter des balises en utilisant AWS CLI :

```
aws rds add-tags-to-resource \
    --resource-name arn:aws:rds:region:account-number:cluster:cluster-name \
    --tags Key=Environment,Value=Development
```

Ordre et phases de mise à niveau

La politique de déploiement des mises à niveau prend en charge trois ordres de mise à niveau :

- [premier] - Généralement utilisé pour les environnements de développement ou de test
- [deuxième] - Généralement utilisé pour les environnements d'assurance qualité. Ordre par défaut des ressources si la politique n'est pas spécifiquement configurée
- [last] - Généralement réservé aux environnements de production

Lorsqu'une nouvelle version mineure du moteur devient éligible à la mise à niveau automatique :

- Les ressources ayant reçu un ordre de mise à niveau [en premier] deviennent éligibles aux mises à niveau pendant leurs fenêtres de maintenance configurées.
- Après un certain temps de cuisson, les ressources dont l'ordre de mise à niveau est [second] deviennent éligibles aux mises à niveau pendant leurs fenêtres de maintenance.
- Après un autre temps de cuisson défini, les ressources dont l'ordre de mise à niveau est [dernier] deviennent éligibles aux mises à niveau pendant leurs fenêtres de maintenance.
- La campagne de mise à niveau automatique des versions mineures prend fin une fois que toutes les ressources éligibles ayant reçu des ordres de mise à niveau [première], [deuxième] et [dernière] ont été mises à niveau, ou lorsque la campagne atteint sa date de fin prévue, selon la première éventualité.

> **Note**
>
> Toutes les mises à niveau automatiques des versions mineures sont effectuées pendant la période de maintenance configurée pour chaque cluster afin de minimiser l'impact potentiel sur vos applications.

Observabilité

AWS Santé et surveillance

Vous recevez des notifications AWS de santé :

- Avant le début d'une campagne de mise à niveau automatique d'une version mineure
- Transition entre chaque phase pour faciliter le suivi et le suivi de la progression de la mise à niveau
- Mises à jour de progression indiquant le nombre de ressources améliorées au sein de votre flotte dans la console AWS Health

Notifications d'événements Amazon RDS :

- Notifications relatives aux ressources activées pour les mises à niveau automatiques des versions mineures, notamment :
  - Quand votre ressource devient éligible à la mise à niveau en fonction de son ordre d'amélioration ([première], [deuxième] ou [dernière])

  - Chronologie de mise à niveau planifiée pendant la fenêtre de maintenance
  - État de début et d'achèvement de la mise à niveau individuelle de la base
- Abonnez-vous à ces événements via Amazon EventBridge 0 pour une surveillance automatisée

Considérations

Voici quelques points à prendre en compte :

- La politique s'applique à toutes les futures campagnes de mise à niveau automatique des versions mineures, y compris les modifications de politique effectuées pendant les campagnes actives.
- Si vous participez à une campagne de mise à niveau en cours, vos ressources suivent l'ordre de mise à niveau en cours et n'attendent pas la configuration d'une politique.
- Les ressources non configurées avec une politique de mise à niveau ou exclues de cette politique reçoivent automatiquement un ordre de mise à niveau de [seconde].
- La politique prévoit des périodes de validation entre les phases de mise à niveau avant de passer à la phase suivante.
- Les modifications apportées à la politique ou aux balises de ressources nécessitent un certain temps pour se propager avant que le nouvel ordre de mise à niveau ne soit appliqué.
- La politique s'applique uniquement aux ressources Aurora pour lesquelles les mises à niveau automatiques des versions mineures sont activées.
- Si vous détectez un problème dans un environnement, vous pouvez désactiver les mises à niveau automatiques des versions mineures pour les environnements suivants ou utiliser la période de validation pour résoudre les problèmes avant que les mises à niveau ne passent à l'ordre de mise à niveau suivant.

Pour plus d'informations sur le balisage des ressources RDS, consultez. Marquage des ressources Amazon Aurora et Amazon RDS Pour obtenir des instructions détaillées sur la configuration et l'utilisation de la politique de déploiement des mises à niveau, voir Getting started with AWS Organizations dans le guide de l'AWS Organizations utilisateur.

Redémarrage d'un cluster de bases de données Amazon Aurora ou d'une instance de base de données Amazon Aurora

Vous devrez peut-être redémarrer votre cluster de bases de données ou certaines instances du cluster, généralement pour des raisons de maintenance. Par exemple, supposons que vous modifiiez les paramètres d'un groupe de paramètres ou que vous associiez un autre groupe de paramètres à votre cluster. Dans ce cas, vous devez redémarrer le cluster pour que les modifications prennent effet. De même, vous pouvez redémarrer une ou plusieurs instance de base de données du lecteur au sein du cluster. Vous pouvez organiser les opérations de redémarrage pour des instances individuelles afin de réduire les temps d'arrêt pour l'ensemble du cluster.

Le temps nécessaire au redémarrage de chaque instance de base de données de votre cluster dépend de l'activité de la base de données au moment du redémarrage. Il dépend également du processus de récupération de votre moteur de base de données spécifique. Si c'est pratique, réduisez l'activité de la base de données sur cette instance particulière avant de démarrer le processus de redémarrage. Cela peut réduire le temps nécessaire au redémarrage de la base de données.

Vous ne pouvez redémarrer chaque instance de base de données de votre cluster que lorsque celle-ci est à l'état disponible. Une instance de base de données peut être indisponible pour plusieurs raisons. Il s'agit notamment des situations suivantes : le cluster est à d'arrêt, une modification est appliquée à l'instance et une action de fenêtre de maintenance telle qu'une mise à niveau de version se produit.

Le redémarrage d'une instance de base de données entraîne celui du processus du moteur de base de données. Le redémarrage d'une instance de bases de données entraîne une interruption momentanée, au cours de laquelle le statut de l'instance de bases de données est défini sur redémarrage.

> **Note**
>
> Si une instance de base de données n'utilise pas les dernières modifications apportées à son groupe de paramètres de base de données associé, AWS Management Console affiche le groupe de paramètres de base de données avec le statut suivant : pending-reboot. Le statut de groupe de paramètres pending-reboot n'entraîne pas de redémarrage automatique lors de la fenêtre de maintenance suivante. Pour appliquer les modifications de paramètre les plus récentes apportées à cette instance de base de données, vous devez la redémarrer

manuellement. Pour plus d'informations sur les groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora.

Rubriques

Redémarrage d’une instance de base de données au sein d’un cluster Aurora

Cette procédure est l’opération la plus importante que vous effectuez lorsque vous effectuez des redémarrages avec Aurora. La plupart des procédures de maintenance impliquent le redémarrage d’une ou de plusieurs instances de base de données Aurora dans un ordre particulier.

Console

Pour redémarrer une instance de base de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis l’instance de base de données à redémarrer.
3. Pour Actions, choisissez Redémarrer.

   La page Redémarrer l’instance de base de données s’affiche.
4. Choisissez Redémarrer pour redémarrer votre instance de base de données.

   Ou choisissez Cancel (Annuler).

AWS CLI

Pour redémarrer une instance de base de données à l'aide de l'AWS CLI, appelez la commande `reboot-db-instance`.

Example

Pour Linux, macOS ou Unix :

```
aws rds reboot-db-instance \
    --db-instance-identifier mydbinstance
```

Pour Windows :

```
aws rds reboot-db-instance ^
    --db-instance-identifier mydbinstance
```

API RDS

Pour redémarrer une instance de base de données à l'aide de l'API Amazon RDS, appelez l'opération `RebootDBInstance`.

Redémarrage d'un cluster Aurora avec disponibilité en lecture

Grâce à la fonctionnalité de disponibilité en lecture, vous pouvez redémarrer l'instance d'enregistreur de votre cluster Aurora sans redémarrer les instances de lecteur dans le cluster de bases de données principal ou secondaire. Cela peut contribuer à maintenir une haute disponibilité du cluster pour les opérations de lecture pendant que vous redémarrez l'instance d'enregistreur. Vous pouvez redémarrer les instances de lecteur plus tard, selon un calendrier qui vous convient. Par exemple, dans un cluster de production, vous pouvez redémarrer les instances de lecteur une par une, en commençant uniquement une fois le redémarrage de l'instance principale terminé. Pour chaque instance de base de données que vous redémarrez, suivez la procédure décrite dans Redémarrage d'une instance de base de données au sein d'un cluster Aurora.

La fonctionnalité de disponibilité en lecture pour les clusters de bases de données principaux est disponible dans Aurora MySQL 2.10 et versions ultérieures. La disponibilité en lecture pour les clusters de bases de données secondaires est disponible dans Aurora MySQL 3.06 et versions ultérieures.

Pour Aurora PostgreSQL, cette fonctionnalité est disponible par défaut dans les versions suivantes :

- 15.2 et versions 15 ultérieures
- Versions 14.7 et 14 ultérieures
- Versions 13.10 et 13 ultérieures
- 12.14 et versions 12 ultérieures

Pour plus d'informations sur la fonctionnalité de disponibilité en lecture dans Aurora PostgreSQL, consultez Amélioration de la disponibilité en lecture des réplicas Aurora.

Avant cette fonctionnalité, le redémarrage de l'instance principale entraînait un redémarrage de chaque instance de lecteur au même moment. Si votre cluster Aurora exécute une version plus ancienne, utilisez plutôt la procédure de redémarrage dans Redémarrage d'un cluster Aurora sans disponibilité en lecture.

> **Note**
>
> La modification du comportement de redémarrage dans le cluster de bases de données Aurora avec disponibilité en lecture est différente pour les bases de données globales Aurora dans les versions d'Aurora MySQL antérieures à 3.06. Si vous redémarrez l'instance d'enregistreur pour le cluster principal dans une base de données globale Aurora, les instances de lecteur du cluster principal restent disponibles. Toutefois, les instances de base de données de tous les clusters secondaires redémarrent en même temps.
> Une version limitée de la fonctionnalité améliorée de disponibilité en lecture est prise en charge par les bases de données globales Aurora pour Aurora PostgreSQL 12.16, 13.12, 14.9, 15.4 et versions ultérieures.

Vous redémarrez fréquemment le cluster après avoir apporté des modifications aux groupes de paramètres du cluster. Vous modifiez les paramètres en suivant les procédures décrites dans Groupes de paramètres pour Amazon Aurora. Supposons que vous redémarriez l'instance de base de données d'enregistreur dans un cluster Aurora pour appliquer des modifications aux paramètres du cluster. Certaines ou toutes les instances de base de données de lecteur peuvent continuer à utiliser les anciens paramètres. Toutefois, les différents paramètres de paramètres n'affectent pas l'intégrité des données du cluster. Tous les paramètres de cluster qui affectent l'organisation des fichiers de données sont uniquement utilisés par l'instance de base de données d'enregistreur.

Par exemple, dans un cluster Aurora MySQL, vous pouvez mettre à jour des paramètres de cluster tels que `binlog_format` et `innodb_purge_threads` sur l'instance d'enregistreur

avant les instances de lecteur. Seule l'instance d'enregistreur écrit des journaux binaires et purge les enregistrements d'annulation. Pour les paramètres qui modifient la façon dont les requêtes interprètent les instructions SQL ou la sortie de requête, vous devrez peut-être redémarrer immédiatement les instances de lecteur. Vous procédez de cette façon pour éviter tout comportement inattendu de l'application lors des requêtes. Par exemple, supposons que vous modifiiez le paramètre `lower_case_table_names` et que vous redémarriez l'instance d'enregistreur. Dans ce cas, il se peut que les instances de lecteur ne puissent pas accéder à une table récemment créée tant qu'elles n'ont pas toutes été redémarrées.

Pour obtenir la liste de tous les paramètres du cluster Aurora MySQL, consultez Paramètres de niveau cluster.

Pour obtenir la liste de tous les paramètres de cluster Aurora PostgreSQL, consultez Paramètres de niveau cluster d'Aurora PostgreSQL.

> **Tip**
>
> Aurora MySQL peut encore redémarrer certaines instances de lecteur avec l'instance d'enregistreur si votre cluster traite une application à haut débit.
> La réduction du nombre de redémarrages s'applique également lors des opérations de basculement. Lors d'un basculement, Aurora MySQL redémarre uniquement l'instance de base de données du scripteur et la cible de basculement. D'autres instances de base de données de lecteur dans le cluster restent disponibles pour continuer le traitement des requêtes par le biais de connexions au point de terminaison du lecteur. Vous pouvez donc améliorer la disponibilité lors d'un basculement en disposant de plusieurs instances de base de données de lecteur dans un cluster.

Redémarrage d'un cluster Aurora sans disponibilité en lecture

Grâce à la fonctionnalité de disponibilité en lecture, vous redémarrez un cluster de bases de données Aurora complet en redémarrant l'instance de base de données d'enregistreur de ce cluster. Pour ce faire, suivez la procédure décrite dans Redémarrage d'une instance de base de données au sein d'un cluster Aurora.

Le redémarrage de l'instance de base de données d'enregistreur déclenche également un redémarrage pour chaque instance de base de données d'enregistreur dans le cluster. De cette façon, tous les changements de paramètres à l'échelle du cluster sont appliqués simultanément à toutes les instances de base de données. Toutefois, le redémarrage de toutes les instances de

base de données entraîne une brève interruption du cluster. Les instances de base de données de lecteur restent indisponibles jusqu'à ce que l'instance de base de données d'enregistreur ait fini de redémarrer et soit disponible.

Ce comportement de redémarrage s'applique à tous les clusters de bases de données créés dans Aurora MySQL version 2.09 et antérieures.

Pour Aurora PostgreSQL, ce comportement s'applique aux versions suivantes :

- Version 14.6 et versions 14 antérieures
- Version 13.9 et versions 13 antérieures
- Version 12.13 et versions 12 antérieures
- Toutes les versions de PostgreSQL 11

Dans la console RDS, l'instance de base de données d'enregistreur a la valeur Writer (Enregistreur) sous la colonneRole (Rôle) de la page Databases (Bases de données). Dans l'interface de ligne de commande CLI RDS, la sortie de la commande `describe-db-clusters` comprend une section `DBClusterMembers`. L'élément `DBClusterMembers` représentant l'instance de base de données d'enregistreur a la valeur `true` du champ `IsClusterWriter`.

> **Important**
>
> Avec la fonctionnalité de disponibilité en lecture, le comportement de redémarrage est différent dans Aurora MySQL et Aurora PostgreSQL : les instances de base de données de lecteur restent généralement disponibles pendant que vous redémarrez l'instance d'enregistreur. Vous pouvez ensuite redémarrer les instances de lecteur à un moment opportun. Vous pouvez redémarrer les instances de lecteur selon un calendrier échelonné si vous souhaitez que certaines de ses instances soient toujours disponibles. Pour de plus amples informations, consultez Redémarrage d’un cluster Aurora avec disponibilité en lecture.

Vérification de la disponibilité des clusters et des instances Aurora

Vous pouvez vérifier et surveiller la durée écoulé depuis le dernier redémarrage de chaque instance de base de données de votre cluster Aurora. La CloudWatch métrique Amazon `EngineUptime` indique le nombre de secondes écoulées depuis le dernier démarrage d'une instance de base de

données. Vous pouvez examiner cette métrique à un moment donné pour connaître le temps de disponibilité de l'instance de base de données. Vous pouvez également contrôler cette métrique au fil du temps pour détecter le moment où l'instance est redémarrée.

Vous pouvez également examiner la métrique `EngineUptime` au niveau du cluster. Les dimensions `Minimum` et `Maximum` indiquent les valeurs de disponibilité les plus petites et les plus importantes pour toutes les instances de base de données du cluster. Pour vérifier le moment le plus récent où une instance du lecteur d'un cluster a été redémarrée, ou a été redémarrée pour une autre raison, contrôlez la métrique au niveau du cluster à l'aide de la dimension `Minimum`. Pour vérifier quelle instance du cluster est restée le plus longtemps sans redémarrage, contrôlez la métrique au niveau du cluster à l'aide de la dimension `Maximum`. Par exemple, vous pouvez confirmer que toutes les instances de base de données du cluster ont été redémarrées après une modification de la configuration.

> **Tip**
>
> Pour la surveillance à long terme, nous recommandons de contrôler la métrique `EngineUptime` pour des instances individuelles, plutôt qu'au niveau du cluster. La métrique `EngineUptime` au niveau du cluster est définie sur zéro lorsqu'une nouvelle instance de base de données est ajoutée au cluster. Ces modifications de cluster peuvent se produire dans le cadre d'opérations de maintenance et de mise à l'échelle, telles que celles effectuées par Auto Scaling.

Les exemples d'interface de ligne de commande CLI suivants montrent comment examiner la métrique `EngineUptime` pour les instances d'enregistreur et de lecteur dans un cluster. Les exemples utilisent un cluster nommé `tpch100g`. Ce cluster possède une instance de base de données d'enregistreur `instance-1234`. Il dispose également de deux instances de base de données de lecteur, `instance-7448` et`instance-6305`.

Tout d'abord, la commande `reboot-db-instance` redémarre l'une des instances de lecteur. La commande `wait` attend que le redémarrage de l'instance soit terminé.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6305
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-6305",
        "DBInstanceStatus": "rebooting",
...
```

```
$ aws rds wait db-instance-available --db-instance-id instance-6305
```

La CloudWatch `get-metric-statistics` commande examine la `EngineUptime` métrique au cours des cinq dernières minutes à intervalles d'une minute. Le temps de disponibilité de l'instance `instance-6305` est remis à zéro et recommence un compte à rebours. Cet AWS CLI exemple pour Linux utilise la substitution de `$()` variables pour insérer les horodatages appropriés dans les commandes de la CLI. Il utilise également la commande `sort` Linux pour ordonner la sortie au moment où la métrique a été collectée. Cette valeur d'horodatage est le troisième champ de chaque ligne de sortie.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-6305 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 231.0 2021-03-16T18:19:00+00:00 Seconds
DATAPOINTS 291.0 2021-03-16T18:20:00+00:00 Seconds
DATAPOINTS 351.0 2021-03-16T18:21:00+00:00 Seconds
DATAPOINTS 411.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 471.0 2021-03-16T18:23:00+00:00 Seconds
```

La durée de disponibilité minimale du cluster est remise à zéro, car l'une des instances du cluster a été redémarrée. La durée de disponibilité maximale du cluster n'est pas réinitialisée, car au moins une des instances de base de données du cluster est restée disponible.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Minimum \
  --dimensions Name=DBClusterIdentifier,Value=tpch100g --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63099.0 2021-03-16T18:12:00+00:00 Seconds
DATAPOINTS 63159.0 2021-03-16T18:13:00+00:00 Seconds
DATAPOINTS 63219.0 2021-03-16T18:14:00+00:00 Seconds
DATAPOINTS 63279.0 2021-03-16T18:15:00+00:00 Seconds
DATAPOINTS 51.0 2021-03-16T18:16:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
```

```
  --dimensions Name=DBClusterIdentifier,Value=tpch100g --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63389.0 2021-03-16T18:16:00+00:00 Seconds
DATAPOINTS 63449.0 2021-03-16T18:17:00+00:00 Seconds
DATAPOINTS 63509.0 2021-03-16T18:18:00+00:00 Seconds
DATAPOINTS 63569.0 2021-03-16T18:19:00+00:00 Seconds
DATAPOINTS 63629.0 2021-03-16T18:20:00+00:00 Seconds
```

Ensuite, une autre commande `reboot-db-instance` redémarre l'instance d'enregistreur du cluster. Une autre commande `wait` s'arrête jusqu'à ce que l'instance d'enregistreur ait terminé le redémarrage.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-1234
{
  "DBInstanceIdentifier": "instance-1234",
  "DBInstanceStatus": "rebooting",
...
$ aws rds wait db-instance-available --db-instance-id instance-1234
```

La métrique `EngineUptime`de l'instance d'enregistreur indique désormais que l'instance `instance-1234` a été redémarrée récemment. L'instance de lecteur `instance-6305` a également été redémarrée automatiquement avec l'instance d'enregistreur. Ce cluster exécute Aurora MySQL 2.09, ce qui ne permet pas de maintenir les instances de lecteur en cours d'exécution au redémarrage de l'instance d'enregistreur.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-1234 --output text \
  | sort -k 3
EngineUptime
DATAPOINTS 63749.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 63809.0 2021-03-16T18:23:00+00:00 Seconds
DATAPOINTS 63869.0 2021-03-16T18:24:00+00:00 Seconds
DATAPOINTS 41.0 2021-03-16T18:25:00+00:00 Seconds
DATAPOINTS 101.0 2021-03-16T18:26:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" \
  --period 60 --namespace "AWS/RDS" --statistics Maximum \
  --dimensions Name=DBInstanceIdentifier,Value=instance-6305 --output text \
```

```
  | sort -k 3
EngineUptime
DATAPOINTS 411.0 2021-03-16T18:22:00+00:00 Seconds
DATAPOINTS 471.0 2021-03-16T18:23:00+00:00 Seconds
DATAPOINTS 531.0 2021-03-16T18:24:00+00:00 Seconds
DATAPOINTS 49.0 2021-03-16T18:26:00+00:00 Seconds
```

Exemples d'opérations de redémarrage Aurora

Les exemples Aurora MySQL suivants illustrent différentes combinaisons d'opérations de redémarrage pour les instances de base de données de lecteur et d'enregistreur dans un cluster de bases de données Aurora. Après chaque redémarrage, les requêtes SQL indiquent le temps de disponibilité des instances du cluster.

Rubriques

- Recherche des instances d'enregistreur et de lecteur pour un cluster Aurora
- Redémarrage d'une instance de lecteur unique
- Redémarrage de l'instance d'enregistreur
- Redémarrage indépendant de l'enregistreur et des lecteurs
- Application d'une modification de paramètre de cluster à un cluster Aurora MySQL version 2.10

Recherche des instances d'enregistreur et de lecteur pour un cluster Aurora

Dans un cluster Aurora MySQL comportant plusieurs instances de base de données, il est important de savoir laquelle est l'enregistreur et lesquelles sont les lecteurs. Les instances d'enregistreur et de lecteur peuvent également changer de rôle lorsqu'une opération de basculement se produit. Il est donc préférable d'effectuer une vérification telle que la suivante avant d'effectuer toute opération nécessitant une instance d'enregistreur ou de lecteur. Dans ce cas, les valeurs `False` pour `IsClusterWriter` identifient les instances de lecteur `instance-6305` et`instance-7448`. La valeur `True` identifie l'instance d'enregistreur `instance-1234`.

```
$ aws rds describe-db-clusters --db-cluster-id tpch100g \
  --query "*[].['Cluster:',DBClusterIdentifier,DBClusterMembers[*].
['Instance:',DBInstanceIdentifier,IsClusterWriter]]" \
  --output text
Cluster:     tpch100g
Instance:    instance-6305    False
Instance:    instance-7448    False
```

```
Instance:    instance-1234    True
```

Avant d'entamer les exemples de redémarrage, l'instance d'enregistreur a un temps de disponibilité d'environ une semaine. La requête SQL de cet exemple montre un moyen spécifique à MySQL de vérifier la disponibilité. Vous pouvez utiliser cette technique dans une application de base de données. Pour une autre technique qui utilise l'AWS CLI et fonctionne pour les deux moteurs Aurora, consultez Vérification de la disponibilité des clusters et des instances Aurora.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status
    -> where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-08 17:49:06.000000 | 174h 42m|
+----------------------------+---------+
```

Redémarrage d'une instance de lecteur unique

Cet exemple montre comment redémarrer l'une des instances de base de données de lecteur. Peut-être que cette instance a été surchargée par une requête trop importante ou par de nombreuses connexions simultanées. Elle est peut-être également restée derrière l'instance d'enregistreur en raison d'un problème de réseau. Après le lancement de l'opération de redémarrage, l'exemple utilise une commande `wait` pour effectuer une mise en pause jusqu'à ce que l'instance soit disponible. À ce moment-là, le temps de disponibilité de l'instance est de quelques minutes.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6305
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-6305",
        "DBInstanceStatus": "rebooting",
...
    }
}
$ aws rds wait db-instance-available --db-instance-id instance-6305
$ mysql -h instance-6305.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
```

```
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status
    -> where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:35:02.000000 | 00h 03m |
+----------------------------+---------+
```

Le redémarrage de l'instance de lecteur n'a pas affecté le temps de disponibilité de l'instance d'enregistreur. Elle a encore un temps de disponibilité d'environ une semaine.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+----------+
| Last Startup               | Uptime   |
+----------------------------+----------+
| 2021-03-08 17:49:06.000000 | 174h 49m |
+----------------------------+----------+
```

Redémarrage de l'instance d'enregistreur

Cet exemple montre comment redémarrer l'instance d'enregistreur. Ce cluster exécute Aurora MySQL version 2.09. Étant donné que la version de Aurora MySQL est inférieure à la version 2.10, le redémarrage de l'instance d'enregistreur redémarre également toutes les instances de lecteur du cluster.

Une commande `wait` s'arrête jusqu'à ce que le redémarrage soit terminé. Le temps de disponibilité de cette instance est maintenant remis à zéro. Il est possible qu'une opération de redémarrage prenne des temps sensiblement différents pour les instances de base de données d'enregistreur et de lecteur. Les instances de base de données d'enregistreur et de lecteur effectuent différents types d'opérations de nettoyage en fonction de leurs rôles.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-1234
{
    "DBInstance": {
        "DBInstanceIdentifier": "instance-1234",
        "DBInstanceStatus": "rebooting",
```

```
...
    }
}
$ aws rds wait db-instance-available --db-instance-id instance-1234
$ mysql -h instance-1234.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:27.000000 | 00h 00m |
+----------------------------+---------+
```

Après le redémarrage de l'instance de base de données d'enregistreur, les deux instances de base de données de lecteur ont également réinitialisé leur temps de disponibilité. Le redémarrage de l'instance d'enregistreur a également provoqué le redémarrage des instances de lecteur. Ce comportement s'applique aux clusters Aurora PostgreSQL et aux clusters Aurora MySQL antérieurs à la version 2.10.

```
$ mysql -h instance-7448.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:35.000000 | 00h 00m |
+----------------------------+---------+

$ mysql -h instance-6305.a12345.us-east-1.rds.amazonaws.com -P 3306 -u my-user -p
...
mysql> select date_sub(now(), interval variable_value second) "Last Startup",
    -> time_format(sec_to_time(variable_value),'%Hh %im') as "Uptime"
    -> from performance_schema.global_status where variable_name='Uptime';
+----------------------------+---------+
| Last Startup               | Uptime  |
+----------------------------+---------+
| 2021-03-16 00:40:33.000000 | 00h 01m |
+----------------------------+---------+
```

Redémarrage indépendant de l'enregistreur et des lecteurs

Les exemples suivants montrent un cluster exécutant Aurora MySQL version 2.10. Dans cette version d'Aurora MySQL et les versions ultérieures, vous pouvez redémarrer l'instance d'enregistreur sans provoquer de redémarrage pour toutes les instances de lecteur. De cette façon, vos applications exigeantes en requêtes ne subissent aucune panne lorsque vous redémarrez l'instance d'enregistreur. Vous pouvez redémarrer les instances de lecteur ultérieurement. Vous pouvez effectuer ces redémarrages à un moment où le trafic de requêtes est faible. Vous pouvez également redémarrer les instances de lecteur une par une. De cette façon, au moins une instance de lecteur est toujours disponible pour le trafic de requêtes de votre application.

L'exemple suivant utilise un cluster nommé `cluster-2393` exécutant Aurora MySQL version `5.7.mysql_aurora.2.10.0`. Ce cluster possède une instance d'enregistreur nommée `instance-9404` et trois instances de lecteur nommées `instance-6772` `instance-2470` et `instance-5138`.

```
$ aws rds describe-db-clusters --db-cluster-id cluster-2393 \
  --query "*[].['Cluster:',DBClusterIdentifier,DBClusterMembers[*].
['Instance:',DBInstanceIdentifier,IsClusterWriter]]" \
  --output text
Cluster:        cluster-2393
Instance:       instance-5138        False
Instance:       instance-2470        False
Instance:       instance-6772        False
Instance:       instance-9404        True
```

La vérification de la valeur `uptime` de chaque instance de base de données à l'aide de la commande `mysql` montre que chacune a à peu près le même temps de disponibilité. Par exemple, voici le temps de disponibilité pour `instance-5138`.

```
mysql> SHOW GLOBAL STATUS LIKE 'uptime';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Uptime        | 3866  |
+---------------+-------+
```

En utilisant CloudWatch, nous pouvons obtenir les informations sur le temps de disponibilité correspondantes sans se connecter aux instances. De cette façon, un administrateur peut contrôler la base de données, mais il ne peut ni afficher ni modifier aucune donnée de table. Dans ce cas, nous

spécifions une période de cinq minutes et nous vérifions la valeur du temps de disponibilité à chaque minute. Les valeurs de temps de disponibilité croissantes démontrent que les instances n'ont pas été redémarrées au cours de cette période.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4648.0 2021-03-17T23:42:00+00:00 Seconds
DATAPOINTS 4708.0 2021-03-17T23:43:00+00:00 Seconds
DATAPOINTS 4768.0 2021-03-17T23:44:00+00:00 Seconds
DATAPOINTS 4828.0 2021-03-17T23:45:00+00:00 Seconds
DATAPOINTS 4888.0 2021-03-17T23:46:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-6772 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4315.0 2021-03-17T23:42:00+00:00 Seconds
DATAPOINTS 4375.0 2021-03-17T23:43:00+00:00 Seconds
DATAPOINTS 4435.0 2021-03-17T23:44:00+00:00 Seconds
DATAPOINTS 4495.0 2021-03-17T23:45:00+00:00 Seconds
DATAPOINTS 4555.0 2021-03-17T23:46:00+00:00 Seconds
```

Maintenant, nous redémarrons l'une des instances de lecteu, `instance-5138`. Nous attendons que l'instance soit de nouveau disponible après le redémarrage. À présent, la surveillance du temps de disponibilité sur une période de cinq minutes montre que ce dernier a été remis à zéro pendant cette période. La valeur de disponibilité la plus récente a été mesurée cinq secondes après la fin du redémarrage.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-5138

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
```

```
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-5138 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 4500.0 2021-03-17T23:46:00+00:00 Seconds
DATAPOINTS 4560.0 2021-03-17T23:47:00+00:00 Seconds
DATAPOINTS 4620.0 2021-03-17T23:48:00+00:00 Seconds
DATAPOINTS 4680.0 2021-03-17T23:49:00+00:00 Seconds
DATAPOINTS  5.0 2021-03-17T23:50:00+00:00 Seconds
```

Ensuite, nous effectuons un redémarrage pour l'instance d'enregistreur, `instance-9404`. Nous comparons les valeurs de temps de disponibilité de l'instance d'enregistreur et de l'une des instances de lecteur. Ce faisant, nous pouvons constater que le redémarrage de l'enregistreur n'a pas provoqué de redémarrage pour les lecteurs. Dans les versions antérieures à la version Aurora MySQL 2.10, les valeurs de temps de disponibilité de tous les lecteurs seraient réinitialisées en même temps que pour l'enregistreur.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-9404
{
  "DBInstanceIdentifier": "instance-9404",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-9404

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 371.0 2021-03-17T23:57:00+00:00 Seconds
DATAPOINTS 431.0 2021-03-17T23:58:00+00:00 Seconds
DATAPOINTS 491.0 2021-03-17T23:59:00+00:00 Seconds
DATAPOINTS 551.0 2021-03-18T00:00:00+00:00 Seconds
DATAPOINTS  37.0  2021-03-18T00:01:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-6772 \
  --output text | sort -k 3
```

```
EngineUptime
DATAPOINTS 5215.0 2021-03-17T23:57:00+00:00 Seconds
DATAPOINTS 5275.0 2021-03-17T23:58:00+00:00 Seconds
DATAPOINTS 5335.0 2021-03-17T23:59:00+00:00 Seconds
DATAPOINTS 5395.0 2021-03-18T00:00:00+00:00 Seconds
DATAPOINTS 5455.0 2021-03-18T00:01:00+00:00 Seconds
```

Pour vous assurer que toutes les instances de lecteur ont toutes les mêmes modifications apportées aux paramètres de configuration que l'instance d'enregistreur, redémarrez toutes les instances de lecteur après l'enregistreur. Cet exemple redémarre tous les lecteurs, puis attend qu'ils soient tous disponibles avant de continuer.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-6772
{
  "DBInstanceIdentifier": "instance-6772",
  "DBInstanceStatus": "rebooting"
}

$ aws rds reboot-db-instance --db-instance-identifier instance-2470
{
  "DBInstanceIdentifier": "instance-2470",
  "DBInstanceStatus": "rebooting"
}

$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}

$ aws rds wait db-instance-available --db-instance-id instance-6772
$ aws rds wait db-instance-available --db-instance-id instance-2470
$ aws rds wait db-instance-available --db-instance-id instance-5138
```

Nous pouvons maintenant constater que l'instance de base de données d'enregistreur a le temps de disponibilité le plus élevé. La valeur de temps de disponibilité de cette instance a augmenté régulièrement tout au long de la période de surveillance. Les instances de base de données du lecteur ont toutes été redémarrées après le lecteur. Nous pouvons voir le moment où chaque lecteur a été redémarré et où son temps de disponibilité a été remis à zéro pendant la période de surveillance.

```
$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-9404 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 457.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS 517.0 2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 577.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS 637.0 2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS  697.0 2021-03-18T00:12:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-2470 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 5819.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS  35.0  2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 95.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS 155.0 2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS 215.0 2021-03-18T00:12:00+00:00 Seconds

$ aws cloudwatch get-metric-statistics --metric-name "EngineUptime" \
  --start-time "$(date -d '5 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --namespace "AWS/RDS" --statistics Minimum --dimensions
 Name=DBInstanceIdentifier,Value=instance-5138 \
  --output text | sort -k 3
EngineUptime
DATAPOINTS 1085.0 2021-03-18T00:08:00+00:00 Seconds
DATAPOINTS 1145.0 2021-03-18T00:09:00+00:00 Seconds
DATAPOINTS 1205.0 2021-03-18T00:10:00+00:00 Seconds
DATAPOINTS  49.0  2021-03-18T00:11:00+00:00 Seconds
DATAPOINTS 109.0 2021-03-18T00:12:00+00:00 Seconds
```

Application d'une modification de paramètre de cluster à un cluster Aurora MySQL version 2.10

L'exemple suivant montre comment appliquer une modification de paramètre à toutes les instances de base de données de votre cluster Aurora MySQL 2.10. Avec cette version d'Aurora MySQL, vous redémarrez indépendamment l'instance d'enregistreur et toutes les instances de lecteur.

L'exemple utilise le paramètre de configuration de MySQL `lower_case_table_names` à titre d'illustration. Lorsque ce paramètre est différent entre les instances de base de données d'enregistreur et de lecteur, il se peut qu'une requête ne soit pas en mesure d'accéder à une table déclarée avec un nom en majuscules ou à casse mixte. Ou, si deux noms de table ne diffèrent que par des majuscules et des minuscules, une requête peut accéder à la mauvaise table.

Cet exemple montre comment déterminer les instances d'enregistreur et de lecteur dans le cluster en examinant l'attribut `IsClusterWriter` de chaque instance. Le cluster se nomme `cluster-2393`. Le cluster possède une instance d'enregistreur nommée `instance-9404`. Les instances de lecteur du cluster sont nommées `instance-5138` et `instance-2470`.

```
$ aws rds describe-db-clusters --db-cluster-id cluster-2393 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' \
  --output text
cluster-2393
instance-5138        False
instance-2470        False
instance-9404        True
```

Pour démontrer les effets de la modification du paramètre `lower_case_table_names`, nous avons configuré deux groupes de paramètres de cluster de bases de données. Ce paramètre est défini sur 0 pour le groupe de paramètres `lower-case-table-names-0`. Le groupe de paramètres `lower-case-table-names-1` est défini sur 1.

```
$ aws rds create-db-cluster-parameter-group --description 'lower-case-table-names-0' \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-cluster-parameter-group-name lower-case-table-names-0
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "lower-case-table-names-0",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "lower-case-table-names-0"
    }
}

$ aws rds create-db-cluster-parameter-group --description 'lower-case-table-names-1' \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-cluster-parameter-group-name lower-case-table-names-1
{
    "DBClusterParameterGroup": {
```

```
        "DBClusterParameterGroupName": "lower-case-table-names-1",
        "DBParameterGroupFamily": "aurora-mysql5.7",
        "Description": "lower-case-table-names-1"
    }
}

$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lower-case-table-names-0 \
  --parameters
 ParameterName=lower_case_table_names,ParameterValue=0,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "lower-case-table-names-0"
}

$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lower-case-table-names-1 \
    --parameters
 ParameterName=lower_case_table_names,ParameterValue=1,ApplyMethod=pending-reboot
{
    "DBClusterParameterGroupName": "lower-case-table-names-1"
}
```

La valeur par défaut de `lower_case_table_names` est 0. Avec ce paramètre, la table `foo` est distincte de la table F00. Cet exemple vérifie que le paramètre est toujours à sa valeur par défaut. Ensuite, l'exemple crée trois tables qui ne diffèrent que par des lettres majuscules et minuscules dans leur nom.

```
mysql> create database lctn;
Query OK, 1 row affected (0.07 sec)

mysql> use lctn;
Database changed
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        0 |
+--------------------------+

mysql> create table foo (s varchar(128));
mysql> insert into foo values ('Lowercase table name foo');

mysql> create table Foo (s varchar(128));
```

```
mysql> insert into Foo values ('Mixed-case table name Foo');

mysql> create table FOO (s varchar(128));
mysql> insert into FOO values ('Uppercase table name FOO');

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                         |
+--------------------------+
| Mixed-case table name Foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Uppercase table name FOO |
+--------------------------+
```

Ensuite, nous associons le groupe de paramètres de base de données au cluster pour définir le paramètre `lower_case_table_names` sur 1. Cette modification ne prend effet qu'après le redémarrage de chaque instance de base de données.

```
$ aws rds modify-db-cluster --db-cluster-identifier cluster-2393 \
  --db-cluster-parameter-group-name lower-case-table-names-1
{
  "DBClusterIdentifier": "cluster-2393",
  "DBClusterParameterGroup": "lower-case-table-names-1",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.10.0"
}
```

Le premier redémarrage que nous effectuons concerne l'instance de base de données d'enregistreur. Ensuite, nous attendons que l'instance soit de nouveau disponible. À ce stade, nous nous connectons au point de terminaison de l'enregistreur et vérifions que l'instance d'enregistreur

présente la valeur du paramètre modifiée. La commande `SHOW TABLES` confirme que la base de données contient les trois tables différentes. Toutefois, les requêtes qui font référence à des tables nommées `foo`, `Foo` ou `F00` accèdent toutes à la table dont le nom est entièrement en minuscules, `foo`.

```
# Rebooting the writer instance
$ aws rds reboot-db-instance --db-instance-identifier instance-9404
$ aws rds wait db-instance-available --db-instance-id instance-9404
```

Désormais, les requêtes utilisant le point de terminaison du cluster montrent les effets de la modification de paramètre. Que le nom de table de la requête soit en majuscules, en minuscules ou en casse mixte, l'instruction SQL accède à la table dont le nom est entièrement en minuscules.

```
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        1 |
+--------------------------+

mysql> use lctn;
mysql> show tables;
+----------------+
| Tables_in_lctn |
+----------------+
| FOO            |
| Foo            |
| foo            |
+----------------+

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
```

```
+--------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+
```

L'exemple suivant montre les mêmes requêtes que le précédent. Dans ce cas, les requêtes utilisent le point de terminaison du lecteur et s'exécutent sur l'une des instances de base de données de lecteur. Ces instances n'ont pas encore été redémarrées. Ainsi, elles ont toujours le réglage d'origine du paramètre `lower_case_table_names`. Cela signifie que les requêtes peuvent accéder à chacune des tables `foo`, `Foo` et `FOO`.

```
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        0 |
+--------------------------+

mysql> use lctn;

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+---------------------------+
| s                         |
+---------------------------+
| Mixed-case table name Foo |
+---------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Uppercase table name FOO |
```

```
+--------------------------+
```

Ensuite, nous redémarrons l'une des instances de lecteur et nous attendons qu'elle soit à nouveau disponible.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-2470
{
  "DBInstanceIdentifier": "instance-2470",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-2470
```

Lorsqu'elle est connectée au point de terminaison de l'instance pour `instance-2470`, une requête indique que le nouveau paramètre est en vigueur.

```
mysql> select @@lower_case_table_names;
+--------------------------+
| @@lower_case_table_names |
+--------------------------+
|                        1 |
+--------------------------+
```

À ce stade, les deux instances de lecteur du cluster sont exécutées avec des paramètres `lower_case_table_names` différents. Ainsi, toute connexion au point de terminaison du lecteur du cluster utilise une valeur imprévisible pour ce paramètre. Il est important de redémarrer immédiatement l'autre instance de lecteur afin qu'elles aient toutes les deux des paramètres cohérents.

```
$ aws rds reboot-db-instance --db-instance-identifier instance-5138
{
  "DBInstanceIdentifier": "instance-5138",
  "DBInstanceStatus": "rebooting"
}
$ aws rds wait db-instance-available --db-instance-id instance-5138
```

L'exemple suivant confirme que toutes les instances de lecteur ont le même paramètre `lower_case_table_names`. Les commandes vérifient la valeur du paramètre `lower_case_table_names` sur chaque instance de lecteur. Ensuite, la même commande utilisant le point de terminaison de lecteur montre que chaque connexion au point de terminaison du lecteur utilise l'une des instances de lecteur, mais il n'est pas possible de prévoir laquelle.

```
# Check lower_case_table_names setting on each reader instance.

$ mysql -h instance-5138.a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-5138            |                        1 |
+--------------------------+--------------------------+

$ mysql -h instance-2470.a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-2470            |                        1 |
+--------------------------+--------------------------+

# Check lower_case_table_names setting on the reader endpoint of the cluster.

$ mysql -h cluster-2393.cluster-ro-a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-5138            |                        1 |
+--------------------------+--------------------------+

# Run query on writer instance

$ mysql -h cluster-2393.cluster-a12345.us-east-1.rds.amazonaws.com \
  -u my-user -p -e 'select @@aurora_server_id, @@lower_case_table_names'
+--------------------------+--------------------------+
| @@aurora_server_id       | @@lower_case_table_names |
+--------------------------+--------------------------+
| instance-9404            |                        1 |
+--------------------------+--------------------------+
```

Avec le changement de paramètre appliqué partout, nous pouvons voir l'effet du réglage `lower_case_table_names=1`. S'il est fait référence à la table en tant que `foo`, `Foo` ou `F00`, la requête convertit le nom en `foo` et accède à la même table dans chacun des cas.

```
mysql> use lctn;

mysql> select * from foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from Foo;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+

mysql> select * from FOO;
+--------------------------+
| s                        |
+--------------------------+
| Lowercase table name foo |
+--------------------------+
```

Basculement vers un cluster de bases de données Amazon Aurora

Vous pouvez effectuer le basculement manuel d'un cluster de bases de données Aurora, notamment lorsque vous souhaitez remplacer une instance de base de données d'enregistreur provisionnée par une instance d'enregistreur Aurora Serverless v2.

Aurora bascule vers une nouvelle instance de base de données principale de l'une des deux façons suivantes :

- En faisant d'un réplica de base de données de lecteur existant la nouvelle instance principale
- En créant une autre instance principale

Si le cluster de bases de données possède une ou plusieurs instances de lecteur, un lecteur est promu au statut d'instance principale lors d'un événement d'échec. Pour augmenter la disponibilité de votre cluster de bases de données, nous vous recommandons de créer au moins une instance de lecteur dans deux zones de disponibilité ou plus. Pour plus d'informations sur le mécanisme de basculement, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.

Vous pouvez utiliser la AWS Management Console, l'AWS CLI ou l'API RDS pour effectuer un basculement manuel.

Console

Pour effectuer le basculement d'un cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis sélectionnez l'instance de base de données du cluster de bases de données dont vous souhaitez effectuer le basculement.
3. Pour Actions, choisissez Failover (Basculement).

   Une page de confirmation s'affiche.
4. Choisissez Basculement.

   La page Bases de données indique que l'état du cluster de bases de données est Basculement. Cet état indiquera Disponible dès que le basculement sera terminé, et les rôles des nouvelles et anciennes instances de base de données principales seront affichés.

AWS CLI

Pour effectuer le basculement manuel d'un cluster de bases de données multi-AZ à l'aide de l'AWS CLI, utilisez la commande failover-db-cluster. Spécifiez les paramètres suivants :

- `--db-cluster-identifier` : cluster de bases de données multi-AZ que vous voulez faire basculer.
- `--target-db-instance-identifier` : nom de l'instance de base de données à promouvoir en instance de base de données principale.

Example

Pour Linux, macOS ou Unix :

```
aws rds failover-db-cluster \
    --db-cluster-identifier mydbcluster \
    --target-db-instance-identifier mydbcluster-instance-2
```

Pour Windows :

```
aws rds failover-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --target-db-instance-identifier mydbcluster-instance-2
```

API RDS

Pour modifier un cluster de bases de données à partir de l'API Amazon RDS, appelez l'opération FailoverDBCluster. Spécifiez les paramètres suivants :

- DBClusterIdentifier
- TargetDBInstanceIdentifier

Suppression de clusters de bases de données Aurora et d'instances de bases de données

Vous pouvez supprimer un cluster de bases de données Aurora lorsque vous n'en avez plus besoin. La suppression du cluster supprime le volume de cluster contenant toutes vos données. Avant de supprimer le cluster, vous pouvez enregistrer un instantané de vos données. Vous pouvez restaurer l'instantané ultérieurement pour créer un nouveau cluster contenant les mêmes données.

Vous pouvez également supprimer des instances de base de données d'un cluster, tout en préservant le cluster à proprement parler, ainsi que les données qu'il contient. La suppression d'instances de base de données peut vous permettre de limiter les frais si le cluster n'est pas occupé et ne requiert pas la capacité de calcul de plusieurs instances de base de données.

Rubriques

Suppression d'un cluster de bases de données Aurora

Aurora ne fournit pas de méthode à une seule étape pour supprimer un cluster de bases de données. Ce choix de conception est destiné à vous éviter de perdre des données ou de mettre votre application hors ligne accidentellement. Généralement, les applications Aurora sont critiques et nécessitent une haute disponibilité. Ainsi, Aurora facilite la mise à l'échelle de la capacité du cluster en ajoutant et supprimant des instances de bases de données. La suppression du cluster de bases de données lui-même vous oblige à effectuer une suppression distincte.

Suivez la procédure générale ci-dessous pour supprimer toutes les instances de bases de données d'un cluster, puis supprimez le cluster lui-même.

1. Supprimez toutes les instances de lecteur du cluster. Utilisez la procédure disponible dans Suppression d'une instance de base de données d'un cluster de bases de données Aurora.

Si le cluster comporte plusieurs instances de lecteur, la suppression de l'une d'entre elles réduit la capacité de calcul du cluster. La suppression des instances de lecteur assure la disponibilité du cluster tout au long de la procédure et permet d'éviter les opérations de basculement inutiles.

2. Supprimez l'instance de scripteur du cluster. Là encore, utilisez la procédure disponible dans Suppression d'une instance de base de données d'un cluster de bases de données Aurora.

   Quand vous supprimez les instances de base de données, le cluster et son volume de cluster associé sont conservés même après avoir supprimé toutes les instances de base de données.

3. Supprimez le cluster de bases de données.

   - AWS Management Console : choisissez le cluster, puis choisissez Supprimer dans le menu Actions. Vous pouvez choisir les options suivantes pour conserver les données du cluster au cas où vous en auriez besoin ultérieurement :
     - Créez un instantané final du volume de cluster. Le paramètre par défaut consiste à créer un instantané final.
     - Conservation des sauvegardes automatiques. Le paramètre par défaut est de ne pas conserver les sauvegardes automatiques.

     > **Note**
     >
     > Les sauvegardes automatisées des clusters de bases de données Aurora Serverless v1 ne sont pas conservées.

     Aurora exige également que vous confirmiez votre intention de supprimer le cluster.
   - Interface CLI et API : appelez la commande CLI `delete-db-cluster` ou l'opération d'API `DeleteDBCluster`. Vous pouvez choisir les options suivantes pour conserver les données du cluster au cas où vous en auriez besoin ultérieurement :
     - Créez un instantané final du volume de cluster.
     - Conservation des sauvegardes automatiques.

     > **Note**
     >
     > Les sauvegardes automatisées des clusters de bases de données Aurora Serverless v1 ne sont pas conservées.

Rubriques

Suppression d’un cluster Aurora vide

Vous pouvez supprimer un cluster de base de données vide à l'aide de l'API AWS Management Console AWS CLI, ou Amazon RDS.

> **Tip**
>
> Vous pouvez conserver un cluster sans instance de base de données pour préserver vos données sans frais d’UC pour le cluster. Vous pouvez rapidement réutiliser le cluster en créant une ou plusieurs nouvelles instances de bases de données. Toutefois, vous ne pouvez ajouter de nouvelles instances de base de données qu'à l'aide de l'API AWS CLI ou de l'API RDS. Vous ne pouvez pas ajouter de nouvelles instances de base de données à l’aide de la console Amazon RDS. Vous pouvez effectuer des opérations d’administration Aurora sur le cluster sans qu’une instance de base de données ne lui soit associée. Vous ne pouvez simplement pas accéder aux données ou effectuer des opérations nécessitant une connexion à une instance de base de données.

Console

Pour supprimer un cluster DB

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Bases de données, puis choisissez le cluster de bases de données que vous souhaitez supprimer.
3. Pour Actions, choisissez Supprimer.
4. Pour créer un instantané de base de données final pour le cluster de bases de données, choisissez Créer un instantané final ?. Il s’agit du paramètre par défaut.
5. Si vous avez choisi de créer un instantané final, entrez le paramètre Final snapshot name (Nom de l’instantané final).

6. Pour conserver les sauvegardes automatiques, choisissez Conserver les sauvegardes automatiques. Ceci n'est pas le paramètre par défaut.
7. Saisissez **`delete me`** dans la zone.
8. Sélectionnez Delete (Supprimer).

INTERFACE DE LIGNE DE COMMANDE (CLI)

Pour supprimer un cluster de base de données Aurora vide à l'aide de AWS CLI, appelez la delete-db-clustercommande.

Supposons que le cluster vide `deleteme-zero-instances` ait été uniquement utilisé à des fins de développement et de test et ne contienne pas de données importantes. Dans ce cas, vous n'avez pas besoin de conserver un instantané du volume de cluster lorsque vous supprimez le cluster. L'exemple suivant montre qu'un cluster ne contient aucune instance de base de données, puis supprime le cluster vide sans créer d'instantané final ni conserver de sauvegardes automatiques.

```
$ aws rds describe-db-clusters --db-cluster-identifier deleteme-zero-instances --output
 text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-zero-instances

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-zero-instances \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-zero-instances",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

API RDS

Pour supprimer un cluster de base de données Aurora vide à l'aide de l'API Amazon RDS, appelez l'DBClusteropération Delete.

Suppression d'un cluster Aurora avec une seule instance de base de données

Vous pouvez supprimer la dernière instance de base de données, même si la protection contre la suppression est activée pour le cluster de bases de données. Dans ce cas, le cluster de bases de

données continue d'exister et vos données sont préservées. Vous pouvez accéder à nouveau à ces données en attachant une nouvelle instance de base de données au cluster.

L'exemple suivant montre que la commande `delete-db-cluster` ne fonctionne pas lorsque des instances de bases de données sont toujours associées au cluster. Ce cluster possède une instance de base de données de scripteur unique. Lorsque nous examinons les instances de bases de données du cluster, nous vérifions l'attribut `IsClusterWriter` de chacune. Le cluster peut avoir zéro, voire une instance de base de données de scripteur. Une valeur de `true` indique une instance de base de données de scripteur. Une valeur de `false` indique une instance de base de données de lecteur. Le cluster peut avoir zéro, une ou plusieurs instances de bases de données de lecteur. Dans ce cas, nous supprimons l'instance de base de données de scripteur à l'aide de la commande `delete-db-instance`. Une fois l'instance de base de données dans l'état `deleting`, nous pouvons également supprimer le cluster. Dans cet exemple également, supposons que le cluster ne contienne pas de données à conserver. Par conséquent, nous ne créons pas d'instantané du volume de cluster et ne conservons pas de sauvegardes automatiques.

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-only --skip-final-
snapshot
An error occurred (InvalidDBClusterStateFault) when calling the DeleteDBCluster
 operation:
  Cluster cannot be deleted, it still contains DB instances in non-deleting state.

$ aws rds describe-db-clusters --db-cluster-identifier deleteme-writer-only \
  --query '*[].[DBClusterIdentifier,Status,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]'
[
    [
        "deleteme-writer-only",
        "available",
        [
            [
                "instance-2130",
                true
            ]
        ]
    ]
]

$ aws rds delete-db-instance --db-instance-identifier instance-2130
{
  "DBInstanceIdentifier": "instance-2130",
```

```
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-only \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-writer-only",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

Suppression d'un Aurora cluster avec plusieurs instances de bases de données

Si votre cluster contient plusieurs instances de bases de données, il existe généralement une seule instance de scripteur et une ou plusieurs instances de lecteur. Les instances de lecteur facilitent la haute disponibilité, en étant en veille pour prendre le relais si l'instance de scripteur rencontre un problème. Vous pouvez également utiliser des instances de lecteur pour mettre à l'échelle le cluster afin de gérer une charge de travail intensive en lecture, sans ajouter de frais à l'instance de scripteur.

Pour supprimer un cluster avec plusieurs instances de bases de données de lecteur, supprimez d'abord les instances de lecteur, puis l'instance de scripteur. La suppression de l'instance d'enregistreur n'a aucune incidence sur le cluster et ses données. Vous supprimez le cluster par le biais d'une action distincte.

- Pour obtenir la procédure de suppression d'une instance de base de données Aurora, consultez Suppression d'une instance de base de données d'un cluster de bases de données Aurora.
- Pour obtenir la procédure de suppression de l'instance de base de données de scripteur dans un cluster Aurora, consultez Suppression d'un cluster Aurora avec une seule instance de base de données.
- Pour obtenir la procédure de suppression d'un cluster Aurora vide, consultez Suppression d'un cluster Aurora vide.

Cet exemple d'interface CLI montre comment supprimer un cluster contenant une instance de base de données d'enregistreur et une instance de base de données de lecteur unique. La sortie `describe-db-clusters` indique que `instance-7384` correspond à l'instance de scripteur et `instance-1039` à l'instance de lecteur. L'exemple supprime d'abord l'instance de lecteur, car la suppression de l'instance de scripteur en présence d'une instance de lecteur entraînerait

une opération de basculement. Il n'est pas pertinent de promouvoir l'instance de lecteur vers un scripteur si vous envisagez de supprimer également cette instance. Là encore, supposons que ces instances `db.t2.small` sont uniquement utilisées à des fins de développement et de test, si bien que l'opération de suppression ignore l'instantané final et ne conserve pas les sauvegardes automatiques.

```
$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-and-reader --skip-
final-snapshot

An error occurred (InvalidDBClusterStateFault) when calling the DeleteDBCluster
 operation:
  Cluster cannot be deleted, it still contains DB instances in non-deleting state.

$ aws rds describe-db-clusters --db-cluster-identifier deleteme-writer-and-reader --
output text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-writer-and-reader
Instance:       instance-1039  False
Instance:       instance-7384   True

$ aws rds delete-db-instance --db-instance-identifier instance-1039
{
  "DBInstanceIdentifier": "instance-1039",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-7384
{
  "DBInstanceIdentifier": "instance-7384",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-writer-and-reader \
  --skip-final-snapshot \
  --delete-automated-backups
{
  "DBClusterIdentifier": "deleteme-writer-and-reader",
  "Status": "available",
  "Engine": "aurora-mysql"
```

```
}
```

L'exemple suivant montre comment supprimer un cluster de bases de données contenant une instance de base de données de scripteur et plusieurs instances de bases de données de lecteur. Il utilise une sortie concise issue de la commande `describe-db-clusters` pour obtenir un rapport des instances de scripteur et de lecteur. Là encore, nous supprimons toutes les instances de bases de données de lecteur avant de supprimer l'instance de base de données de scripteur. L'ordre dans lequel nous supprimons les instances de bases de données de lecteur n'a pas d'importance.

Supposons que ce cluster avec plusieurs instances de bases de données contiennent des données qui méritent d'être conservées. La commande `delete-db-cluster` de cet exemple inclut les paramètres `--no-skip-final-snapshot` et `--final-db-snapshot-identifier` pour spécifier les détails de l'instantané à créer. Il inclut également le paramètre `--no-delete-automated-backups` pour conserver les sauvegardes automatisées.

```
$ aws rds describe-db-clusters --db-cluster-identifier deleteme-multiple-readers --
output text \
  --query '*[].["Cluster:",DBClusterIdentifier,DBClusterMembers[*].
["Instance:",DBInstanceIdentifier,IsClusterWriter]]
Cluster:        deleteme-multiple-readers
Instance:       instance-1010   False
Instance:       instance-5410   False
Instance:       instance-9948   False
Instance:       instance-8451   True

$ aws rds delete-db-instance --db-instance-identifier instance-1010
{
  "DBInstanceIdentifier": "instance-1010",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-5410
{
  "DBInstanceIdentifier": "instance-5410",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-9948
{
  "DBInstanceIdentifier": "instance-9948",
```

```
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-instance --db-instance-identifier instance-8451
{
  "DBInstanceIdentifier": "instance-8451",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql"
}

$ aws rds delete-db-cluster --db-cluster-identifier deleteme-multiple-readers \
  --no-delete-automated-backups \
  --no-skip-final-snapshot \
  --final-db-snapshot-identifier deleteme-multiple-readers-final-snapshot
{
  "DBClusterIdentifier": "deleteme-multiple-readers",
  "Status": "available",
  "Engine": "aurora-mysql"
}
```

L'exemple suivant montre comment vérifier que Aurora a créé l'instantané demandé. Vous pouvez demander les détails relatifs à l'instantané spécifique en indiquant son identifiant `deleteme-multiple-readers-final-snapshot`. Vous pouvez également obtenir un rapport de tous les instantanés du cluster supprimé en indiquant l'identifiant du cluster `deleteme-multiple-readers`. Ces deux commandes renvoient des informations relatifs au même instantané.

```
$ aws rds describe-db-cluster-snapshots \
  --db-cluster-snapshot-identifier deleteme-multiple-readers-final-snapshot
{
    "DBClusterSnapshots": [
        {
            "AvailabilityZones": [],
            "DBClusterSnapshotIdentifier": "deleteme-multiple-readers-final-snapshot",
            "DBClusterIdentifier": "deleteme-multiple-readers",
            "SnapshotCreateTime": "11T01:40:07.354000+00:00",
            "Engine": "aurora-mysql",
...

$ aws rds describe-db-cluster-snapshots --db-cluster-identifier deleteme-multiple-
readers
{
    "DBClusterSnapshots": [
```

```
        {
            "AvailabilityZones": [],
            "DBClusterSnapshotIdentifier": "deleteme-multiple-readers-final-snapshot",
            "DBClusterIdentifier": "deleteme-multiple-readers",
            "SnapshotCreateTime": "11T01:40:07.354000+00:00",
            "Engine": "aurora-mysql",
...
```

Protection contre la suppression pour les clusters Aurora

Vous ne pouvez pas supprimer les clusters dont la protection contre la suppression est activée. Vous pouvez supprimer les instances de bases de données du cluster, mais pas le cluster à proprement parler. Ainsi, le volume de cluster contenant vos données est protégé contre toute suppression accidentelle. Aurora applique la protection contre la suppression pour un cluster de base de données, que vous essayiez de supprimer le cluster à l'aide de la console, de l' AWS CLI API RDS ou de l'API RDS.

La protection contre la suppression est activée par défaut lorsque vous créez un cluster de bases de données de production à l'aide d AWS Management Console. Toutefois, la protection contre la suppression est désactivée par défaut si vous créez un cluster à l'aide de l'API AWS CLI or. L'activation ou la désactivation de la protection contre la suppression n'entraîne pas d'interruption de service. Pour pouvoir supprimer le cluster, modifiez-le et désactivez la protection contre la suppression. Pour de plus amples informations sur l'activation et la désactivation de la protection contre la suppression, consultez Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API.

> **Tip**
>
> Malgré la suppression de toutes les instances de bases de données, vous pouvez accéder aux données en créant une nouvelle instance de base de données dans le cluster.

Suppression d'un cluster Aurora arrêté

Vous ne pouvez pas supprimer un cluster dont l'état est `stopped`. Dans ce cas, démarrez le cluster avant de le supprimer. Pour plus d'informations, consultez Démarrage d'un cluster de bases de données Aurora.

Suppression de clusters Aurora MySQL correspondant à des réplicas en lecture

Pour Aurora MySQL, vous ne pouvez pas supprimer une instance de base de données dans un cluster de bases de données si les deux conditions suivantes sont vraies :

- Le cluster de bases de données est un réplica en lecture d'un autre cluster de bases de données Aurora.
- L'instance de base de données est la seule instance dans le cluster de bases de données.

Pour supprimer une instance de base de données dans ce cas-ci, effectuez d'abord la promotion du cluster de bases de données afin qu'il ne soit plus un réplica en lecture. Une fois la promotion terminée, vous pouvez supprimer l'instance finale de base de données dans votre cluster de base de données. Pour plus d'informations, consultez Réplication de clusters de bases de données Amazon Aurora MySQL dans différentes Régions AWS.

Instantané final lors de la suppression d'un cluster

Dans cette section, les exemples montrent comment choisir de prendre ou non un instantané final lorsque vous supprimez un cluster Aurora. Si vous choisissez de prendre un instantané final mais que le nom que vous spécifiez correspond à un instantané existant, l'opération s'arrête avec une erreur. Dans ce cas, examinez les détails de l'instantané pour vérifier s'il s'agit bien de votre instantané actuel ou d'un instantané plus ancien. Si l'instantané existant ne contient pas les données les plus récentes que vous souhaitez conserver, renommez-le et réessayez, ou spécifiez un autre nom pour le paramètre d'instantané final.

Suppression d'une instance de base de données d'un cluster de bases de données Aurora

Vous pouvez supprimer une instance de base de données d'un cluster de bases de données Aurora dans le cadre du processus de suppression du cluster entier. Si votre cluster contient un certain nombre d'instances de bases de données, la suppression du cluster implique la suppression de chacune de ces instances de bases de données. Vous pouvez également supprimer une ou plusieurs instances de lecteur d'un cluster sans arrêter l'exécution de celui-ci. Vous pouvez procéder ainsi pour réduire la capacité de calcul et les frais associés si votre cluster n'est pas occupé.

Pour supprimer une instance de base de données, vous devez spécifiez le nom de l'instance.

Vous pouvez supprimer une instance de base de données à l'aide de l'API AWS Management Console, de AWS CLI, ou de l'API RDS.

> **Note**
>
> Quand un réplica Aurora est supprimé, le point de terminaison de son instance est supprimé immédiatement et le réplica Aurora est supprimé du point de terminaison du lecteur. S'il y a des instructions qui s'exécutent sur le réplica Aurora en cours de suppression, une période de grâce de trois minutes est accordée. Les instructions existantes peuvent se terminer pendant la période de grâce. Lorsque la période de grâce se termine, le réplica Aurora est arrêté et supprimé.

Pour les clusters de bases de données Aurora, la suppression d'une instance de base de données ne supprime pas systématiquement le cluster entier. Vous pouvez supprimer une instance de base de données dans un cluster Aurora pour réduire la capacité de calcul et les frais associés lorsque le cluster n'est pas occupé. Pour plus d'informations sur les cas particuliers liés aux clusters Aurora ne présentant aucune instance de base de données, voire une instance de base de données, consultez Suppression d'un cluster Aurora avec une seule instance de base de données et Suppression d'un cluster Aurora vide.

> **Note**
>
> Vous ne pouvez pas supprimer un cluster de bases de données lorsque la protection contre la suppression est activée pour celui-ci. Pour plus d'informations, consultez Protection contre la suppression pour les clusters Aurora.
> Vous pouvez désactiver la protection contre la suppression en modifiant le cluster de bases de données. Pour de plus amples informations, veuillez consulter Modification d'un cluster de bases de données Amazon Aurora.

Console

Pour supprimer une instance de base de données dans un cluster de bases de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.

2. Dans le panneau de navigation, choisissez Bases de données, puis l'instance de base de données que vous souhaitez supprimer.
3. Pour Actions, choisissez Supprimer.
4. Saisissez **`delete me`** dans la zone.
5. Sélectionnez Delete (Supprimer).

AWS CLI

Pour supprimer une instance de base de données à l'aide de AWS CLI, appelez la delete-db-instancecommande et spécifiez la `--db-instance-identifier` valeur.

Example

Pour Linux, macOS ou Unix :

```
aws rds delete-db-instance \
    --db-instance-identifier mydbinstance
```

Pour Windows :

```
aws rds delete-db-instance ^
    --db-instance-identifier mydbinstance
```

API RDS

Pour supprimer une instance de base de données à l'aide de l'API Amazon RDS, appelez l'DBInstanceopération Delete et spécifiez le `DBInstanceIdentifier` paramètre.

> **Note**
>
> Lorsque le statut d'une instance de base de données est`deleting`, la valeur de son certificat CA n'apparaît pas dans la console RDS ni dans les sorties des AWS CLI commandes ou des opérations de l'API RDS. Pour plus d'informations sur les certificats d'autorité de certification, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.

Marquage des ressources Amazon Aurora et Amazon RDS

Une balise Amazon RDS est une paire nom-valeur que vous définissez et associez avec une ressource Amazon RDS, comme une instance de base de données ou un instantané de base de données. Le nom s'appelle la clé. Vous pouvez éventuellement fournir une valeur pour la clé.

Vous pouvez utiliser l'API AWS Management ConsoleAWS CLI, la ou l'API Amazon RDS pour ajouter, répertorier et supprimer des balises sur les ressources Amazon RDS. Lorsque vous utilisez l'interface de ligne de commande ou l'API, assurez-vous de fournir l'Amazon Resource Name (ARN) pour la ressource RDS avec laquelle vous souhaitez travailler. Pour plus d'informations sur la création d'un ARN, consultez Création d'un ARN pour Amazon RDS.

Vous pouvez utiliser des balises pour ajouter des métadonnées à vos ressources Aurora et Amazon RDS. Vous pouvez utiliser les balises pour ajouter vos propres notations sur les instances de base de données, les instantanés, les clusters Aurora, etc. Cela peut vous aider à documenter vos ressources Aurora et Amazon RDS. Vous pouvez également utiliser les balises avec des procédures de maintenance automatisées.

Vous pouvez notamment utiliser ces balises avec les politiques IAM. Vous pouvez les utiliser pour gérer l'accès aux ressources Aurora et Amazon RDS et contrôler les actions qui peuvent être appliquées à ces ressources. Vous pouvez également utiliser ces balises pour suivre les coûts en regroupant les dépenses pour des ressources balisées de la même façon.

Vous pouvez baliser les ressources Aurora et Amazon RDS suivantes :

- Instances DB
- Clusters DB
- Clusters globaux Aurora
- Points de terminaison de cluster de bases de données
- Réplicas en lecture
- Instantanés de base de données
- Instantanés de cluster DB
- Instances DB réservées
- Abonnements aux événements
- Groupes d'options DB
- Groupes de paramètres DB

- Groupes de paramètres de cluster DB
- Groupes de sous-réseaux DB
- Proxys RDS
- Points de terminaison RDS Proxy
- Déploiements bleus/verts
- Intégrations sans ETL
- Sauvegardes automatiques
- Sauvegardes automatisées en cluster

> **Note**
>
> Lorsque vous balisez une instance de base de données, Aurora applique automatiquement ces balises aux ressources Performance Insights associées. Actuellement, vous ne pouvez pas baliser les proxys RDS et les points de terminaison des proxys RDS à l'aide du. AWS Management Console

Rubriques

Pourquoi utiliser des balises des ressources Amazon RDS ?

Vous pouvez utiliser des balises pour effectuer les tâches suivantes :

- Classer vos ressources RDS par application, projet, département, environnement, etc. Par exemple, vous pouvez utiliser une clé de balise pour définir une catégorie, et la valeur de balise peut être un élément de cette catégorie. Vous pouvez créer la balise `environment=prod`. Par

exemple, vous pouvez définir une clé de balise appelée `project` et une valeur de balise appelée `Salix`, en indiquant que la ressource Amazon RDS est attribuée au projet Salix.

- Automatisez les tâches de gestion des ressources. Par exemple, vous pouvez créer une fenêtre de maintenance pour les instances balisées avec `environment=prod` différentes de la fenêtre pour les instances balisées avec `environment=test`. Vous pouvez également configurer des instantanés de base de données automatiques pour les instances balisées avec `environment=prod`.
- Contrôlez l'accès aux ressources RDS dans le cadre d'une politique IAM. Pour cela, vous devez utiliser la clé de condition globale `aws:ResourceTag/`*`tag-key`*. Par exemple, une politique peut autoriser uniquement les utilisateurs du groupe `DBAdmin` à modifier les instances de base de données balisées avec `environment=prod`. Pour plus d'informations sur la gestion de l'accès aux ressources balisées à l'aide de politiques IAM, consultez Identity and Access Management pour Amazon Aurora la section Contrôle de l'accès aux AWS ressources dans le guide de l'utilisateur d'AWSIdentity and Access Management.
- Surveillez les ressources en fonction d'une balise. Par exemple, vous pouvez créer un tableau de CloudWatch bord Amazon pour les instances de base de données étiquetées avec`environment=prod`.
- Suivez les coûts en regroupant les dépenses pour des ressources balisées de la même façon. Par exemple, si vous balisez les ressources RDS associées au projet Salix avec `project=Salix`, vous pouvez générer des rapports de coûts et allouer des dépenses à ce projet. Pour plus d'informations, consultez Comment fonctionne AWS la facturation avec les tags dans Amazon RDS.

Fonctionnement des balises des ressources Amazon RDS

AWSn'applique aucune signification sémantique à vos balises. Les balises sont interprétées de façon stricte, en tant que chaîne de caractères.

Rubriques

- Ensembles de balises dans Amazon RDS
- Structure des balises dans Amazon RDS
- Ressources Amazon RDS éligibles au balisage
- Comment fonctionne AWS la facturation avec les tags dans Amazon RDS

Ensembles de balises dans Amazon RDS

Chaque ressource Amazon RDS possède un conteneur appelé ensemble de balises. Le conteneur inclut toutes les balises attribuées à la ressource. Une ressource possède exactement un ensemble de balises.

Un ensemble de balises contient de 0 à 50 balises. Si vous ajoutez une balise à une ressource RDS ayant la même clé qu'une balise existante sur cette ressource, la nouvelle valeur remplace l'ancienne valeur.

Structure des balises dans Amazon RDS

La structure d'une balise RDS se présente comme suit :

Clé de balise

La clé de balise correspond au nom obligatoire de la balise. La valeur de la chaîne peut comporter de 1 à 128 caractères Unicode et elle ne peut pas être précédée de `aws:` ou de `rds:`. La chaîne peut uniquement contenir l'ensemble de lettres, de chiffres et d'espaces Unicode, _, `.`, `:`, `/`, =, +, `-` et `@`. L'expression régulière Java est `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"`. Les clés de balises sont sensibles à la casse. Ainsi, les clés `project` et `Project` sont distinctes.

Une clé est propre à un ensemble de balises. Par exemple, vous ne pouvez pas avoir une paire-clé dans un ensemble de balises avec la clé identique mais des valeurs différentes comme `project=Trinity` et `project=Xanadu`.

Valeur de balise

La valeur de balise correspond à la valeur de chaîne facultative d'une balise. La valeur de la chaîne peut comporter de 1 à 256 caractères Unicode. La chaîne peut uniquement contenir l'ensemble de lettres, de chiffres et d'espaces Unicode, _, `.`, `:`, `/`, =, +, `-` et `@`. L'expression régulière Java est `"^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-@]*)$"`. Les valeurs de balises sont sensibles à la casse. Ainsi, les valeurs `prod` et `Prod` sont distinctes.

Les valeurs ne doivent pas nécessairement être uniques et peuvent être null. Par exemple, vous pouvez avoir une paire clé-valeur dans un ensemble de balises `project=Trinity` et `cost-center=Trinity`.

Ressources Amazon RDS éligibles au balisage

Vous pouvez baliser les ressources Amazon RDS suivantes :

- Instances DB
- Clusters DB
- Points de terminaison de cluster de bases de données
- Réplicas en lecture
- Instantanés de base de données
- Instantanés de cluster DB
- Instances DB réservées
- Abonnements aux événements
- Groupes d'options DB
- Groupes de paramètres DB
- Groupes de paramètres de cluster DB
- Groupes de sous-réseaux DB
- Proxys RDS
- Points de terminaison RDS Proxy

> **Note**
>
> À l'heure actuelle, vous ne pouvez pas étiqueter les proxys RDS et les points de terminaison RDS Proxy à l'aide de la AWS Management Console.

- Déploiements bleus/verts
- Intégrations zéro ETL (version préliminaire)
- Sauvegardes automatiques
- Sauvegardes automatisées en cluster

Comment fonctionne AWS la facturation avec les tags dans Amazon RDS

Utilisez des balises pour organiser votre AWS facture afin de refléter votre propre structure de coûts. Pour ce faire, inscrivez-vous pour recevoir votre Compte AWS facture avec les valeurs clés du tag incluses. Ensuite, pour voir le coût de vos ressources combinées, organisez vos informations de facturation en fonction des ressources possédant les mêmes valeurs de clé de balise. Par exemple, vous pouvez baliser plusieurs ressources avec un nom d'application spécifique, puis organiser vos informations de facturation pour afficher le coût total de cette application dans plusieurs services.

Pour plus d'informations, consultez Utilisation des balises d'allocation des coûts dans le Guide de l'utilisateur AWS Billing.

Fonctionnement des balises de répartition des coûts avec les instantanés de cluster de bases de données

Vous pouvez ajouter un cluster de bases de données à un instantané. Toutefois, votre facture ne reflètera pas ce groupement. Pour que des balises de répartition des coûts s'appliquent aux instantanés du cluster de bases de données, les conditions suivantes doivent être remplies :

- Les balises doivent être attachées à l'instance de base de données parent.
- L'instance de base de données parent doit exister au même endroit Compte AWS que le snapshot du cluster de base de données.
- L'instance de base de données parent doit exister au même endroit Région AWS que le snapshot du cluster de base de données.

Les instantanés de cluster de bases de données sont considérés comme orphelins s'ils n'existent pas dans la même région que l'instance de base de données parent ou si l'instance de base de données parent est supprimée. Les instantanés de base de données orphelins ne prennent pas en charge les balises de répartition des coûts. Les coûts des instantanés orphelins sont agrégés dans un seul élément de ligne non balisé. Les instantanés de cluster de bases de données entre comptes ne sont pas considérés comme orphelins lorsque les conditions suivantes sont remplies :

- Ils existent dans la même région que l'instance de base de données parent.
- L'instance de base de données parent appartient au compte source.

> **Note**
>
> Si l'instance de base de données parent appartient à un autre compte, les balises de répartition des coûts ne s'appliquent pas aux instantanés entre comptes du compte de destination.

Bonnes pratiques relatives au balisage de ressources Amazon RDS

Lorsque vous utilisez des balises, nous vous recommandons de respecter les bonnes pratiques ci-dessous :

- Documentez les conventions relatives à l'utilisation des balises qui sont suivies par toutes les équipes de votre organisation. En particulier, assurez-vous que les noms sont à la fois descriptifs et cohérents. Par exemple, normalisez le format `environment:prod` plutôt que de baliser certaines ressources avec `env:production`.

  > **Important**
  >
  > Ne stockez pas d'informations personnelles identifiables (PII) ou d'autres informations confidentielles ou sensibles dans des balises.

- Automatisez le balisage pour garantir la cohérence. Par exemple, vous pouvez utiliser les techniques suivantes :
  - Incluez des balises dans un CloudFormation modèle. Lorsque vous créez des ressources à l'aide du modèle, elles sont balisées automatiquement.
  - Définissez et appliquez des balises à l'aide de AWS Lambda fonctions.
  - Créez un document SSM qui inclut les étapes pour ajouter des balises à vos ressources RDS.
- Utilisez des balises uniquement lorsque cela est nécessaire. Vous pouvez ajouter jusqu'à 50 balises pour une seule ressource RDS, mais une bonne pratique consiste à éviter la prolifération et la complexité inutiles des balises.
- Vérifiez régulièrement la pertinence et l'exactitude des balises. Supprimez ou modifiez les balises obsolètes selon vos besoins.
- Pensez à créer des balises à l'aide de l'éditeur de AWS balises dans leAWS Management Console. Vous pouvez utiliser l'éditeur de balises pour ajouter des balises à plusieurs AWS ressources prises en charge, y compris les ressources RDS, en même temps. Pour plus d'informations, consultez Tag Editor dans le Guide de l'utilisateur d'AWS Resource Groups.

Copie de balises vers des instantanés de cluster de bases de données

Lorsque vous créez ou restaurez un cluster de bases de données, vous pouvez spécifier que les balises du cluster sont copiées vers des instantanés du cluster de bases de données. La copie des balises garantit que les métadonnées pour les instantanés de base de données correspondent au cluster de bases de données source. Elle garantit également que toutes les stratégies d'accès pour l'instantané de base de données correspondent également au cluster de bases de données source. Les balises ne sont pas copiées par défaut.

Vous pouvez spécifier que les balises soient copiées vers des snapshots DB pour les actions suivantes :

- Création d'un cluster de bases de données
- Restauration d'un cluster de bases de données
- Création d'un réplica en lecture
- Copie d'un instantané de cluster de bases de données

> **Note**
>
> Dans certains cas, vous pouvez inclure une valeur pour le `--tags` paramètre de la create-db-snapshotAWS CLIcommande. Vous pouvez également fournir au moins une balise à l'opération Create DBSnapshot API. Dans ces cas, RDS ne copie pas les balises de l'instance de base de données source vers le nouvel instantané de base de données. Cette fonctionnalité s'applique même si l'option `--copy-tags-to-snapshot` (`CopyTagsToSnapshot`) est activée sur l'instance de base de données source.
> Si vous optez pour cette approche, vous pouvez créer une copie d'une instance de base de données à partir d'un instantané de base de données. Cette approche évite d'ajouter des balises qui ne s'appliquent pas à la nouvelle instance de base de données. Vous créez votre instantané de base de données à l'aide de la AWS CLI `create-db-snapshot` commande (ou de l'opération de l'API `CreateDBSnapshot` RDS). Après avoir créé votre instantané de base de données, vous pouvez ajouter des balises comme décrit plus loin dans cette rubrique.

Marquage des ressources de sauvegarde automatisées

Les ressources de sauvegarde automatisées sont créées lorsque vous définissez une valeur de période de conservation des sauvegardes comprise entre 0 et une valeur supérieure à 0. Vous pouvez étiqueter les ressources de sauvegarde automatisées de l'instance ou du cluster lors de la création à l'aide du `--tag-specifications` paramètre.

Paramètre de spécifications des balises

APIs qui prennent en charge le paramètre de `--tag-specifications` requête (comme create-db-instancerestore-db-instance-from-db-snapshot create-db-cluster, etc.) peuvent baliser les

sauvegardes automatisées (type de ressource : `auto-backup` ou`cluster-auto-backup`) lors de la création.

Balisage des sauvegardes automatisées du cluster

`--tag-specifications`À utiliser `ResourceType=cluster-auto-backup` lors de la création de clusters de bases de données sur lesquels les sauvegardes automatiques sont activées.

> **Note**
>
> - Les balises de sauvegarde automatisées sont indépendantes des balises d'instance de base de données source, des balises de cluster de base de données ou des balises de capture de base de données.

Ajout et suppression de balises dans Amazon RDS

Vous pouvez effectuer les opérations suivantes :

- Créez des balises lorsque vous créez une ressource, par exemple lorsque vous exécutez la AWS CLI command`create-db-instance`.
- Ajoutez des balises à une ressource existante à l'aide de la commande `add-tags-to-resource`.
- Répertoriez les balises associées à une ressource spécifique à l'aide de la commande `list-tags-for-resource`.
- Mettez à jour les balises à l'aide de la commande `add-tags-to-resource`.
- Supprimez des balises d'une ressource à l'aide de la commande `remove-tags-from-resource`.

Les procédures suivantes montrent comment effectuer des opérations d'étiquetage courantes sur des ressources liées à des instances de base de données et à des clusters de bases de données Aurora. Notez que les balises sont mises en cache à des fins d'autorisation. C'est pourquoi, lorsque vous ajoutez ou mettez à jour des balises sur les ressources Amazon RDS, plusieurs minutes peuvent s'écouler avant que les modifications ne soient disponibles.

Console

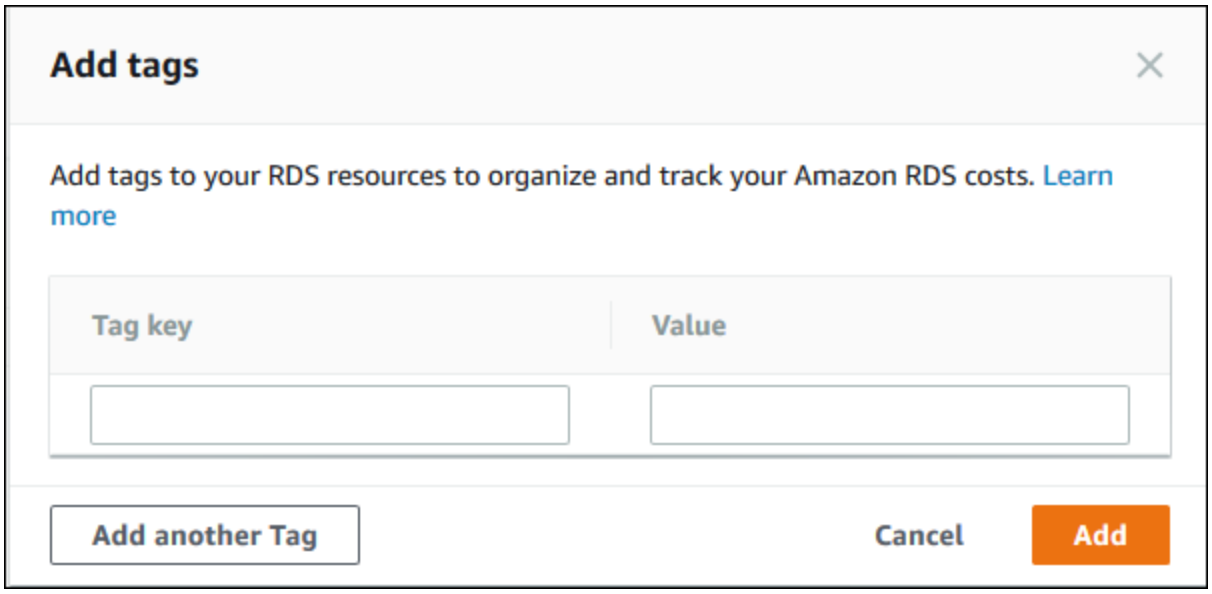
La processus de balisage d'une ressource Amazon RDS est semblable pour toutes les ressources. La procédure suivante indique comment baliser une instance de base de données Amazon RDS.

Pour ajouter une balise à une instance de base de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).

> **Note**
>
> Pour filtrer la liste des instances de base de données dans le volet Bases de données, saisissez une chaîne de texte dans Filter databases (Filtrer les bases de données). Seules les instances de base de données qui contiennent la chaîne apparaissent.

3. Sélectionnez le nom de l'instance de base de données que vous souhaitez baliser pour afficher ses détails.
4. Dans la section des détails, faites défiler jusqu'à la section Balises.
5. Choisissez Ajouter. La fenêtre Ajouter des balises s'affiche.



6. Saisissez une valeur pour Tag key (Clé de balise) et Valeur.
7. Pour ajouter une autre balise, vous pouvez choisir Ajouter une autre balise et saisir une valeur pour Tag key (Clé de balise) et Valeur.

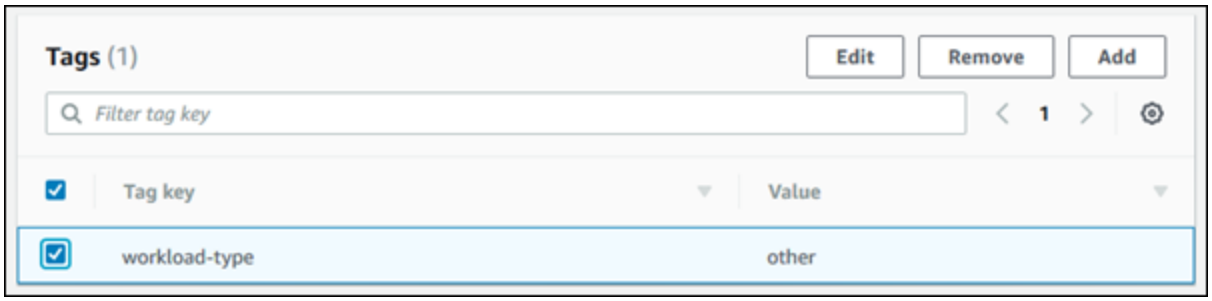
   Répétez cette étape autant de fois que nécessaire.
8. Choisissez Ajouter.

Pour supprimer une balise d’une instance de base de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).

> **Note**
>
> Pour filtrer la liste des instances de base de données dans le volet Bases de données, saisissez une chaîne de texte dans la zone Filter databases (Filtrer les bases de données). Seules les instances de base de données qui contiennent la chaîne apparaissent.

3. Sélectionnez le nom de l’instance de base de données pour afficher ses détails.
4. Dans la section des détails, faites défiler jusqu’à la section Balises.
5. Choisissez la balise que vous souhaitez supprimer.



6. Choisissez Supprimer, puis Supprimer dans la fenêtre Supprimer les balises.

AWS CLI

Vous pouvez ajouter, répertorier ou supprimer des balises pour une instance de base de données à l’aide de l AWS CLI.

- Pour ajouter une ou plusieurs balises à une ressource Amazon RDS, utilisez la AWS CLI commande `add-tags-to-resource`.
- Pour répertorier les balises d'une ressource Amazon RDS, utilisez la AWS CLI commande `list-tags-for-resource`.
- Pour supprimer une ou plusieurs balises d'une ressource Amazon RDS, utilisez la AWS CLI commande `remove-tags-from-resource`.

Pour en savoir sur la création de l'ARN requis, consultez Création d'un ARN pour Amazon RDS.

API RDS

Vous pouvez ajouter, répertorier ou supprimer des balises pour une instance de base de données à l'aide de l'API Amazon RDS.

- Pour ajouter une balise à une ressource Amazon RDS, utilisez l'opération `AddTagsToResource`.
- Pour répertorier des balises assignées à une ressource Amazon RDS, utilisez l'opération `ListTagsForResource`.
- Pour supprimer des balises d'une ressource Amazon RDS, utilisez l'opération `RemoveTagsFromResource`.

Pour en savoir sur la création de l'ARN requis, consultez Création d'un ARN pour Amazon RDS.

Lorsque vous travaillez avec XML à l'aide de l'API Amazon RDS, les balises utilisent le schéma suivant :

```
<Tagging>
      <TagSet>
          <Tag>
              <Key>Project</Key>
              <Value>Trinity</Value>
          </Tag>
          <Tag>
              <Key>User</Key>
              <Value>Jones</Value>
          </Tag>
      </TagSet>
  </Tagging>
```

Le tableau suivant fournit une liste des balises XML autorisées et leurs caractéristiques. Les valeurs pour `Key` et `Value` sont sensibles à la casse. Par exemple, `project=Trinity` et `PROJECT=Trinity` sont des balises différentes.

Elément de balisage	Description
TagSet	Un ensemble de balises contient toutes les balises assignées à une ressource Amazon RDS. Il ne peut y avoir qu'un ensemble de balises par ressource. Vous travaillez avec un TagSet uniquement via l'API Amazon RDS.
Balise	Une balise est une paire clé-valeur définie par l'utilisateur. Il peut y avoir de 1 à 50 balises dans un ensemble de balises.
Key	Une clé est le nom obligatoire de la balise. Pour les restrictions, consultez Structure des balises dans Amazon RDS. La valeur de la chaîne peut comporter de 1 à 128 caractères Unicode et elle ne peut pas être précédée de `aws:` ou de `rds:`. La chaîne peut uniquement contenir l'ensemble de lettres Unicode, de chiffres, d'espaces, « _ », « . », « / », « = », « + », « - », (expression Java : "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). Les clés doivent être propres à un ensemble de balises. Par exemple, une paire de clés ne peut pas être définie dans une balise avec la même clé mais avec des valeurs différentes, telles que project/Trinity Project/Xanadu.
Value	Une valeur est la valeur facultative de la balise. Pour les restrictions, consultez Structure des balises dans Amazon RDS. La valeur de la chaîne peut comporter de 1 à 256 caractères Unicode et elle ne peut pas être précédée de `aws:` ou de `rds:`. La chaîne peut uniquement contenir l'ensemble de lettres Unicode, de chiffres, d'espaces, « _ », « . », « / », « = », « + », « - », (expression Java : "^([\\p{L}\\p{Z}\\p{N}_.:/=+\\-]*)$"). Les valeurs comprises dans un ensemble de balises ne doivent pas nécessair ement être uniques et peuvent être null. Par exemple, vous pouvez avoir une paire clé-valeur dans un ensemble de balises composé de project/Trinity Cost-Center/ Trinity.

Didacticiel : Utilisation de balises pour spécifier les clusters de bases de données Aurora à arrêter

Supposons que vous créez un certain nombre de clusters de base de données Aurora dans un environnement de développement ou de test. Vous devez conserver tous ces clusters pendant plusieurs jours. Certains clusters exécutent des tests pendant la nuit. D'autres clusters peuvent être arrêtés pendant la nuit et redémarrés le lendemain. L'exemple suivant montre comment affecter une balise aux clusters qu'il convient d'arrêter pendant la nuit. Ensuite, l'exemple montre comment un script peut détecter les clusters qui possèdent cette balise, puis comment arrêter ces clusters. Dans cet exemple, la partie valeur de la paire clé-valeur n'a pas d'importance. La présence de la balise `stoppable` signifie que le cluster possède cette propriété définie par l'utilisateur.

Spécifier les clusters de bases de données Aurora à arrêter

1. Déterminez l'ARN d'un cluster que vous voulez désigner comme pouvant être arrêté.

   Les commandes et les API pour le balisage fonctionnent avec les ARN. De cette façon, on arrive à un fonctionnement en toute transparence entre les régions AWS, les comptes AWS et différents types de ressources qui pourraient avoir des noms courts identiques. Vous pouvez spécifier l'ARN au lieu de l'ID du cluster dans les commandes CLI qui fonctionnent sur les clusters. Remplacez le nom de votre propre cluster par *`dev-test-cluster`*. Dans les commandes suivantes qui utilisent des paramètres d'ARN, remplacez l'ARN de votre propre cluster. L'ARN inclut l'ID de votre propre compte AWS et le nom de la région AWS où se trouve votre cluster.

   ```
   $ aws rds describe-db-clusters --db-cluster-identifier dev-test-cluster \
     --query "*[].{DBClusterArn:DBClusterArn}" --output text
   arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
   ```

2. Ajoutez la balise `stoppable` à ce cluster.

   Vous choisissez le nom de cette balise. Cette approche signifie que vous pouvez éviter de concevoir une convention de dénomination qui encode toutes les informations pertinentes dans les noms. Dans une telle convention, vous pouvez encoder des informations dans le nom de l'instance de base de données ou les noms d'autres ressources. Étant donné que cet exemple traite la balise comme un attribut présent ou absent, il omet la partie `Value=` du paramètre `--tags`.

   ```
   $ aws rds add-tags-to-resource \
   ```

```
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster \
  --tags Key=stoppable
```

3. Confirmez que la balise est présente dans le cluster.

   Ces commandes récupèrent les informations sur la balise pour le cluster au format JSON et en texte brut séparé par des tabulations.

```
$ aws rds list-tags-for-resource \
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
{
    "TagList": [
        {
            "Key": "stoppable",
            "Value": ""

        }
    ]
}
$ aws rds list-tags-for-resource \
  --resource-name arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster --output
 text
TAGLIST stoppable
```

4. Pour arrêter tous les clusters désignés comme `stoppable`, préparez une liste de tous vos clusters. Passez la liste en revue et vérifiez que chaque cluster est balisé avec l'attribut approprié.

   Cet exemple Linux utilise le scripting Shell pour enregistrer la liste des ARN de cluster dans un fichier temporaire, puis pour exécuter des commandes CLI pour chaque cluster.

```
$ aws rds describe-db-clusters --query "*[].[DBClusterArn]" --output text >/tmp/
cluster_arns.lst
$ for arn in $(cat /tmp/cluster_arns.lst)
do
  match="$(aws rds list-tags-for-resource --resource-name $arn --output text | grep
 'TAGLIST\tstoppable')"
  if [[ ! -z "$match" ]]
  then
      echo "Cluster $arn is tagged as stoppable. Stopping it now."
# Note that you can specify the full ARN value as the parameter instead of the
 short ID 'dev-test-cluster'.
      aws rds stop-db-cluster --db-cluster-identifier $arn
```

```
  fi
done

Cluster arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster is tagged as
 stoppable. Stopping it now.
{
    "DBCluster": {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "us-east-1e",
            "us-east-1c",
            "us-east-1d"
        ],
        "BackupRetentionPeriod": 1,
        "DBClusterIdentifier": "dev-test-cluster",
        ...
```

Vous pouvez exécuter un script comme celui-ci à la fin de chaque journée pour vous assurer que les clusters non essentiels sont arrêtés. Vous pouvez également planifier une tâche à l'aide d'un utilitaire tel que `cron` pour effectuer une telle vérification chaque nuit. Par exemple, vous pouvez le faire si certains clusters de bases de données restaient en cours d'exécution par erreur. Dans ce cas, vous pouvez optimiser la commande qui prépare la liste des clusters à vérifier.

La commande suivante crée une liste de vos clusters, mais uniquement ceux au statut `available`. Le script peut ignorer les clusters qui sont déjà arrêtés, car ils auront des valeurs de statut différentes telles que `stopped` ou `stopping`.

```
$ aws rds describe-db-clusters \
  --query '*[].{DBClusterArn:DBClusterArn,Status:Status}|[?Status == `available`]|[].
{DBClusterArn:DBClusterArn}' \
  --output text
arn:aws:rds:us-east-1:123456789:cluster:cluster-2447
arn:aws:rds:us-east-1:123456789:cluster:cluster-3395
arn:aws:rds:us-east-1:123456789:cluster:dev-test-cluster
arn:aws:rds:us-east-1:123456789:cluster:pg2-cluster
```

> **Tip**
>
> Vous pouvez utiliser l'attribution de balises et la recherche de clusters qui ont ces balises pour réduire les coûts par d'autres moyens. Par exemple, prenez ce scénario avec des

clusters de base de données Aurora utilisés pour le développement et les tests. Ici, vous pouvez désigner certains clusters à supprimer à la fin de chaque journée, ou pour lesquels il faut uniquement supprimer leurs instances de base de données du lecteur. Vous pouvez également désigner celles pour lesquelles leurs instances de base de données sont remplacées par de petites classes d'instance de base de données pendant les périodes de faible utilisation prévues.

Amazon Resource Names (ARN) dans Amazon RDS

Les ressources créées dans Amazon Web Services sont chacune identifiées de façon unique par un Amazon Resource Name (ARN). Pour certaines opérations Amazon RDS, vous devez identifier une ressource Amazon RDS de manière unique en spécifiant son ARN. Par exemple, lorsque vous créez un réplica en lecture d'instance de base de données RDS, vous devez fournir l'ARN pour l'instance de base de données source.

Pour en savoir plus sur la construction d'un ARN et l'obtention d'un ARN existant, consultez les rubriques suivantes.

Rubriques

Création d'un ARN pour Amazon RDS

Les ressources créées dans Amazon Web Services sont chacune identifiées de façon unique par un Amazon Resource Name (ARN). Vous pouvez construire un ARN pour une ressource Amazon RDS en utilisant la syntaxe suivante.

`arn:aws:rds:<region>:<account number>:<resourcetype>:<name>`

Pour les ressources du cluster global, l'ARN n'inclut pas de Région AWS :`arn:aws:rds::<account number>:global-cluster:<name>`. ARNs pour les clusters globaux n'apparaissent pas dans leAWS Management Console.

Nom de la région	Région	Point de terminaison	Protocole	
US East (Ohio)	us-east-2	rds.us-east-2.amazonaws.com	HTTPS	
		rds-fips.us-east-2.api.aws	HTTPS	
		rds.us-east-2.api.aws	HTTPS	
		rds-fips.us-east-2.amazonaws.com	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole
USA Est (Virginie du Nord)	us-east-1	rds.us-east-1.amazonaws.com	HTTPS
		rds-fips.us-east-1.api.aws	HTTPS
		rds-fips.us-east-1.amazonaws.com	HTTPS
		rds.us-east-1.api.aws	HTTPS
USA Ouest (Californie du Nord)	us-west-1	rds.us-west-1.amazonaws.com	HTTPS
		rds.us-west-1.api.aws	HTTPS
		rds-fips.us-west-1.amazonaws.com	HTTPS
		rds-fips.us-west-1.api.aws	HTTPS
US West (Oregon)	us-west-2	rds.us-west-2.amazonaws.com	HTTPS
		rds-fips.us-west-2.amazonaws.com	HTTPS
		rds.us-west-2.api.aws	HTTPS
		rds-fips.us-west-2.api.aws	HTTPS
Afrique (Le Cap)	af-south-1	rds.af-south-1.amazonaws.com	HTTPS
		rds.af-south-1.api.aws	HTTPS
Asie-Pacifique (Hong Kong)	ap-east-1	rds.ap-east-1.amazonaws.com	HTTPS
		rds.ap-east-1.api.aws	HTTPS
Asie-Pacifique (Hyderabad)	ap-south-2	rds.ap-south-2.amazonaws.com	HTTPS
		rds.ap-south-2.api.aws	HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Asie-Pacifique (Jakarta)	ap-southeast-3	rds.ap-southeast-3.amazonaws.com rds.ap-southeast-3.api.aws	HTTPS HTTPS
Asie-Pacifique (Malaisie)	ap-southeast-5	rds.ap-southeast-5.amazonaws.com	HTTPS
Asie-Pacifique (Melbourne)	ap-southeast-4	rds.ap-southeast-4.amazonaws.com rds.ap-southeast-4.api.aws	HTTPS HTTPS
Asia Pacific (Mumbai)	ap-south-1	rds.ap-south-1.amazonaws.com rds.ap-south-1.api.aws	HTTPS HTTPS
Asie-Pacifique (Nouvelle Zélande)	ap-southeast-6	rds.ap-southeast-6.amazonaws.com	HTTPS
Asie-Pacifique (Osaka)	ap-northeast-3	rds.ap-northeast-3.amazonaws.com rds.ap-northeast-3.api.aws	HTTPS HTTPS
Asia Pacific (Seoul)	ap-northeast-2	rds.ap-northeast-2.amazonaws.com rds.ap-northeast-2.api.aws	HTTPS HTTPS
Asie-Pacifique (Singapour)	ap-southeast-1	rds.ap-southeast-1.amazonaws.com rds.ap-southeast-1.api.aws	HTTPS HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Asia Pacific (Sydney)	ap-southeast-2	rds.ap-southeast-2.amazonaws.com rds.ap-southeast-2.api.aws	HTTPS HTTPS
Asie-Pacifique (Taipei)	ap-east-2	rds.ap-east-2.amazonaws.com	HTTPS
Asie-Pacifique (Thaïlande)	ap-southeast-7	rds.ap-southeast-7.amazonaws.com	HTTPS
Asie-Pacifique (Tokyo)	ap-northeast-1	rds.ap-northeast-1.amazonaws.com rds.ap-northeast-1.api.aws	HTTPS HTTPS
Canada (Centre)	ca-central-1	rds.ca-central-1.amazonaws.com rds.ca-central-1.api.aws rds-fips.ca-central-1.api.aws rds-fips.ca-central-1.amazonaws.com	HTTPS HTTPS HTTPS HTTPS
Canada-Ouest (Calgary)	ca-west-1	rds.ca-west-1.amazonaws.com rds-fips.ca-west-1.amazonaws.com	HTTPS HTTPS
Europe (Francfort)	eu-central-1	rds.eu-central-1.amazonaws.com rds.eu-central-1.api.aws	HTTPS HTTPS
Europe (Irlande)	eu-west-1	rds.eu-west-1.amazonaws.com rds.eu-west-1.api.aws	HTTPS HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Europe (Londres)	eu-west-2	rds.eu-west-2.amazonaws.com rds.eu-west-2.api.aws	HTTPS HTTPS
Europe (Milan)	eu-south-1	rds.eu-south-1.amazonaws.com rds.eu-south-1.api.aws	HTTPS HTTPS
Europe (Paris)	eu-west-3	rds.eu-west-3.amazonaws.com rds.eu-west-3.api.aws	HTTPS HTTPS
Europe (Espagne)	eu-south-2	rds.eu-south-2.amazonaws.com rds.eu-south-2.api.aws	HTTPS HTTPS
Europe (Stockholm)	eu-north-1	rds.eu-north-1.amazonaws.com rds.eu-north-1.api.aws	HTTPS HTTPS
Europe (Zurich)	eu-central-2	rds.eu-central-2.amazonaws.com rds.eu-central-2.api.aws	HTTPS HTTPS
Israël (Tel Aviv)	il-central-1	rds.il-central-1.amazonaws.com rds.il-central-1.api.aws	HTTPS HTTPS
Mexique (Centre)	mx-central-1	rds.mx-central-1.amazonaws.com	HTTPS
Moyen-Orient (Bahreïn)	me-south-1	rds.me-south-1.amazonaws.com rds.me-south-1.api.aws	HTTPS HTTPS
Moyen-Orient (EAU)	me-central-1	rds.me-central-1.amazonaws.com rds.me-central-1.api.aws	HTTPS HTTPS

Nom de la région	Région	Point de terminaison	Protocole
Amérique du Sud (São Paulo)	sa-east-1	rds.sa-east-1.amazonaws.com rds.sa-east-1.api.aws	HTTPS HTTPS
AWSGovCloud (USA Est)	us-gov-east-1	rds.us-gov-east-1.amazonaws.com rds.us-gov-east-1.api.aws	HTTPS HTTPS
AWSGovCloud (US-Ouest)	us-gov-west-1	rds.us-gov-west-1.amazonaws.com rds.us-gov-west-1.api.aws	HTTPS HTTPS

Le tableau suivant indique le format à utiliser lors de la création d'un ARN pour un type de ressource Amazon RDS particulier.

Type de ressource	Format ARN
instance de base de données	arn:aws:rds : ::db : *<region>* *<account>* *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :db:my-mysql-instance-1`
Cluster DB	arn:aws:rds : ::cluster : *<region>* *<account>* *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :cluster: my-aurora-cluster-1`
Abonnement aux événements	arn:aws:rds : ::es : *<region>* *<account>* *<name>* Par exemple :

Type de ressource	Format ARN
	`arn:aws:rds:us-east-2:123456789012:es:my-subscription`
Groupe de paramètres de base de données	arn:aws:rds : ::pg : *<region> <account> <name>* Par exemple : `arn:aws:rds:us-east-2:123456789012:pg:my-param-enable-logs`
Groupe de paramètres de cluster de bases de données	arn:aws:rds : ::cluster-pg : *<region> <account> <name>* Par exemple : `arn:aws:rds:us-east-2:123456789012:cluster-pg:my-cluster-param-timezone`
instance de base de données réservée	arn:aws:rds : ::ri : *<region> <account> <name>* Par exemple : `arn:aws:rds:us-east-2:123456789012:ri:my-reserved-postgresql`
Groupe de sécurité de base de données	arn:aws:rds : ::secgrp : *<region> <account> <name>* Par exemple : `arn:aws:rds:us-east-2:123456789012:secgrp:my-public`
Instantané de base de données automatique	arn:aws:rds : :snapshot:rds : *<region> <account> <name>* Par exemple : `arn:aws:rds:us-east-2:123456789012:snapshot:rds:my-mysql-db-2019-07-22-07-23`

Type de ressource	Format ARN
Instantané de cluster de bases de données automatique	arn:aws:rds : ::cluster-snapshot:rds : *<region>* *<account>* *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot:rds: my-aurora-cluster-2019-07-22-16-16`
Instantané de base de données manuel	arn:aws:rds : ::snapshot : *<region>* *<account>* *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :snapshot: my-mysql-db-snap`
Instantané de cluster de bases de données manuel	arn:aws:rds : ::cluster-snapshot : *<region>* *<account>* *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :cluster-snapshot: my-aurora-cluster-snap`
Groupe de sous-réseaux de base de données	arn:aws:rds : ::subgrp : *<region>* *<account>* *<name>* Par exemple : `arn:aws:rds: us-east-2 :123456789012 :subgrp:my-subnet-10`
Cluster mondial	arn:aws:rds : :global-cluster : *<account>* *<name>* Par exemple : `arn:aws:rds:: 123456789012 :global-cluster: my-aurora-global-cluster-1`

Obtention d’un ARN existant pour Amazon RDS

Vous pouvez obtenir l'ARN d'une ressource RDS en utilisant l'APIAWS Management Console, AWS Command Line Interface (AWS CLI) ou RDS.

Console

Pour obtenir un ARN auprès duAWS Management Console, accédez à la ressource pour laquelle vous souhaitez un ARN et consultez les détails de cette ressource.

Par exemple, vous pouvez obtenir l’ARN d’un cluster de bases de données dans l’onglet Configuration des détails de ce cluster.
ARN du cluster de bases de données.

AWS CLI

Pour obtenir un ARN à partir du AWS CLI pour une ressource RDS particulière, vous devez utiliser la `describe` commande correspondant à cette ressource. Le tableau suivant présente chaque AWS CLI commande, ainsi que la propriété ARN utilisée avec la commande pour obtenir un ARN.

AWS CLIcommande	Propriété d’ARN
describe-event-subscriptions	EventSubscriptionArn
describe-certificates	CertificateArn
describe-db-parameter-groups	DBParameterGroupArn
describe-db-cluster-parameter-groupes	DBClusterParameterGroupArn
describe-db-instances	DBInstanceArn
describe-db-security-groups	DBSecurityGroupArn
describe-db-snapshots	DBSnapshotArn
describe-events	SourceArn
describe-reserved-db-instances	DBInstanceArne réservée

AWS CLIcommande	Propriété d'ARN
describe-db-subnet-groups	DBSubnetGroupArn
describe-db-clusters	DBClusterArn
describe-db-cluster-snapshots	DBClusterSnapshotArn

Par exemple, la AWS CLI commande suivante obtient l'ARN d'une instance de base de données.

Example

Pour Linux, macOS ou Unix :

```
aws rds describe-db-instances \
--db-instance-identifier DBInstanceIdentifier \
--region us-west-2 \
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

Pour Windows :

```
aws rds describe-db-instances ^
--db-instance-identifier DBInstanceIdentifier ^
--region us-west-2 ^
--query "*[].{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceArn:DBInstanceArn}"
```

La sortie de cette commande se présente comme suit :

```
[
    {
        "DBInstanceArn": "arn:aws:rds:us-west-2:account_id:db:instance_id",
        "DBInstanceIdentifier": "instance_id"
    }
]
```

API RDS

Pour obtenir un ARN pour une ressource RDS particulière, vous pouvez appeler les opérations d'API RDS suivantes et utiliser les propriétés d'ARN illustrées ci-après.

Opération d'API RDS	Propriété d'ARN
DescribeEventSubscriptions	EventSubscriptionArn
DescribeCertificates	CertificateArn
Décrire DBParameter les groupes	DBParameterGroupArn
DécrireDBClusterParameterGroups	DBClusterParameterGroupArn
DécrireDBInstances	DBInstanceArn
Décrire DBSecurity les groupes	DBSecurityGroupArn
DécrireDBSnapshots	DBSnapshotArn
DescribeEvents	SourceArn
DescribeReservedDBInstances	DBInstanceArne réservée
Décrire DBSubnet les groupes	DBSubnetGroupArn
DécrireDBClusters	DBClusterArn
Décrire les DBCluster instantanés	DBClusterSnapshotArn

Mises à jour d'Amazon Aurora

Amazon Aurora publie régulièrement des mises à jour. Ces mises à jour sont appliquées aux clusters de bases de données Amazon Aurora durant les fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour. Comme les mises à jour nécessitent un redémarrage de la base de données, vous rencontrerez 20 à 30 secondes d'indisponibilité. À l'issue de ce délai, vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter ou modifier vos paramètres de créneau de maintenance dans AWS Management Console.

> **Note**
>
> Le temps requis pour redémarrer votre instance de base de données dépend du processus de récupération sur incident, de l'activité de la base de données au moment du redémarrage et du comportement de votre moteur de base de données spécifique. Pour améliorer le délai de redémarrage, nous vous recommandons de réduire l'activité de base de donnée autant que possible pendant le processus de redémarrage. Cela a pour effet de réduire l'activité de restauration pour les transactions en transit.

Pour plus d'informations sur les mises à jour du système d'exploitation pour Amazon Aurora, consultez Mises à jour du système d'exploitation pour les clusters de bases de données Aurora.

Certaines mises à jour sont spécifiques à un moteur de base de données pris en charge par Aurora. Pour plus d'informations sur les mises à jour de moteur de base de données, reportez-vous au tableau suivant.

Moteur de base de données	Mises à jour
Amazon Aurora MySQL	Consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL
Amazon Aurora PostgreSQL	Consultez Mises à jour du moteur de base de données pour Amazon Aurora PostgreSQL

Identification de votre version d’Amazon Aurora

Amazon Aurora inclut certaines fonctions qui sont générales d’Aurora et disponibles pour tous les clusters de bases de données Aurora. Aurora inclut d’autres fonctions spécifiques d’un moteur de base de données particulier qu’Aurora prend en charge. Ces fonctions sont uniquement disponibles pour les clusters de bases de données Aurora qui utilisent ce moteur de base de données, comme Aurora PostgreSQL.

Une instance de base de données Aurora a deux numéros de version, le numéro de version Aurora et le numéro de version du moteur de base de données Les numéros de version d’Aurora utilisent le format suivant.

```
<major version>.<minor version>.<patch version>
```

Pour obtenir le numéro de version d’Aurora d’une instance de base de données Aurora utilisant un moteur de bases de données particulier, utilisez l’une des requêtes suivantes.

Moteur de base de données	Requêtes
Amazon Aurora MySQL	`SELECT AURORA_VERSION();` `SHOW @@aurora_version;`
Amazon Aurora PostgreSQL	`SELECT AURORA_VERSION();`

Support étendu Amazon RDS avec Amazon Aurora

Le support étendu RDS vous permet de continuer à exécuter votre base de données sur une version majeure du moteur au-delà de la date de fin de support standard Aurora moyennant un coût supplémentaire.

Vous ne pouvez inscrire une base de données au support étendu RDS qu'en activant le support étendu RDS lorsque vous créez ou restaurez une instance de base de données pour la première fois. L'état d'inscription au support étendu RDS ne peut pas être mis à jour sur les instances de base de données existantes, à moins qu'elles ne soient restaurées.

Si le support étendu RDS a été activé au moment de la création ou de la restauration d'une instance de base de données, Amazon Aurora inscrira automatiquement cette instance de base de données au support étendu RDS après la date de fin du support standard Aurora. L'inscription automatique au support étendu RDS ne modifie pas le moteur de base de données et n'a aucun impact sur la durée de fonctionnement ou les performances de votre instance de base de données.

Le support étendu RDS fournit les mises à jour et le support technique suivants :

- Mises à jour de sécurité pour les CVE critiques et élevées pour votre instance de base de données ou votre cluster de bases de données, y compris le moteur de base de données
- Corrections de bogues et correctifs pour les problèmes critiques
- Possibilité d'ouvrir des cas de support et de recevoir une assistance au dépannage dans le cadre du contrat de niveau de service Amazon RDS standard

Cette offre payante vous permet de disposer d'un délai supplémentaire pour effectuer la mise à niveau vers une version majeure du moteur prise en charge. Par exemple, la date de fin du support standard Aurora pour Aurora MySQL version 2 est le 31 octobre 2024. Toutefois, vous n'êtes pas prêt à effectuer une mise à niveau manuelle vers Aurora MySQL version 3 avant cette date. Dans ce cas, Amazon Aurora inscrit automatiquement votre cluster au support étendu RDS le 31 octobre 2024 et vous pouvez continuer à exécuter Aurora MySQL version 2. À compter du 1er décembre 2024, Amazon Aurora vous facture automatiquement le support étendu RDS.

Le support étendu RDS est disponible jusqu'à 3 ans après la date de fin de vie communautaire d'une version majeure de moteur (3 ans et 4 mois pour Aurora MySQL version 2). Après cette période, si vous n'avez pas encore mis à niveau votre version majeure de moteur vers une version prise en charge, Amazon Aurora mettra automatiquement à niveau votre version majeure de moteur. Nous

vous recommandons de mettre à niveau vers une version majeure prise en charge du moteur dès que possible.

Pour plus d'informations sur les dates de fin du support standard Aurora et les dates de fin du support étendu RDS, consultez le calendrier des versions majeures d'Aurora MySQL et le calendrier des versions majeures d'Aurora PostgreSQL.

Rubriques

Présentation du support étendu Amazon RDS

Après la date de fin du support standard Aurora, si vous n'avez pas désactivé le support étendu RDS lors de la création ou de la restauration de vos instances de base de données, Amazon Aurora les inscrira automatiquement au support étendu RDS. Aurora met automatiquement à niveau votre instance de base de données vers la dernière version mineure publiée avant la date de fin du support standard Aurora, si ce n'est pas déjà le cas. Amazon Aurora ne mettra à niveau votre version mineure qu'après la date de fin du support standard Aurora pour votre version majeure de moteur.

Vous pouvez créer de nouvelles bases de données avec les versions majeures de moteur qui ont atteint la date de fin du support standard Aurora. Aurora inscrit automatiquement ces nouvelles bases de données au support étendu RDS et vous facture cette offre.

Si vous effectuez une mise à niveau vers un moteur toujours couvert par le support standard Aurora avant la date de fin du support standard Aurora, Amazon Aurora n'inscrira pas votre moteur au support étendu RDS.

Si vous tentez de restaurer un instantané d'une base de données compatible avec moteur dont le support standard Aurora est expiré et qui n'est pas couvert par le support étendu RDS, Amazon Aurora mettra automatiquement à jour l'instantané pour le rendre compatible avec la dernière version du moteur encore couverte par le support standard Aurora. Si la restauration échoue, Amazon Aurora inscrira automatiquement votre moteur au support étendu RDS avec une version compatible avec l'instantané.

Vous pouvez mettre fin à votre inscription au support étendu RDS à tout moment. Pour mettre fin à l'inscription, mettez à niveau chaque moteur inscrit vers une version plus récente toujours couverte par le support standard Aurora. La fin de l'inscription au support étendu RDS prendra effet le jour où vous effectuerez la mise à niveau vers une version plus récente du moteur toujours couverte par le support standard Aurora.

Pour plus d'informations sur les dates de fin du support standard Aurora et les dates de fin du support étendu RDS, consultez le calendrier des versions majeures d'Aurora MySQL et le calendrier des versions majeures d'Aurora PostgreSQL.

Frais de support étendu Amazon RDS

Les moteurs inscrits au support étendu RDS seront facturés dès le jour suivant la fin du support standard Aurora. Pour connaître la date de fin du support standard Aurora, consultez Versions majeures d'Amazon Aurora.

Les frais supplémentaires du support étendu RDS cessent automatiquement lorsque vous effectuez l'une des actions suivantes :

- Effectuer une mise à niveau vers une version du moteur couverte par le support standard.
- Supprimer la base de données qui exécute une version majeure après la date de fin du support standard Aurora.

Les frais seront de nouveau appliqués si la version de votre moteur cible relève du support étendu RDS à l'avenir.

Par exemple, Aurora PostgreSQL 11 ont été inclus dans le support étendu le 1er mars 2024, mais la facturation n'a commencé que le 1er avril 2024. Votre base de données Aurora PostgreSQL 11 a été mise à niveau vers Aurora PostgreSQL 12 le 30 avril 2024. Seuls 30 jours de support étendu sur Aurora PostgreSQL 11 vous seront facturés. Vous continuez à exécuter Aurora PostgreSQL 12 sur

cette instance de base de données après la fin du support standard RDS, le 28 février 2025. Dès le 1er mars 2025, votre base de données sera de nouveau facturée dans le cadre du support étendu RDS.

Pour plus d'informations, consultez Tarification d'Amazon Aurora.

Éviter les frais liés au support étendu Amazon RDS

Pour éviter toute facturation au titre du support étendu RDS, empêchez Aurora de créer ou de restaurer un cluster de bases de données Aurora ou un cluster global après la date de fin du support standard Aurora. Pour ce faire, utilisez l'AWS CLI ou l'API RDS.

Dans l'AWS CLI, indiquez `open-source-rds-extended-support-disabled` pour l'option `--engine-lifecycle-support`. Dans l'API RDS, indiquez la valeur `open-source-rds-extended-support-disabled` pour le paramètre `LifeCycleSupport`. Pour plus d'informations, consultez Création d'un cluster de bases de données Aurora ou d'un cluster global ou Restauration d'un cluster de bases de données Aurora ou d'un cluster global.

Versions bénéficiant du support étendu Amazon RDS

Le support étendu RDS est disponible pour Aurora MySQL et pour Aurora PostgreSQL. Pour plus d'informations, consultez Versions majeures d'Amazon Aurora.

Le support étendu RDS n'est disponible que sur certaines versions mineures. Les versions mineures deviennent compatibles avec le support étendu RDS uniquement après que la version majeure correspondante a atteint sa fin de vie communautaire. Pour plus d'informations, consultez Calendrier de publication d'Aurora MySQL dans Notes de mise à jour d'Aurora MySQL et Calendrier de publication d'Aurora PostgreSQL dans Notes de mise à jour d'Aurora PostgreSQL.

Le support étendu RDS n'est disponible que sur Aurora Serverless v2. Il n'est pas disponible sur Aurora Serverless v1.

Vous pouvez également consulter les informations sur les dates de support des versions du moteur à l'aide de l'AWS CLI ou de l'API RDS. Pour plus d'informations, consultez Affichage des dates de support pour les versions de moteur dans Support étendu Amazon RDS.

Responsabilités d’Amazon Aurora et du client dans le cadre du support étendu Amazon RDS

Le contenu suivant décrit les responsabilités d’Amazon Aurora et les vôtres dans le cade du support étendu RDS.

Rubriques

Responsabilités d’Amazon Aurora

Après la date de fin du support standard Aurora, Amazon Aurora fournira des correctifs, des correctifs de bogues et des mises à niveau pour les moteurs inscrits au support étendu RDS. Cette disposition est valable jusqu’à trois ans, ou jusqu’à l’arrêt de l’utilisation des moteurs, selon la première éventualité.

Les correctifs concerneront les vulnérabilités CVE Critique et Élevée, conformément aux niveaux de gravité CVSS de la National Vulnerability Database (NVD). Pour plus d’informations, consultez Métriques de vulnérabilité (langue française non garantie).

Vos responsabilités

Vous êtes responsable de l’application des correctifs, des correctifs de bogues et des mises à niveau fournis pour les les clusters de bases de données Aurora ou les clusters globaux inscrits au support étendu RDS. Amazon Aurora se réserve le droit de modifier, de remplacer ou de retirer ces correctifs, correctifs de bogues et mises à niveau à tout moment. Si un correctif est nécessaire pour résoudre des problèmes de sécurité ou de stabilité critiques, Amazon Aurora se réserve le droit de mettre à jour vos vos clusters de bases de données Aurora ou vos clusters globaux avec le correctif, ou d’exiger que vous l’installiez vous-même.

Vous êtes également responsable de la mise à niveau de votre moteur vers une version plus récente avant la date de fin du support étendu RDS. Le support étendu RDS prend généralement fin trois ans après la date de fin de vie de la communauté. . Pour connaître la date de fin du support étendu RDS de la version majeure de votre moteur de base de données, consultez Versions majeures d’Amazon Aurora.

Si vous ne mettez pas à niveau votre moteur, après la date de fin du support étendu RDS, Amazon Aurora mettra automatiquement à niveau votre moteur vers une version plus récente toujours couverte par le support standard Aurora. Si la mise à niveau échoue, Amazon Aurora se réserve le droit de supprimer l'le cluster de bases de données Aurora ou le cluster global qui exécute le moteur après la date de fin du support standard Aurora. Toutefois, avant d'effectuer cette opération, Amazon Aurora conservera les données issues de ce moteur.

Création d'un cluster de bases de données Aurora ou d'un cluster global avec le support étendu Amazon RDS

Lorsque vous créez un cluster de bases de données Aurora ou un cluster global, sélectionnez Activer le support étendu RDS dans la console ou utilisez l'option Support étendu dans l'AWS CLI ou le paramètre de l'API RDS. Lorsque vous inscrivez un cluster de bases de données Aurora ou un cluster global dans le support étendu Amazon RDS, celui-ci est inscrit de façon permanente au support étendu RDS pendant toute la durée de vie de du cluster de bases de données Aurora ou du cluster global.

Si vous utilisez la console, vous devez sélectionner Activer le support étendu RDS. Le paramètre n'est pas sélectionné par défaut.

Si le paramètre Support étendu RDS n'est pas indiqué dans l'AWS CLI ou dans l'API RDS, il est activé par défaut par Amazon RDS. Si vous utilisez l'automatisation via CloudFormationou d'autres services, ce comportement par défaut assure la disponibilité de votre base de données après la fin du support standard Aurora.

Vous pouvez empêcher l'inscription au support étendu RDS en utilisant l'AWS CLIou l'API RDS pour créer un cluster de bases de données Aurora ou un cluster global.

Rubriques

- Comportement du support étendu RDS
- Considérations relatives au support étendu RDS
- Créez un cluster de bases de données Aurora ou un cluster global avec le support étendu RDS

Comportement du support étendu RDS

Le tableau suivant décrit le comportement observé lorsqu'une version majeure du moteur atteint la fin du support standard Aurora.

Statut du support étendu RDS*	Comportement
Activé	Amazon RDS vous facture le support étendu RDS.
Désactivé	Amazon RDS met à niveau votre cluster de bases de données Aurora ou cluster global vers une version de moteur prise en charge. Cette mise à niveau s'effectue à la date de fin du support standard Aurora ou peu après.

* La console RDS affiche Oui ou Non pour l'état du support étendu RDS, tandis que l'AWS CLI ou l'API RDS renvoie les valeurs correspondantes `open-source-rds-extended-support` ou `open-source-rds-extended-support-disabled`.

Considérations relatives au support étendu RDS

Avant de créer un cluster de bases de données Aurora ou un cluster global, tenez compte des points suivants :

- Une fois la date de fin du support standard Aurora dépassée, vous pouvez bloquer la création d'un cluster de bases de données Aurora ou d'un cluster global, pour éviter toute facturation du support étendu RDS. Pour ce faire, utilisez l'AWS CLI ou l'API RDS. Dans l'AWS CLI, indiquez `open-source-rds-extended-support-disabled` pour l'option `--engine-lifecycle-support`. Dans l'API RDS, indiquez la valeur `open-source-rds-extended-support-disabled` pour le paramètre `LifeCycleSupport`. Si vous indiquez `open-source-rds-extended-support-disabled` et que la date de fin du support standard de Aurora est dépassée, la création d'un cluster de bases de données Aurora ou d'un cluster global échouera systématiquement.
- Le support étendu RDS est défini au niveau du cluster. Les membres d'un cluster auront toujours le même paramètre pour le support étendu RDS dans la console RDS, `--engine-lifecycle-support` dans l'AWS CLI et `EngineLifecycleSupport` dans l'API RDS.

Pour plus d'informations, consultez Versions d'Amazon Aurora.

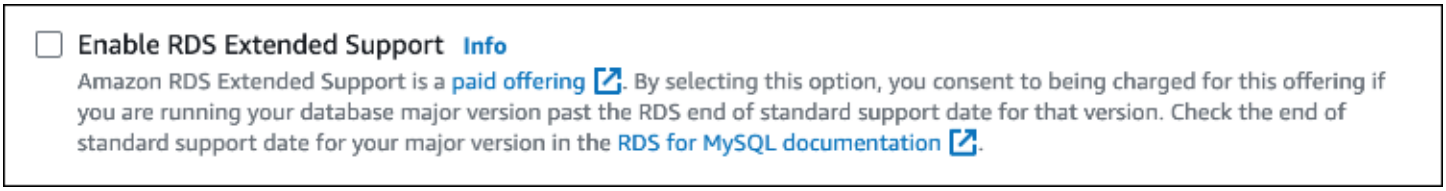
Créez un cluster de bases de données Aurora ou un cluster global avec le support étendu RDS

Vous pouvez créer un cluster de bases de données Aurora ou un cluster global avec une version de RDS couverte par le support étendu à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API RDS.

Console

Lorsque vous créez un cluster de bases de données Aurora ou un cluster global, dans la section Options de moteur, sélectionnez Activer le support étendu RDS. Ce paramètre n'est pas sélectionné par défaut.

L'image suivante montre le paramètre Activer le support étendu RDS :



AWS CLI

Lorsque vous exécutez la commande AWS CLI create-db-cluster ou create-global-cluster, sélectionnez le support étendu RDS en indiquant `open-source-rds-extended-support` pour l'option `--engine-lifecycle-support`. Cette option est définie par défaut sur `open-source-rds-extended-support`.

Pour empêcher la création d'un cluster de bases de données Aurora ou d'un cluster global après la date de fin du support standard Aurora, indiquez `open-source-rds-extended-support-disabled` pour l'option `--engine-lifecycle-support`. Ce faisant, vous éviterez tous les frais associés au support étendu RDS.

API RDS

Lorsque vous utilisez l'opération CreateDBCluster ou CreateGlobalCluster de l'API Amazon RDS, sélectionnez le support étendu RDS en définissant le paramètre `EngineLifecycleSupport` sur `open-source-rds-extended-support`. Par défaut, ce paramètre est défini sur `open-source-rds-extended-support`.

Pour empêcher la création d'un cluster de bases de données Aurora ou d'un cluster global après la date de fin du support standard Aurora, indiquez `open-source-rds-extended-support-`

`disabled` pour le paramètre `EngineLifecycleSupport`. Ce faisant, vous éviterez tous les frais associés au support étendu RDS.

Pour plus d'informations, consultez les rubriques suivantes :

- Pour créer un cluster de bases de données Aurora, suivez les instructions relatives à votre moteur de base de données dans Création d'un cluster de bases de données Amazon Aurora.
- Pour créer un cluster global, suivez les instructions relatives à votre moteur de base de données dans Création d'une base de données Amazon Aurora globale.

Affichage de l'inscription de vos clusters de bases de données Aurora ou de vos clusters globaux dans le support étendu Amazon RDS

Vous pouvez consulter l'inscription de vos clusters de bases de données Aurora ou de vos clusters globaux dans le support étendu RDS à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API RDS.

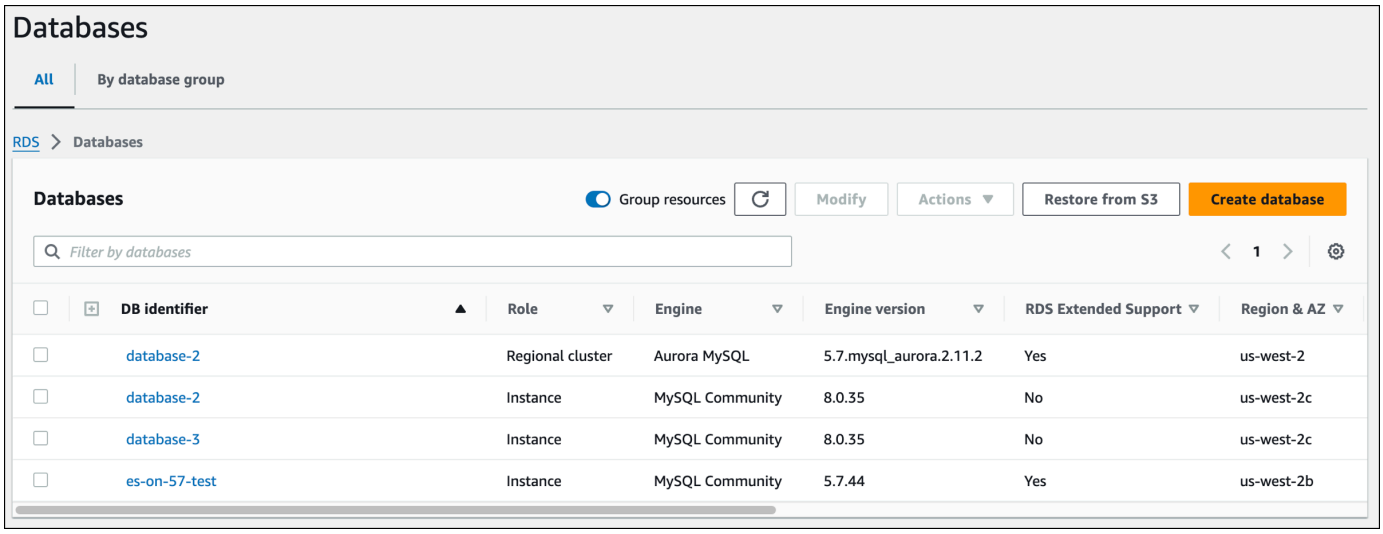
> **Note**
>
> La colonne Support étendu RDS dans la console, l'option `--engine-lifecycle-support`dans l'AWS CLI, et le paramètre `EngineLifecycleSupport` dans l'API RDS indiquent uniquement l'inscription au support étendu RDS. Le support étendu RDS ne vous est facturé qu'une fois le support standard Aurora de la version de votre moteur de base de données terminé. Pour plus d'informations, consultez Versions majeures d'Amazon Aurora. Prenons l'exemple d'une base de données Aurora PostgreSQL 11 inscrite au support étendu RDS. Amazon RDS a commencé à vous facturer le support étendu RDS pour cette base de données le 1er avril 2024. Le 31 juillet 2024, vous avez mis à niveau cette base de données vers Aurora PostgreSQL 12. L'état du support étendu RDS pour cette base de données reste activé. Cependant, la facturation du support étendu RDS pour cette base de données a été suspendue, car Aurora PostgreSQL 12 n'avait pas encore atteint la fin du support standard Aurora. Amazon RDS ne vous facturera pas le support étendu RDS pour cette base de données avant le 1er mars 2025, date à laquelle le support standard Aurora prendra fin pour RDS pour Aurora PostgreSQL 12.

Console

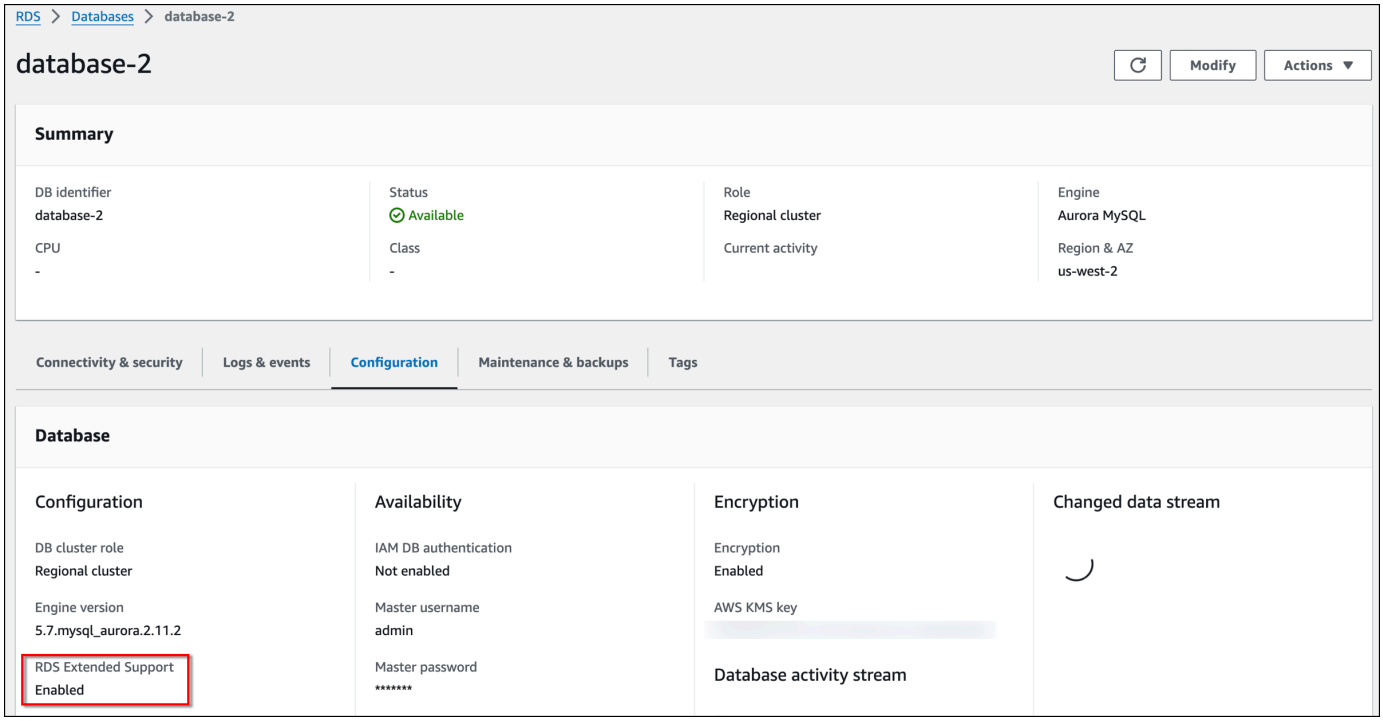
Pour consulter l'inscription de vos clusters de bases de données Aurora ou de vos clusters globaux dans le support étendu RDS

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données. La valeur figurant sous Support étendu RDS indique si un cluster de bases de données Aurora ou un cluster global est inscrit au support étendu RDS. Si aucune valeur n'apparaît, cela signifie que le support étendu RDS n'est pas disponible pour votre base de données.

> **Tip**
>
> Si la colonne Support étendu RDS n'apparaît pas, choisissez l'icône Préférences, puis activez le Support étendu RDS.



3. Vous pouvez également consulter l'inscription dans l'onglet Configuration de chaque base de données. Choisissez une base de données sous identifiant de base de données. Dans l'onglet Configuration, consultez Support étendu pour vérifier si la base de données est inscrite ou non.



AWS CLI

Pour consulter l'inscription de vos bases de données au support étendu RDS à l'aide de l'AWS CLI, exécutez la commande describe-db-clusters, describe-global-clusters.

Si le support étendu RDS est disponible pour une base de données, la réponse inclut le paramètre `EngineLifecycleSupport`. La valeur `open-source-rds-extended-support` indique si un cluster de bases de données Aurora ou un cluster global est inscrit au support étendu RDS. La valeur `open-source-rds-extended-support-disabled` indique que l'inscription d'un cluster de bases de données Aurora ou d'un cluster global au support étendu RDS a été désactivée.

Exemple

La commande suivante renvoie des informations pour tous vos clusters de bases de données Aurora :

```
aws rds describe-db-clusters
```

La réponse suivante indique qu'un moteur Aurora PostgreSQL exécuté sur le cluster de bases de données Aurora `database-1` est inscrit au support étendu RDS :

```
{
```

```
    "DBClusterIdentifier": "database-1",
    ...
    "Engine": "aurora-postgresql",
    ...
    "EngineLifecycleSupport": "open-source-rds-extended-support"
}
```

API RDS

Pour consulter l'inscription de vos bases de données au support étendu à l'aide de l'API Amazon RDS, utilisez l'opération DescribeDBClusters ou DescribeGlobalClusters.

Si le support étendu RDS est disponible pour une base de données, la réponse inclut le paramètre `EngineLifecycleSupport`. La valeur `open-source-rds-extended-support` indique si un cluster de bases de données Aurora ou un cluster global est inscrit au support étendu RDS. La valeur `open-source-rds-extended-support-disabled` indique que l'inscription d'un cluster de bases de données Aurora ou d'un cluster global au support étendu RDS a été désactivée.

Affichage des dates de support pour les versions de moteur dans Support étendu Amazon RDS

Découvrez comment afficher les informations relatives aux dates de support pour les versions de moteur dans les clusters de bases de données Aurora ou les clusters globaux dans le support étendu Amazon RDS à l'aide de l'AWS CLI ou de l'API RDS. Ces informations peuvent vous aider à planifier les mises à niveau.

Les commandes AWS CLI et les opérations de l'API RDS renvoient les dates de début et de fin du support standard Aurora et le support étendu RDS. Ces dates figurent également dans les tableaux des versions majeures du moteur. Pour plus d'informations, consultez Versions majeures d'Amazon Aurora.

AWS CLI

Pour consulter les dates de début et de fin du support standard Aurora et du support étendu RDS pour les versions majeures de votre moteur à l'aide de l'AWS CLI, exécutez la commande describe-db-major-engine-versions.

Cette commande renvoie les paramètres pertinents suivants :

- `SupportedEngineLifecycles` : ce paramètre est un tableau d'objets comprenant `LifecycleSupportName`, `LifecycleSupportStartDate` et `LifecycleSupportEndDate`.
- `LifecycleSupportName` : ce paramètre indique le type de support dont bénéficie la version du moteur : support standard Aurora (`open-source-rds-standard-support`) ou support étendu RDS (`open-source-rds-extended-support`).
- `LifecycleSupportStartDate` : ce paramètre indique la date de début du support standard Aurora ou du support étendu RDS pour la version majeure du moteur, en fonction de la valeur `LifecycleSupportName`.
- `LifecycleSupportEndDate` : ce paramètre indique la date de fin du support standard Aurora ou du support étendu RDS pour la version majeure du moteur, en fonction de la valeur `LifecycleSupportName`.

Exemple

L'exemple de réponse indique les dates de début et de fin des cycles de vie du moteur pris en charge `open-source-rds-standard-support` et `open-source-rds-extended-support` pour Aurora MySQL version 2 (MySQL 5.7). Le support étendu RDS est disponible pour Aurora MySQL version 2 (MySQL 5.7).

```
{
    "DBMajorEngineVersions": [
        {
            "Engine": "aurora-mysql",
            "MajorEngineVersion": "5.7",
            "SupportedEngineLifecycles": [
                {
                    "LifecycleSupportName": "open-source-rds-standard-support",
                    "LifecycleSupportStartDate": "2018-02-06T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2024-10-31T23:59:59.999000+00:00"
                },
                {
                    "LifecycleSupportName": "open-source-rds-extended-support",
                    "LifecycleSupportStartDate": "2024-11-01T00:00:00+00:00",
                    "LifecycleSupportEndDate": "2027-02-28T23:59:59.999000+00:00"
                }
            ]
        },
        ...
    ]
```

```
}
```

API RDS

Pour consulter les dates de début et de fin du support standard Aurora et du support étendu RDS pour les versions majeures de votre moteur à l'aide de l'API RDS, utilisez l'opération DescribeDBMajorEngineVersions.

Si le support étendu RDS est disponible pour une version du moteur, la réponse inclut le paramètre `SupportedEngineLifeCycles` sous la forme d'un tableau contenant deux objets. L'un des objets inclut les dates de début et de fin du support standard Aurora. Le second objet inclut les dates de début et de fin du support étendu RDS.

Si le support étendu RDS n'est pas disponible pour une version du moteur, la réponse inclut uniquement le paramètre `SupportedEngineLifeCycles` sous la forme d'un tableau contenant un seul objet. Cet objet inclut les dates de début et de fin du support standard Aurora.

Restauration d'un cluster de bases de données Aurora ou d'un cluster global avec le support étendu Amazon RDS

Lorsque vous restaurez un cluster de bases de données Aurora ou un cluster global, sélectionnez Activer le support étendu RDS dans la console ou utilisez l'option Support étendu dans l'AWS CLI ou le paramètre de l'API RDS. Lorsque vous inscrivez un cluster de bases de données Aurora ou un cluster global au support étendu RDS, celui-ci est inscrit de façon permanente au support étendu RDS pendant toute la durée de vie de du cluster de bases de données Aurora ou du cluster global.

La valeur par défaut du paramètre Support étendu RDS varie selon l'outil utilisé pour restaurer la base de données : la console, l'AWS CLI ou l'API RDS. Si vous utilisez la console sans sélectionner Activer le support étendu RDS et que la version majeure du moteur que vous restaurez n'est plus couverte par le support standard Aurora, Amazon Aurora met automatiquement à jour votre instance de base de données vers une version plus récente du moteur. Si vous utilisez l'AWS CLI ou l'API RDS sans indiquer le paramètre Support étendu RDS, Amazon RDS active par défaut le support étendu RDS. Si vous utilisez l'automatisation via CloudFormation ou d'autres services, ce comportement par défaut assure la disponibilité de votre base de données après la fin du support standard Aurora. Vous pouvez désactiver le support étendu RDS à l'aide de l'AWS CLI ou de l'API RDS.

Rubriques

Comportement du support étendu RDS

Le tableau suivant présente le comportement du service lorsqu'une version majeure du moteur d'une d'un cluster de bases de données Aurora ou d'un cluster global que vous restaurez arrive à la fin du support standard Aurora.

Statut du support étendu RDS*	Comportement
Activé	Amazon RDS vous facture le support étendu RDS.
Désactivé	Une fois la restauration terminée, Amazon RDS procède automatiq uement à la mise à niveau de votre cluster de bases de données Aurora ou votre cluster global vers une version de moteur plus récente (lors d'une prochaine fenêtre de maintenance).

* La console RDS affiche Oui ou Non pour l'état du support étendu RDS, tandis que l'AWS CLI ou l'API RDS renvoie les valeurs correspondantes `open-source-rds-extended-support` ou `open-source-rds-extended-support-disabled`.

Considérations relatives au support étendu RDS

Avant de restaurer un cluster de bases de données Aurora ou un cluster global, tenez compte des points suivants :

- Une fois le support standard Aurora expiré, la restauration d'un cluster de bases de données Aurora ou d'un cluster global depuis Amazon S3 ne peut être effectuée qu'à l'aide de l'AWS CLI ou de l'API RDS. Utilisez l'option `--engine-lifecycle-support` de la commande AWS CLI restore-db-cluster-from-s3 ou le paramètre `EngineLifecycleSupport` de l'opération RestoreDBClusterFromS3 de l'API RDS.
- Si vous souhaitez empêcher Aurora de restaurer vos bases de données vers des versions RDS couvertes par le support étendu, indiquez `open-source-rds-extended-support-disabled`

dans l'AWS CLI ou dans l'API RDS. Ce faisant, vous éviterez tous les frais associés au support étendu RDS.

Si ce paramètre est activé, Amazon Aurora mettra automatiquement à niveau votre base de données restaurée vers une version majeure plus récente et prise en charge. Si la mise à niveau échoue lors des vérifications préalables, Amazon Aurora restaurera en toute sécurité la version du moteur RDS bénéficiant du support étendu. Cette base de données restera en mode support étendu RDS, et Amazon Aurora vous facturera le support étendu RDS jusqu'à ce que vous effectuiez manuellement la mise à niveau de votre base de données.

- Le support étendu RDS est défini au niveau du cluster. Les membres d'un cluster auront toujours le même paramètre pour le support étendu RDS dans la console RDS, `--engine-lifecycle-support` dans l'AWS CLI et `EngineLifecycleSupport` dans l'API RDS.

Pour plus d'informations, consultez Versions d'Amazon Aurora.

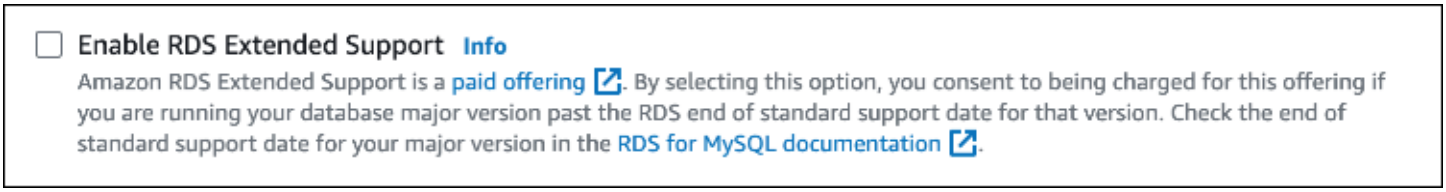
Restaurez un cluster de bases de données Aurora ou un cluster global avec le support étendu RDS

Vous pouvez restaurer un cluster de bases de données Aurora ou un cluster global avec une version de RDS couverte par le support étendu à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API RDS.

Console

Lorsque vous restaurez un cluster de bases de données Aurora ou un cluster global, sélectionnez Activer le support étendu RDS dans la section Options de moteur. Si vous ne sélectionnez pas ce paramètre et que la version majeure du moteur que vous restaurez a dépassé la fin du support standard Aurora, Amazon Aurora met automatiquement à niveau votre cluster de bases de données Aurora ou votre cluster global vers une version bénéficiant du support standard Aurora.

L'image suivante montre le paramètre Activer le support étendu RDS :



AWS CLI

Lorsque vous exécutez la commande AWS CLI restore-db-cluster-from-snapshot, sélectionnez le support étendu RDS en indiquant `open-source-rds-extended-support` pour l'option `--engine-lifecycle-support`.

Si vous souhaitez éviter les frais associés au support étendu RDS, définissez l'option `--engine-lifecycle-support` sur `open-source-rds-extended-support-disabled`. Cette option est définie par défaut sur `open-source-rds-extended-support`.

Vous pouvez également spécifier cette valeur à l'aide des commandes d'AWS CLI suivantes :

- restore-db-cluster-from-s3
- restore-db-cluster-to-point-in-time

API RDS

Lorsque vous utilisez l'opération RestoreDBClusterFromSnapshot de l'API Amazon RDS, sélectionnez le support étendu RDS en définissant le paramètre `EngineLifecycleSupport` sur `open-source-rds-extended-support`.

Si vous souhaitez éviter les frais associés au support étendu RDS, définissez le paramètre `EngineLifecycleSupport` sur `open-source-rds-extended-support-disabled`. Par défaut, ce paramètre est défini sur `open-source-rds-extended-support`.

Vous pouvez également spécifier cette valeur à l'aide des opérations de l'API RDS suivantes :

- RestoreDBClusterFromS3
- RestoreDBClusterToPointInTime

Pour plus d'informations sur la restauration d'un cluster de bases de données Aurora, suivez les instructions relatives à votre moteur de base de données dans Sauvegarde et restauration d'un cluster de bases de données Amazon Aurora.

Utilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données

Un blue/green déploiement copie un environnement de base de données de production vers un environnement intermédiaire distinct et synchronisé. En utilisant Amazon Aurora Blue/Green Deployments, vous pouvez apporter des modifications à la base de données dans l'environnement intermédiaire sans affecter l'environnement de production. Par exemple, vous pouvez mettre à niveau la version majeure ou mineure du moteur de base de données ou modifier les paramètres de la base de données dans l'environnement intermédiaire. Lorsque vous êtes prêt, vous pouvez faire de l'environnement intermédiaire le nouvel environnement de base de données de production, avec une durée d'indisponibilité généralement inférieure à une minute.

Amazon Aurora crée l'environnement intermédiaire en clonant le volume de stockage Aurora sous-jacent dans l'environnement de production. Le volume de cluster de l'environnement intermédiaire stocke uniquement les modifications incrémentielles apportées à cet environnement.

> **Note**
>
> Actuellement, Blue/Green les déploiements sont pris en charge pour Aurora MySQL, Aurora PostgreSQL et Aurora Global Database. Pour connaître la disponibilité d'Amazon RDS, consultez Utilisation des déploiements bleu/vert Amazon RDS pour les mises à jour de base de données dans le Guide de l'utilisateur Amazon RDS.

Rubriques

Présentation des (Amazon Aurora Blue/Green)

En utilisant Amazon Aurora Blue/Green Deployments, vous pouvez apporter et tester des modifications de base de données avant de les implémenter dans un environnement de production. Un déploiement bleu/vert crée un environnement intermédiaire qui copie l'environnement de production. Dans un déploiement bleu/vert, l'environnement bleu est l'environnement de production actuel. L'environnement vert est l'environnement intermédiaire et reste synchronisé avec l'environnement de production actuel.

Vous pouvez apporter des modifications au cluster de bases de données Aurora dans l'environnement vert sans affecter les charges de travail de production. Par exemple, vous pouvez mettre à niveau la version majeure ou mineure du moteur de base de données ou modifier les paramètres de la base de données dans l'environnement intermédiaire. Vous pouvez tester en profondeur les changements dans l'environnement vert. Lorsque vous êtes prêt, vous pouvez basculer les environnements pour faire de l'environnement vert le nouvel environnement de production. La bascule prend généralement moins d'une minute, sans perte de données et sans qu'il soit nécessaire de modifier les applications.

Comme l'environnement vert est une copie de la topologie de l'environnement de production, le cluster de bases de données et toutes ses instances de base de données sont copiés dans le déploiement. L'environnement vert comprend également les fonctionnalités utilisées par le cluster de bases de données, telles que les instantanés de cluster de bases de données, Performance Insights, la surveillance améliorée et Aurora Serverless v2.

> **Note**
>
> Les déploiements bleu/vert sont pris en charge pour Aurora MySQL, Aurora PostgreSQL et Aurora Global Database. Pour connaître la disponibilité d'Amazon RDS, consultez la présentation des Blue/Green déploiements Amazon RDS dans le guide de l'utilisateur Amazon RDS.

Rubriques

- Limites et considérations relatives aux (Amazon Aurora blue/green)
- Bonnes pratiques pour les (Amazon Aurora blue/green)

Disponibilité des régions et des versions

La disponibilité et la prise en charge des fonctionnalités varient selon les versions spécifiques de chaque moteur de base de données, et selon les Régions AWS. Pour de plus amples informations, veuillez consulter the section called “Déploiements bleu/vert”.

Avantages de l'utilisation des déploiements Amazon RDS Blue/Green

En utilisant les Blue/Green déploiements Amazon RDS, vous pouvez rester au courant des correctifs de sécurité, améliorer les performances des bases de données et adopter de nouvelles fonctionnalités de base de données avec des temps d'arrêt courts et prévisibles. Blue/green les déploiements réduisent les risques et les temps d'arrêt liés aux mises à jour de base de données, telles que les mises à niveau majeures ou mineures des versions du moteur.

Les déploiements bleu/vert offrent les avantages suivants :

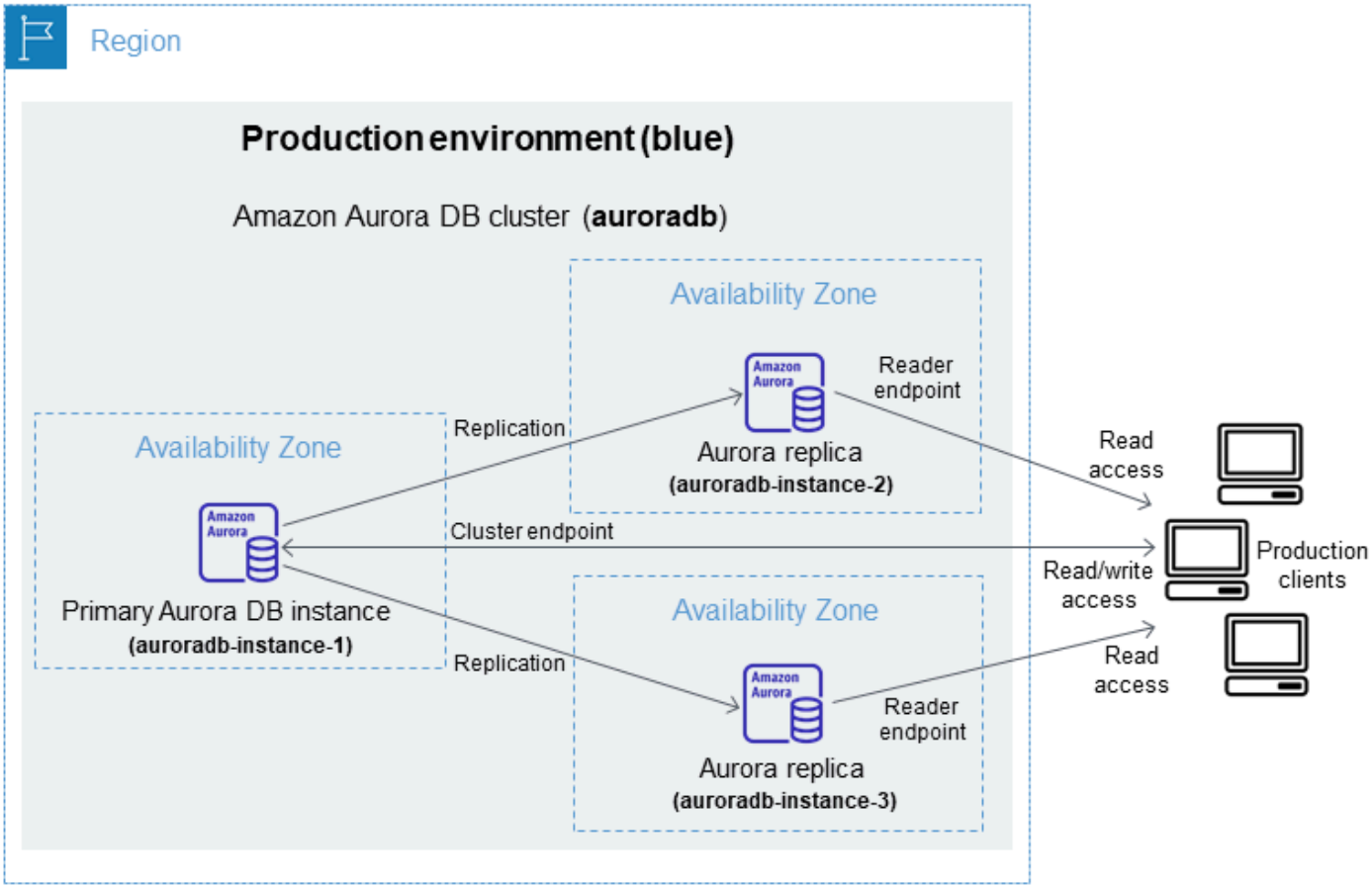
- Créez facilement un environnement intermédiaire prêt pour la production.
- Répliquez automatiquement les modifications apportées aux bases de données de l’environnement de production à l’environnement intermédiaire.
- Testez les modifications apportées aux bases de données dans un environnement intermédiaire sûr sans affecter l’environnement de production.
- Restez à jour des correctifs de base de données et des mises à jour du système.
- Mettez en œuvre et testez les nouvelles fonctionnalités de base de données.
- Basculez votre environnement intermédiaire pour en faire le nouvel environnement de production sans modifier votre application.
- Basculez en toute sécurité grâce aux barrières de protection de bascule intégrées.
- Éliminez les pertes de données pendant la bascule.
- Basculez rapidement, généralement en moins d’une minute en fonction de votre charge de travail.

Flux de travail d'un blue/green déploiement

Effectuez les étapes principales suivantes lorsque vous utilisez un blue/green déploiement pour les mises à jour du cluster de base de données Aurora.

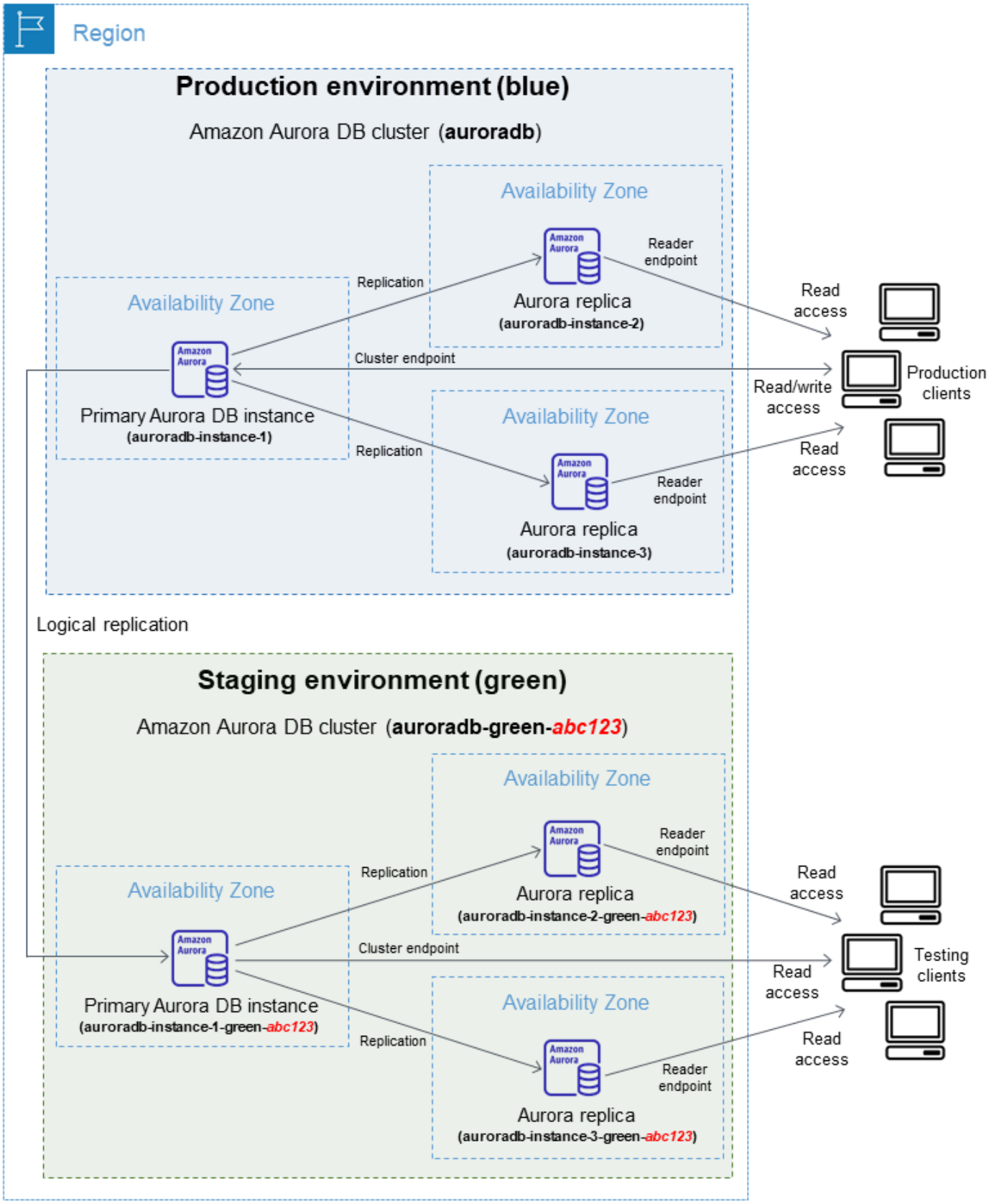
1. Identifiez un cluster de bases de données de production qui nécessite des mises à jour.

   L'image suivante montre un exemple de cluster de bases de données de production.



2. Créez le blue/green déploiement. Pour obtenir des instructions, veuillez consulter Création d'un blue/green déploiement dans).

   L'image suivante montre un exemple de blue/green déploiement de l'environnement de production à partir de l'étape 1. Lors de la création du blue/green déploiement, RDS copie la topologie et la configuration complètes du cluster de base de données Aurora pour créer un environnement écologique. Les noms du cluster de bases de données et des instances de base de données copiés sont complétés par `-green-`*`random-characters`*. L'environnement de mise en scène de l'image contient le cluster de base de données (auroradb-green-). *`abc123`* Il contient également les trois instances de base de données du cluster de base de données (auroradb-instance1-green-, auroradb-instance2-green- et auroradb-instance3-green-)*`abc123`*. *`abc123`* *`abc123`*


Region
Production environment (blue)
Amazon Aurora DB cluster (auroradb)
Availability Zone
Aurora replica
(auroradb-instance-2)
Reader endpoint
Replication
Availability Zone
Primary Aurora DB instance
(auroradb-instance-1)
Cluster endpoint
Read access
Read/write access
Production clients
Availability Zone
Aurora replica
(auroradb-instance-3)
Replication
Reader endpoint
Read access
Logical replication
Staging environment (green)
Amazon Aurora DB cluster (auroradb-green-abc123)
Availability Zone
Aurora replica
(auroradb-instance-2-green-abc123)
Reader endpoint
Replication
Availability Zone
Primary Aurora DB instance
(auroradb-instance-1-green-abc123)
Cluster endpoint
Read access
Testing clients
Availability Zone
Aurora replica
(auroradb-instance-3-green-abc123)
Replication
Reader endpoint
Read access

Lorsque vous créez le blue/green déploiement, vous pouvez spécifier une version supérieure du moteur de base de données et un groupe de paramètres de cluster de base de données différent pour le cluster de base de données dans l'environnement vert. Vous pouvez également spécifier un groupe de paramètres de base de données différent pour les instances de base de données dans le cluster de bases de données.

RDS configure également la réplication de l'instance de base de données principale dans l'environnement bleu vers l'instance de base de données principale dans l'environnement vert.

> **Important**
>
> Pour Aurora MySQL version 3, une fois le blue/green déploiement créé, le cluster de base de données dans l'environnement vert n'autorise pas les opérations d'écriture par défaut. Toutefois, cela ne s'applique pas aux utilisateurs disposant du privilège `CONNECTION_ADMIN`, y compris l'utilisateur principal Aurora. Les utilisateurs dotés de ce privilège peuvent remplacer le comportement `read_only`. Pour plus d'informations, consultez Modèle de privilège basé sur les rôles.

3. Apportez des modifications à l'environnement intermédiaire.

   Par exemple, vous pouvez modifier la classe d'instance de base de données utilisée par une ou plusieurs instances de base de données dans l'environnement vert.

   Pour plus d'informations sur la modification d'un cluster de bases de données, consultez Modification d'un cluster de bases de données Amazon Aurora.

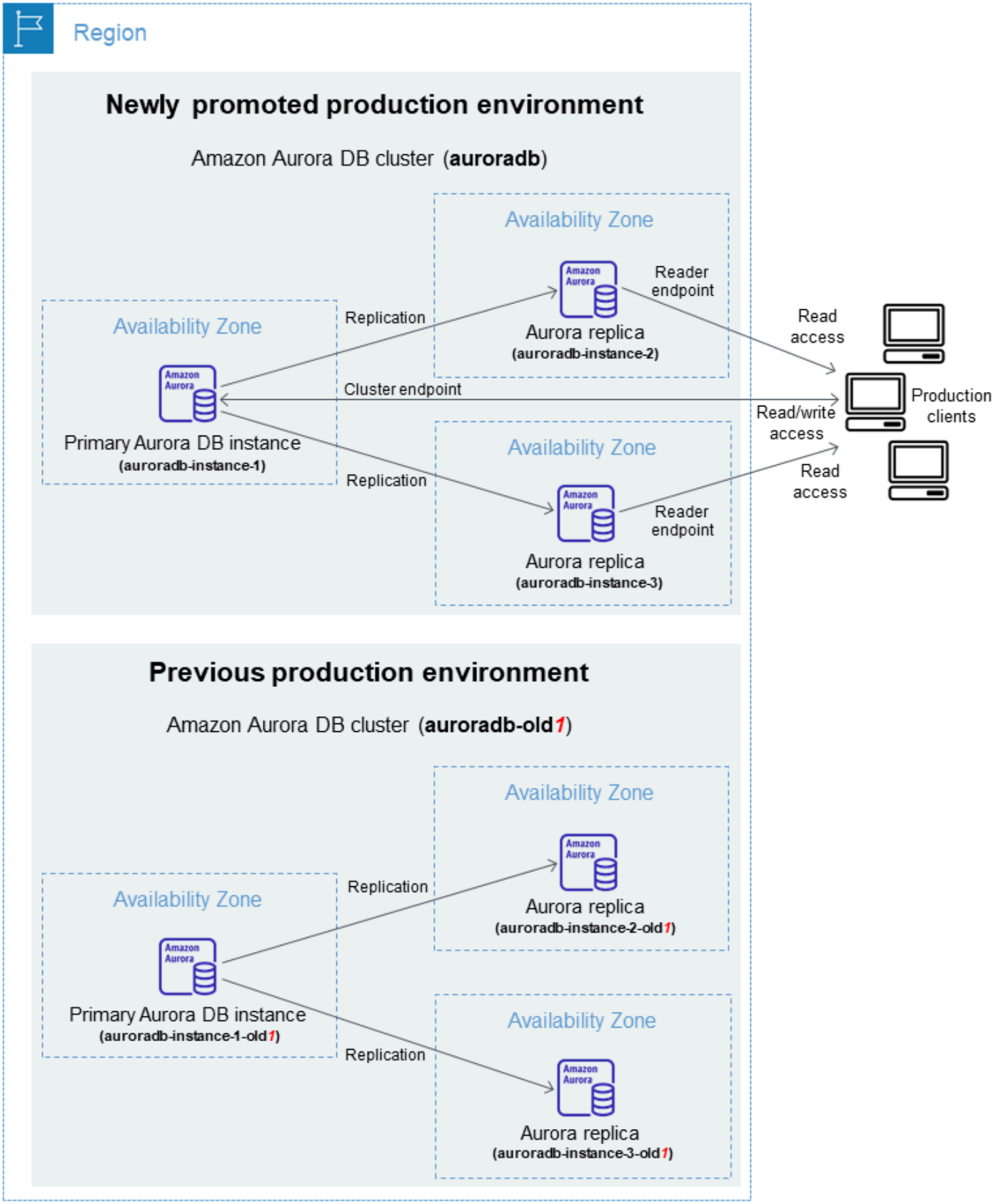
4. Testez votre environnement intermédiaire.

   Pendant les tests, nous vous recommandons de garder vos bases de données dans l'environnement vert en lecture seule. Activez les opérations d'écriture sur l'environnement vert avec prudence, car elles peuvent entraîner des conflits de réplication. Elles peuvent également entraîner la présence de données involontaires dans les bases de données de production après la bascule. Pour activer les opérations d'écriture pour Aurora MySQL, définissez le paramètre `read_only` sur `0`, puis redémarrez l'instance de base de données. Pour Aurora PostgreSQL, définissez le paramètre `default_transaction_read_only` sur `off` au niveau de la session.

5. Une fois prêt, basculez pour faire de l'environnement intermédiaire le nouvel environnement de production. Pour obtenir des instructions, consultez Changer de blue/green déploiement dans).

La bascule entraîne une durée d’indisponibilité. La durée d’indisponibilité est généralement inférieure à une minute, mais elle peut être plus longue en fonction de votre charge de travail.

L’image suivante présente les clusters de bases de données après la bascule.


Region
Newly promoted production environment
Amazon Aurora DB cluster (auroradb)
Availability Zone
Amazon Aurora
Reader endpoint
Availability Zone
Replication
Aurora replica
(auroradb-instance-2)
Read access
Cluster endpoint
Production clients
Read/write access
Primary Aurora DB instance
(auroradb-instance-1)
Availability Zone
Replication
Read access
Reader endpoint
Aurora replica
(auroradb-instance-3)
Previous production environment
Amazon Aurora DB cluster (auroradb-old1)
Availability Zone
Availability Zone
Replication
Aurora replica
(auroradb-instance-2-old1)
Primary Aurora DB instance
(auroradb-instance-1-old1)
Availability Zone
Replication
Aurora replica
(auroradb-instance-3-old1)

Après la bascule, le cluster de bases de données Aurora dans l'environnement vert devient le nouveau cluster de bases de données de production. Les noms et les points de terminaison de l'environnement de production actuel sont affectés à l'environnement de production qui vient d'être basculé. Aucune modification de votre application n'est requise. En conséquence, votre trafic de production s'écoule désormais vers le nouvel environnement de production. Le cluster de bases de données et les instances de base de données dans l'environnement bleu sont renommés en ajoutant `-old`*n* au nom actuel, où *n* est un numéro. Par exemple, supposons que le nom de l'instance de base de données dans l'environnement bleu est `auroradb-instance-1`. Après la bascule, le nom de l'instance de base de données pourrait être `auroradb-instance-1-old1`.

Dans l'exemple de l'image, les changements suivants se produisent pendant la bascule :

- Le cluster de bases de données `auroradb-green-abc123` de l'environnement vert devient le cluster de bases de données de production nommé `auroradb`.
- L'instance de base de données de l'environnement vert nommée `auroradb-instance1-green-abc123` devient l'instance de base de données de production `auroradb-instance1`.
- L'instance de base de données de l'environnement vert nommée `auroradb-instance2-green-abc123` devient l'instance de base de données de production `auroradb-instance2`.
- L'instance de base de données de l'environnement vert nommée `auroradb-instance3-green-abc123` devient l'instance de base de données de production `auroradb-instance3`.
- Le cluster de bases de données de l'environnement bleu nommé `auroradb` devient `auroradb-old1`.
- L'instance de base de données de l'environnement bleu nommée `auroradb-instance1` devient `auroradb-instance1-old1`.
- L'instance de base de données de l'environnement bleu nommée `auroradb-instance2` devient `auroradb-instance2-old1`.
- L'instance de base de données de l'environnement bleu nommée `auroradb-instance3` devient `auroradb-instance3-old1`.

6. Si vous n'avez plus besoin d'un blue/green déploiement, vous pouvez le supprimer. Pour obtenir des instructions, veuillez consulter Suppression d'un déploiement bleu/vert dans Amazon Aurora.

   Après la bascule, l'environnement de production précédent n'est pas supprimé afin que vous puissiez l'utiliser pour les tests de régression, si nécessaire.

Autoriser l'accès aux opérations de déploiement (Amazon Aurora blue/green)

Les utilisateurs doivent disposer des autorisations requises pour effectuer les opérations liées aux déploiements bleu/vert. Vous pouvez créer des politiques IAM qui accordent aux utilisateurs et aux rôles l'autorisation d'effectuer des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. Vous pouvez ensuite attacher ces politiques aux jeux d'autorisations ou rôles IAM qui requièrent ces autorisations. Pour de plus amples informations, veuillez consulter Identity and Access Management pour Amazon Aurora.

L'utilisateur qui crée un blue/green déploiement doit être autorisé à effectuer les opérations RDS suivantes :

- `rds:CreateBlueGreenDeployment`
- `rds:AddTagsToResource`
- `rds:CreateDBCluster`
- `rds:CreateDBInstance`
- `rds:CreateDBClusterEndpoint`

L'utilisateur qui passe d'un blue/green déploiement à un autre doit être autorisé à effectuer les opérations RDS suivantes :

- `rds:SwitchoverBlueGreenDeployment`
- `rds:ModifyDBCluster`
- `rds:PromoteReadReplicaDBCluster`

L'utilisateur qui supprime un blue/green déploiement doit être autorisé à effectuer les opérations RDS suivantes :

- `rds:DeleteBlueGreenDeployment`
- `rds:DeleteDBCluster`
- `rds:DeleteDBInstance`
- `rds:DeleteDBClusterEndpoint`

Aurora met en service et modifie les ressources dans l'environnement intermédiaire en votre nom. Ces ressources incluent des instances de base de données qui utilisent une convention de dénomination définie en interne. Par conséquent, les politiques IAM qui y sont associées ne peuvent pas contenir de modèles de noms de ressources partiels tels que `my-db-prefix-*`. Seuls les caractères génériques (*) sont pris en charge. En général, nous recommandons d'utiliser des balises de ressources et d'autres attributs pris en charge pour contrôler l'accès à ces ressources, plutôt que des caractères génériques. Pour plus d'informations, consultez Actions, ressources et clés de condition pour Amazon RDS.

Autorisations supplémentaires pour les Blue/Green déploiements de bases de données mondiales Aurora

Lors de la création de blue/green déploiements pour les clusters de base de données globale Aurora, outre les autorisations répertoriées ci-dessus, les utilisateurs ont besoin des autorisations suivantes pour effectuer des opérations de gestion de la topologie du cluster global.

L'utilisateur qui crée un blue/green déploiement doit être autorisé à effectuer les opérations RDS suivantes :

- `rds:CreateGlobalCluster`

L'utilisateur qui passe d'un blue/green déploiement à un autre doit être autorisé à effectuer les opérations RDS suivantes :

- `rds:ModifyGlobalCluster`
- `rds:PromoteReadReplicaDBCluster`

L'utilisateur qui supprime un blue/green déploiement doit être autorisé à effectuer les opérations RDS suivantes :

- `rds:DeleteGlobalCluster`

Limites et considérations relatives aux (Amazon Aurora blue/green)

Les déploiements bleu/vert dans Amazon RDS nécessitent une prise en compte attentive de facteurs tels que les emplacements de réplication, la gestion des ressources, le dimensionnement des instances et les impacts potentiels sur les performances de la base de données. Les sections

suivantes fournissent des conseils pour vous aider à optimiser votre stratégie de déploiement afin de garantir une durée d'indisponibilité minimale, des transitions fluides et une gestion efficace de votre environnement de base de données.

Rubriques

Limitations relatives aux blue/green déploiements

Les limitations suivantes s'appliquent aux blue/green déploiements.

Rubriques

Limitations générales pour les déploiements bleu/vert

Les limitations générales suivantes s'appliquent aux blue/green déploiements :

- Vous ne pouvez pas arrêter et démarrer un cluster faisant partie d'un blue/green déploiement.
- Les déploiements bleu/vert ne prennent pas en charge la gestion des mots de passe des utilisateurs principaux avec AWS Secrets Manager.
- Si vous tentez de forcer le cluster de base de données bleu à revenir en arrière, le blue/green déploiement est interrompu et le basculement est bloqué.
- Lors de la bascule, les environnements bleu et vert ne peuvent pas avoir d'intégrations zéro ETL avec Amazon Redshift. Vous devez d'abord supprimer l'intégration et basculer, puis recréer l'intégration.
- Le planificateur d'événements (`event_scheduler`paramètre) doit être désactivé dans l'environnement vert lorsque vous créez un blue/green déploiement. Cela évite que des événements soient générés dans l'environnement vert et provoquent des incohérences.

- Les politiques d’autoscaling configurées sur le cluster de bases de données bleu ne sont pas copiées dans l’environnement vert. Vous devez les reconfigurer après la bascule, qu’elles aient été initialement configurées dans l’environnement bleu ou vert.
- Vous ne pouvez pas transformer un cluster de bases de données non chiffré en un cluster de bases de données chiffré. En outre, vous ne pouvez pas transformer un cluster de bases de données chiffré en cluster de bases de données non chiffré.
- Vous ne pouvez pas transmettre un cluster de bases de données bleu vers une version de moteur supérieure à celle de son cluster de bases de données vert correspondant.
- Les ressources de l’environnement bleu et de l’environnement vert doivent se trouver dans le même Compte AWS.
- Les déploiements bleu/vert ne sont pas pris en charge pour les fonctionnalités suivantes :
  - Proxy Amazon RDS
  - Réplicas en lecture entre Régions
  - Clusters DB Aurora Serverless v1
  - CloudFormation

Limitations d'Aurora MySQL les déploiements blue/green

Les limitations suivantes s'appliquent aux blue/green déploiements Aurora MySQL :

- Le cluster de bases de données source ne peut contenir aucune base de données nommée `tmp`. Les bases de données portant ce nom ne seront pas copiées dans l’environnement vert.
- Le cluster de bases de données bleu ne peut pas être un réplica externe du journal binaire.
- Si le retour sur trace est activé pour le cluster de bases de données source, le cluster de bases de données vert est créé sans prise en charge du retour sur trace. Cela est dû au fait que le retour en arrière ne fonctionne pas avec la réplication des journaux binaires (binlog), qui est requise pour les blue/green déploiements. Pour de plus amples informations, veuillez consulter the section called “Retour en arrière d’un cluster de bases de données”.
- Les déploiements bleu/vert ne prennent pas en charge le pilote AWS JDBC pour MySQL. Pour plus d'informations, consultez la section Limitations connues sur GitHub.

Les limitations suivantes s’appliquent aux déploiements bleu/vert Aurora PostgreSQL .

- Les tables non journalisées ne sont pas répliquées dans l'environnement vert, sauf si le paramètre `rds.logically_replicate_unlogged_tables` est défini sur 1 dans le cluster de bases de données bleues. Ne modifiez pas cette valeur de paramètre après avoir créé un blue/green déploiement afin d'éviter d'éventuelles erreurs de réplication sur les tables non enregistrées.
- Le cluster de bases de données bleues ne peut pas être une source logique (diffuseur de publication) ou un réplica (abonné).
- Si le cluster de bases de données bleu est configuré en tant que serveur externe d'une extension de l'encapsuleur de données externes (FDW), vous devez utiliser le nom du point de terminaison du cluster au lieu des adresses IP. Ainsi, la configuration reste fonctionnelle après la bascule.
- Lors d'un blue/green déploiement, chaque base de données nécessite un emplacement de réplication logique. À mesure que le nombre de bases de données augmente, la surcharge en ressources augmente et peut potentiellement entraîner un retard de réplication, en particulier si la mise à l'échelle du cluster de bases de données est insuffisante. L'impact dépend de facteurs tels que la charge de travail de la base de données et le nombre de connexions. Pour atténuer ce problème, pensez à augmenter verticalement votre classe d'instance de base de données ou à réduire le nombre de bases de données sur l'instance source.
- Les déploiements bleu/vert sont pris en charge pour Babelfish pour Aurora PostgreSQL uniquement pour la version 15.7 et versions 15 ultérieures, et la version 16.3 et versions 16 ultérieures.
- Si vous souhaitez capturer des plans d'exécution dans des réplicas Aurora, vous devez fournir le point de terminaison du cluster de bases de données bleu lorsque vous appelez la fonction `apg_plan_mgmt.create_replica_plan_capture`. Les captures des plans peuvent ainsi continuer de fonctionner après la bascule. Pour plus d'informations, consultez the section called "Capture de plans d'exécution Aurora PostgreSQL dans des réplicas".
- Le processus d'application de la réplication logique dans un environnement vert est à thread unique. Si l'environnement bleu génère un volume élevé de trafic d'écriture, l'environnement vert risque de ne pas être en mesure de suivre le rythme. Cela peut entraîner un retard ou un échec de réplication, en particulier pour les charges de travail qui produisent un débit d'écriture élevé en continu. Assurez-vous de tester minutieusement vos charges de travail. Pour les scénarios nécessitant des mises à niveau des versions majeures et la gestion de charges de travail d'écriture de gros volumes, envisagez d'autres approches telles que l'utilisation de AWS Database Migration Service (AWS DMS) ou la réplication logique autogérée.
- La création de partitions sur des tables partitionnées n'est pas prise en charge lors des déploiements bleu/vert pour Aurora. La création de partitions implique des opérations de langage de définition des données (DDL) comme `CREATE TABLE`, qui ne sont pas répliquées entre

l’environnement bleu et l’environnement vert. Cependant, les tables partitionnées existantes et leurs données sont répliquées dans l’environnement vert.

- Les limitations suivantes s’appliquent aux extensions PostgreSQL :
  - L'`pg_partman`extension doit être désactivée dans l'environnement bleu lorsque vous créez un blue/green déploiement. L’extension exécute des opérations DDL comme `CREATE TABLE`, qui interrompent la réplication logique de l’environnement bleu vers l’environnement vert.
  - L'`pg_cron`extension doit rester désactivée sur toutes les bases de données vertes après la création du blue/green déploiement. L’extension dispose d’exécutants en arrière-plan qui s’exécutent en tant que superutilisateur et contournent le paramètre de lecture seule de l’environnement vert, ce qui peut provoquer des conflits de réplication.
  - Le paramètre `apg_plan_mgmt.capture_plan_baselines` de l’extension `apg_plan_mgmt` doit être défini sur `off` dans toutes les bases de données vertes pour éviter les conflits de clés primaires si un plan identique est capturé dans l’environnement bleu. Pour de plus amples informations, veuillez consulter the section called “Présentation de la gestion des plans de requêtes d’Aurora PostgreSQL”.
  - Les `pgactive` extensions `pglogical` et doivent être désactivées dans l'environnement bleu lorsque vous créez un blue/green déploiement. Après avoir basculé l’environnement vert comme nouvel environnement de production, vous pouvez réactiver les extensions. En outre, la base de données bleue ne peut pas être un abonné logique d’une instance externe.
  - Si vous utilisez l’extension `pgAudit`, elle doit rester dans les bibliothèques partagées (`shared_preload_libraries`) sur les groupes de paramètres de base de données personnalisés pour les instances de base de données bleues et vertes. Pour de plus amples informations, veuillez consulter the section called “Configuration de l’extension pgAudit”.

Limitations spécifiques à la réplication logique pour les déploiements blue/green

Le tableau suivant décrit les limitations de réplication logique qui s’appliquent aux déploiements bleu/vert pour Aurora PostgreSQL. Pour plus d’informations, consultez Restrictions dans la documentation Réplication logique PostgreSQL.

Limitation	Explication
Les instructions DDL (Langage de définition	Si Aurora détecte une modification DDL dans l’environnement bleu, vos bases de données vertes entrent dans un état de réplication dégradée. Vous

Limitation	Explication
de données), comme `CREATE TABLE` et `CREATE SCHEMA`, ne sont pas répliquées de l'environnement bleu vers l'environnement vert.	devez supprimer le blue/green déploiement et toutes les bases de données vertes, puis les recréer.
Les instructions de langage de contrôle de données (DCL), comme `GRANT` et `REVOKE`, ne sont pas répliquées de l'environnement bleu vers l'environnement vert.	Si Aurora détecte une tentative d'exécution d'une instruction DCL dans l'environnement bleu, un message d'avertissement s'affiche. Aucune configuration ou API n'est disponible pour modifier ce comportement, car il s'agit d'une limitation du processus de blue/green déploiement.
Les opérations `NEXTVAL` sur les objets de séquence ne sont pas synchronisées entre l'environnement bleu et l'environnement vert.	Pendant la bascule, Aurora incrémente les valeurs de séquence dans l'environnement vert pour les faire correspondre à celles dans l'environnement bleu. Si vous avez des milliers de séquences, cela peut retarder la bascule.

Limitation	Explication
Les objets volumineux dans l'environnement bleu ne sont pas répliqués dans l'environnement vert. Cela inclut à la fois les grands objets existants et tous les grands objets nouvellement créés ou modifiés au cours du processus de blue/green déploiement.	Si Aurora détecte dans l'environnement bleu la création ou la modification d'objets volumineux qui sont stockés dans la table système `pg_largeobject` , vos bases de données vertes entrent dans un état de réplication dégradée. Vous devez supprimer le blue/green déploiement et toutes les bases de données vertes, puis les recréer.
L'actualisation des vues matérialisées interrompt la réplication.	L'actualisation des vues matérialisées dans l'environnement bleu interrompt la réplication dans l'environnement vert. Évitez d'actualiser les vues matérialisées dans l'environnement bleu. Après la bascule, vous pouvez les actualiser manuellement à l'aide de la commande REFRESH MATERIALIZED VIEW ou planifier une actualisation.
Les opérations UPDATE et DELETE ne sont pas autorisées sur les tables dépourvues de clé primaire.	Avant de créer un blue/green déploiement, assurez-vous que toutes les tables disposent d'une clé primaire ou d'une utilisation`REPLICA IDENTITY FULL`. Toutefois, utilisez `REPLICA IDENTITY FULL` uniquement s'il n'existe aucune clé primaire ou unique, car cela affecte les performances de réplication. Pour plus d'informations, consultez la documentation sur PostgreSQL.

Limites de la base de données globale Aurora pour les blue/green déploiements

Outre les limitations générales et spécifiques au moteur indiquées ci-dessus, les limitations suivantes s'appliquent aux blue/green déploiements pour Aurora Global Database :

- Toutes les opérations doivent être lancées depuis la même région que le cluster de rédacteurs de la base de données globale.
- L'exécution d'un basculement global ou d'un basculement global entraînera l'invalidité du blue/green déploiement actif. Le déploiement bleu-vert doit être supprimé et recréé à partir de la nouvelle région principale.
- Pour Aurora PostgreSQL, si le transfert d'écriture global est activé dans votre environnement de production et que vous créez blue/green un déploiement, le transfert d'écriture est désactivé sur le cluster vert. Il n'est activé dans l'environnement vert qu'après le blue/green passage au numérique, lorsque l'environnement vert devient le nouvel environnement de production. Après le basculement, le transfert d'écriture est désactivé sur le `-old1` cluster.
- La modification de la topologie de la base de données globale après la création du blue/green déploiement entraînera l'invalidité du blue/green déploiement actif. Le déploiement bleu-vert devrait être supprimé et recréé à partir de la nouvelle région principale.
- Les instantanés automatisés sont conservés selon le nombre de jours de conservation des sauvegardes initialement configurés dans l'ancien environnement bleu. Les instantanés automatisés de l'ancien cluster bleu ne sont pas copiés en vert.
- Le basculement global est pris en charge lors d'un blue/green basculement, mais un basculement global n'est pas pris en charge lors d'un basculement. blue/green
- Assurez-vous que le cluster de base de données et les groupes de paramètres de base de données pour l'environnement écologique existent dans toutes les régions secondaires avec des noms identiques. Si le groupe de paramètres d'une région n'est pas disponible, le groupe de paramètres par défaut des régions est utilisé.
- Évitez d'utiliser le proxy RDS sur les membres de la base de données globale lors du changement blue/green de déploiement.

Considérations relatives aux blue/green déploiements

Amazon RDS suit les ressources lors des blue/green déploiements à l'aide de la `DbiResourceId` fin `DbClusterResourceId` de chaque ressource. Cet identifiant de ressource est un identifiant Région AWS unique et immuable pour la ressource.

L’identifiant de ressource est distinct de l’identifiant de cluster de bases de données : Chacun d’entre eux est répertorié dans la configuration de base de données de la console RDS.

Le nom (ID de cluster) d'une ressource change lorsque vous changez de blue/green déploiement, mais chaque ressource conserve le même ID de ressource. Par exemple, l’identifiant d’un cluster

de bases de données aurait pu être `mycluster` dans l'environnement bleu. Après la bascule, le même cluster de bases de données pourrait être renommé en `mycluster-old1`. Cependant, l'ID de ressource du cluster de bases de données ne change pas pendant la bascule. Ainsi, lorsque vous remplacez les ressources vertes par les nouvelles ressources de production, leur ressource IDs ne correspond pas à la ressource bleue IDs qui était auparavant en production.

Après avoir transféré un blue/green déploiement, envisagez de mettre à jour la ressource en IDs fonction de celles des ressources de production récemment transférées pour les fonctionnalités et services intégrés que vous avez utilisés avec les ressources de production. Plus précisément, envisagez les mises à jour suivantes :

- Si vous effectuez un filtrage à l'aide de l'API et de la ressource RDS IDs, ajustez la ressource IDs utilisée pour le filtrage après le passage au numérique.
- Si vous l'utilisez CloudTrail pour auditer des ressources, ajustez les consommateurs de CloudTrail afin de suivre la nouvelle ressource IDs après le passage au numérique. Pour de plus amples informations, veuillez consulter Surveillance des appels d'API Amazon Aurora dansAWS CloudTrail.
- Si vous utilisez des flux d'activité de base de données pour les ressources dans l'environnement bleu, ajustez votre application pour surveiller les événements de base de données pour le nouveau flux après la bascule. Pour de plus amples informations, veuillez consulter Régions et moteurs de base de données Aurora pris en charge pour les flux d'activité de base de données.
- Si vous utilisez l'API Performance Insights, ajustez la ressource IDs dans les appels à l'API après le passage au numérique. Pour de plus amples informations, veuillez consulter Surveillance de la charge de la base de données avec Performance Insights sur .

  Vous pouvez surveiller une base de données avec le même nom après la bascule, mais elle ne contient pas les données d'avant la bascule.
- Si vous utilisez des ressources IDs dans les politiques IAM, assurez-vous d'ajouter la ressource IDs des ressources récemment transférées lorsque cela est nécessaire. Pour de plus amples informations, veuillez consulter Identity and Access Management pour Amazon Aurora.
- Si des rôles IAM sont associés à votre cluster de bases de données, veillez à les réassocier après la bascule. Les rôles attachés ne sont pas automatiquement copiés dans l'environnement vert.
- Si vous vous authentifiez auprès de votre cluster de bases de données à l'aide de l'authentification de base de données IAM, veillez à ce que la politique IAM utilisée pour accéder à la base de données contienne à la fois les bases de données bleues et vertes répertoriées sous l'élément `Resource` de la politique. Cela est nécessaire pour se connecter à la base de données verte après

la bascule. Pour de plus amples informations, veuillez consulter the section called “Création et utilisation d'une politique IAM pour l'accès à une base de données IAM”.

- Si vous souhaitez restaurer un instantané de cluster de base de données manuel pour un cluster de base de données faisant partie d'un blue/green déploiement, assurez-vous de restaurer le cliché de cluster de base de données correct en examinant l'heure à laquelle le cliché a été pris. Pour de plus amples informations, veuillez consulter Restauration à partir d’un instantané de cluster de bases de données.
- Après le changement, AWS Database Migration Service (AWS DMS) les tâches de réplication ne peuvent pas reprendre car le point de contrôle de l'environnement bleu n'est pas valide dans l'environnement vert. Vous devez recréer la tâche DMS avec un nouveau point de contrôle pour poursuivre la réplication.
- Amazon Aurora crée l’environnement vert en clonant le volume de stockage Aurora sous-jacent dans l’environnement bleu. Le volume de cluster vert stocke uniquement les modifications incrémentielles apportées à l’environnement vert. Si vous supprimez le cluster de bases de données dans l’environnement bleu, la taille du volume de stockage Aurora sous-jacent dans l’environnement vert atteint sa taille complète. Pour plus d’informations, consultez the section called “Clonage d’un volume pour un cluster de bases de données Aurora”.
- Lorsque vous ajoutez une instance de base de données au cluster de bases de données dans l’environnement vert d’un déploiement bleu/vert, la nouvelle instance de base de données ne remplacera pas une instance de base de données dans l’environnement bleu lors du basculement. Cependant, la nouvelle instance de base de données est conservée dans le cluster de bases de données et devient une instance de base de données dans le nouvel environnement de production.
- Lorsque vous supprimez une instance de base de données dans le cluster de base de données dans l'environnement vert d'un blue/green deployment, you can't create a new DB instance to replace it in the blue/green déploiement.

  Si vous créez une nouvelle instance de base de données avec le même nom et le même ARN que l’instance de base de données supprimée, elle a une valeur `DbiResourceId` différente, de sorte qu’elle ne fait pas partie de l’environnement vert.

  Le comportement suivant survient si vous supprimez une instance de base de données dans le cluster de bases de données de l’environnement vert :

  - Si l’instance de base de données dans l’environnement bleu avec le même nom existe, elle ne sera pas basculée vers l’instance de base de données dans l’environnement vert. Cette instance de base de données ne sera pas renommée en ajoutant `-old`*n* au nom de l’instance de base de données.

  - Toute application qui pointe vers l’instance de base de données dans l’environnement bleu continue à utiliser la même instance de base de données après la bascule.
- Si vous utilisez des balises de ressources pour le contrôle d'accès ou la gestion opérationnelle, vous devez comprendre que les modifications des balises ne sont pas synchronisées entre les environnements bleu et vert avant le passage au numérique. Lorsque vous créez un blue/green déploiement, les balises de l'environnement bleu sont copiées dans l'environnement vert. Après la création, les modifications de balises que vous apportez à l'un ou l'autre environnement ne sont pas automatiquement synchronisées. Lors du passage au numérique, les balises d'environnement bleues remplacent toutes les balises de l'environnement vert. Appliquez toutes les balises nécessaires à l'environnement bleu avant de créer le blue/green déploiement, ou réappliquez les balises requises au nouvel environnement de production après le passage au numérique. Pour en savoir plus sur les identifications, consultez Marquage des ressources Amazon Aurora et Amazon RDS.

Bonnes pratiques pour les (Amazon Aurora blue/green)

Les meilleures pratiques pour les blue/green déploiements sont les suivantes.

Rubriques

- Bonnes pratiques générales pour les blue/green déploiements
- : bonnes pratiques pour les déploiements blue/green
- Meilleures pratiques d'Aurora PostgreSQL pour les déploiements blue/green
- Bonnes pratiques relatives à la base de données globale Aurora pour les déploiements blue/green

Bonnes pratiques générales pour les blue/green déploiements

Tenez compte des bonnes pratiques générales suivantes lorsque vous créez un déploiement bleu/vert.

- Testez minutieusement le cluster de bases de données Aurora dans l’environnement vert avant le basculement.
- Gardez vos bases de données dans l’environnement vert en lecture seule. Nous vous recommandons d’activer les opérations d’écriture sur l’environnement vert avec prudence, car elles peuvent entraîner des conflits de réplication. Elles peuvent également entraîner la présence de données involontaires dans les bases de données de production après la commutation.

- Si vous utilisez un blue/green déploiement pour implémenter des modifications de schéma, apportez uniquement des modifications compatibles avec la réplication.

  Par exemple, vous pouvez ajouter de nouvelles colonnes à la fin d'une table sans interrompre la réplication du déploiement bleu vers le déploiement vert. Toutefois, les modifications de schéma, telles que le renommage de colonnes ou de tables, interrompent la réplication vers le déploiement vert.

  Pour plus d'informations sur les modifications compatibles avec la réplication, consultez Replication with Differing Table Definitions on Source and Replica dans la documentation MySQL, et Restrictions dans la documentation Réplication logique PostgreSQL.
- Utilisez le point de terminaison du cluster, le point de terminaison de lecture ou le point de terminaison personnalisé pour toutes les connexions dans les deux environnements. N'utilisez pas de points de terminaison d'instance ou de points de terminaison personnalisés avec des listes statiques ou d'exclusion.
- Lorsque vous passez d'un blue/green déploiement à un autre, suivez les meilleures pratiques en la matière. Pour de plus amples informations, veuillez consulter the section called "Bonnes pratiques de bascule".

: bonnes pratiques pour les déploiements blue/green

Tenez compte des meilleures pratiques suivantes lorsque vous créez un blue/green déploiement à partir d'un cluster de base de données Aurora MySQL.

- Si l'environnement vert connaît un retard de réplica, tenez compte des éléments suivants :
  - Désactivez la conservation des journaux binaires si elle n'est pas nécessaire, ou désactivez-la temporairement jusqu'à ce que la réplication rattrape son retard. Pour ce faire, redéfinissez le paramètre du cluster de bases de données `binlog_format` sur `0` et redémarrez l'instance de base de données d'enregistreur verte.
  - Définissez temporairement le paramètre `innodb_flush_log_at_trx_commit` sur `0` dans le groupe de paramètres de base de données vert. Une fois que la réplication a rattrapé son retard, revenez à la valeur par défaut de 1 avant la bascule. Si un arrêt ou un incident inattendu survient avec la valeur du paramètre temporaire, reconstruisez l'environnement vert pour éviter toute corruption de données non détectée. Pour plus d'informations, consultez the section called "Configuration de la fréquence à laquelle le tampon du journal est vidé".

Meilleures pratiques d'Aurora PostgreSQL pour les déploiements blue/green

Tenez compte des meilleures pratiques suivantes lorsque vous créez un blue/green déploiement à partir d'un cluster de base de données Aurora PostgreSQL.

- Surveillez le cache d'écriture simultanée de la réplication logique Aurora PostgreSQL et modifiez la mémoire tampon du cache si nécessaire. Pour plus d'informations, consultez the section called "Surveillance du cache d'écriture simultanée de la réplication logique".
- Augmentez la valeur du paramètre de base de données `logical_decoding_work_mem` dans l'environnement bleu. Cela permet de réduire le nombre de décodages sur disque et d'utiliser de la mémoire à la place. Pour de plus amples informations, veuillez consulter the section called "Ajustement de la mémoire de travail pour le décodage logique".
  - Vous pouvez surveiller le dépassement des transactions écrites sur le disque à l'aide de la `ReplicationSlotDiskUsage` CloudWatch métrique. Cette métrique fournit des informations sur l'utilisation des emplacements de réplication sur le disque, ce qui permet d'identifier les cas où les données de transaction dépassent la capacité de mémoire et sont stockées sur disque. Vous pouvez surveiller la mémoire disponible à l'aide de la `FreeableMemory` CloudWatch métrique. Pour de plus amples informations, veuillez consulter the section called "Métriques de niveau instance pour Amazon Aurora".
  - Dans Aurora PostgreSQL versions 14 et ultérieures, vous pouvez surveiller la taille des fichiers de dépassement logique à l'aide de la vue système `pg_stat_replication_slots`.
- Mettez à jour toutes vos extensions PostgreSQL vers la dernière version avant de créer un déploiement. blue/green Pour de plus amples informations, veuillez consulter the section called "Mise à niveau des extensions PostgreSQL".
- Si vous utilisez l'extension `aws_s3`, autorisez le cluster de bases de données vert à accéder à Amazon S3 via un rôle IAM une fois l'environnement vert créé. Cela permet aux commandes d'importation et d'exportation de continuer à fonctionner après la bascule. Pour obtenir des instructions, consultez the section called "Configuration de l'accès à un compartiment Amazon S3".
- Si vous spécifiez une version du moteur supérieure pour l'environnement vert, exécutez l'opération `ANALYZE` sur toutes les bases de données pour actualiser la table `pg_statistic`. Les statistiques de l'optimiseur ne sont pas transférées lors d'une mise à niveau de version majeure. Vous devez donc régénérer toutes les statistiques pour éviter les problèmes de performances. Pour connaître les bonnes pratiques supplémentaires lors des mises à niveau des versions majeures, consultez the section called "Réalisation d'une mise à niveau de version majeure".
- Évitez de configurer les déclencheurs comme `ENABLE REPLICA` ou `ENABLE ALWAYS` si le déclencheur est utilisé sur la source pour manipuler des données. Dans le cas contraire, le

système de réplication propage les modifications et exécute le déclencheur, ce qui entraîne une duplication.

- Les transactions de longue durée peuvent entraîner un retard de réplica important. Pour réduire le retard de réplica, envisagez les opérations suivantes :
  - Réduisez les transactions de longue durée et les sous-transactions qui peuvent être retardées jusqu'à ce que l'environnement vert rattrape l'environnement bleu.
  - Réduisez les opérations en bloc dans l'environnement bleu jusqu'à ce que l'environnement vert rattrape l'environnement bleu.
  - Lancez une opération manuelle de congélation sous vide sur les tables occupées avant de créer le blue/green déploiement. Pour obtenir des instructions, consultez Réalisation d'un gel manuel du processus vacuum.
  - Dans PostgreSQL versions 12 et ultérieures, désactivez le paramètre `index_cleanup` sur les tables volumineuses ou occupées afin d'améliorer l'efficacité de la maintenance régulière des bases de données bleues. Pour plus d'informations, consultez Vidage d'une table le plus rapidement possible.

    > **Note**
    >
    > Le fait de ne pas nettoyer régulièrement l'index pendant l'opération de vacuum peut entraîner un gonflement de l'index, ce qui peut dégrader les performances d'analyse. Il est recommandé d'utiliser cette approche uniquement lors d'un blue/green déploiement. Une fois le déploiement terminé, nous vous recommandons de reprendre la maintenance et le nettoyage réguliers de l'index.

  - Un retard de réplica peut se produire si les instances de base de données bleues et vertes sont sous-dimensionnées par rapport à la charge de travail. Assurez-vous que vos instances de base de données n'atteignent pas leurs limites de ressources pour le type d'instance. Pour plus d'informations, consultez the section called "Analysez l'utilisation des ressources à l'aide de CloudWatch métriques".
- Une réplication lente peut entraîner des redémarrages fréquents des expéditeurs et des destinataires, ce qui retarde la synchronisation. Pour vous assurer qu'ils restent actifs, désactivez les délais d'expiration en réglant le paramètre `wal_sender_timeout` sur `0` dans l'environnement bleu et le paramètre `wal_receiver_timeout` sur `0` dans l'environnement vert.
- Vérifiez les performances de vos instructions UPDATE et DELETE et déterminez si la création d'un index sur la colonne utilisée dans la clause WHERE peut optimiser ces requêtes. Cela peut améliorer les performances lorsque les opérations sont réexécutées dans un environnement vert.

Pour plus d'informations, consultez the section called "Vérifiez les filtres de prédicat pour détecter les requêtes qui génèrent des attentes".

- Si vous utilisez des déclencheurs, assurez-vous qu'ils n'interfèrent pas avec la création, la mise à jour et la suppression d'objets `pg_catalog.pg_publication`, `pg_catalog.pg_subscription` et `pg_catalog.pg_replication_slots` dont le nom commence par « rds ».
- Si Babelfish est activé sur le cluster de bases de données source, les paramètres suivants doivent avoir les mêmes paramètres dans le groupe de paramètres du cluster de bases de données cible pour l'environnement vert que dans le groupe de paramètres du cluster de bases de données source :
  - rds.babelfish_status
  - babelfishpg_tds.tds_default_numeric_precision
  - babelfishpg_tds.tds_default_numeric_scale
  - babelfishpg_tsql.default_locale
  - babelfishpg_tsql.migration_mode
  - babelfishpg_tsql.server_collation_name

  Pour obtenir plus d'informations sur ces paramètres, consultez the section called "Paramètres du groupe de paramètres de cluster de bases de données pour Babelfish ".

Bonnes pratiques relatives à la base de données globale Aurora pour les déploiements blue/green

Outre les meilleures pratiques générales et spécifiques au moteur répertoriées ci-dessus, tenez compte des meilleures pratiques suivantes pour l' Global Database.

- Surveillez les CloudWatch indicateurs suivants pour identifier les périodes de faible activité dans votre environnement de production :
  - `DatabaseConnections`
  - `ActiveTransactions`

  Planifiez le blue/green passage au numérique pendant votre période de maintenance planifiée ou pendant une période de faible activité.
- Blue/Green switchover duration varies based on your workload and the number of secondary regions. When you initiate a blue/greenlors du basculement, le service attend que le délai de

réplication atteigne zéro avant de poursuivre. Nous vous recommandons de vérifier le délai de réplication avant de lancer un changement.

- Si vous avez l'intention d'utiliser un paramètre de base de données ou un groupe de paramètres de cluster de base de données autre que celui par défaut pour votre environnement écologique, créez le groupe de paramètres souhaité portant le même nom dans toutes les régions secondaires avant de lancer le blue/green déploiement.

Création d'un blue/green déploiement dans)

RDS copie la topologie et les fonctionnalités de l’environnement bleu dans une zone de transit. Si l’instance de base de données bleue comporte des réplicas en lecture, ils sont copiés en tant que réplicas en lecture de l’instance verte. Le stockage alloué à tous les réplicas verts correspond à l’instance principale verte, tandis que les autres paramètres de stockage sont hérités des réplicas bleus.

Lorsque vous créez un blue/green déploiement, vous spécifiez le cluster de base de données à copier dans le déploiement. Le cluster de bases de données que vous choisissez est le cluster de bases de données de production, et il devient le cluster de bases de données dans l’environnement bleu. RDS copie la topologie de l’environnement bleu dans une zone de transit, ainsi que ses fonctionnalités configurées. Le cluster de bases de données est copié dans l’environnement vert, et RDS configure la réplication du cluster de bases de données de l’environnement bleu vers le cluster de bases de données de l’environnement vert. RDS copie également toutes les instances de base de données dans le cluster de bases de données.

Rubriques

- Préparation d’un déploiement bleu/vert
- Spécification des modifications lors de la création d'un blue/green déploiement
- Création d’un déploiement bleu/vert
- Paramètres de création de déploiement bleu/vert

Préparation d’un déploiement bleu/vert

Vous devez suivre certaines étapes avant de créer un blue/green déploiement, en fonction du moteur sur lequel votre de données Aurora est exécutée.

Rubriques

- Préparation d'un cluster de base de données Aurora MySQL pour un blue/green déploiement
- Préparation d'un cluster de base de données Aurora PostgreSQL pour un déploiement blue/green
- Préparation d'un cluster de base de données Aurora Global Database pour un blue/green déploiement

Préparation d'un cluster de base de données Aurora MySQL pour un blue/green déploiement

Avant de créer un blue/green déploiement pour un cluster de base de données Aurora MySQL, le cluster doit être associé à un groupe de paramètres de cluster de base de données personnalisé avec la journalisation binaire (`binlog_format`) activée. La journalisation binaire est requise pour la réplication de l'environnement bleu vers l'environnement vert. Bien que n'importe quel format de journal binaire fonctionne, nous recommandons `ROW` pour réduire le risque d'incohérences de réplication. Pour plus d'informations sur la création d'un groupe de paramètres de cluster de bases de données personnalisé et la définition des paramètres, consultez the section called "Groupes de paramètres de cluster de bases de données".

> **Note**
>
> L'activation de la journalisation binaire augmente le nombre d' I/O opérations d'écriture sur le disque dans le cluster de base de données. Vous pouvez surveiller l'utilisation des IOPS à l'aide de cette `VolumeWriteIOPs` CloudWatch métrique.

Après avoir activé la journalisation binaire, assurez-vous de redémarrer le cluster de base de données afin que vos modifications prennent effet. Blue/green les déploiements nécessitent que l'instance du rédacteur soit synchronisée avec le groupe de paramètres du cluster de base de données, sinon la création échoue. Pour de plus amples informations, veuillez consulter Redémarrage d'une instance de base de données au sein d'un cluster Aurora.

En outre, nous recommandons de remplacer la période de conservation des journaux binaires par une valeur autre que `NULL` pour empêcher la purge des fichiers journaux binaires. Pour de plus amples informations, veuillez consulter the section called "Configuration et affichage de la configuration du journal binaire".

Préparation d'un cluster de base de données Aurora PostgreSQL pour un déploiement blue/green

Avant de créer un blue/green déploiement pour un cluster de base de données Aurora PostgreSQL, veillez à effectuer les opérations suivantes.

- Associez le cluster à un groupe de paramètres de cluster de bases de données personnalisé dont la réplication logique (`rds.logical_replication`) est activée. La réplication logique est requise pour la réplication de l'environnement bleu vers l'environnement vert.

  Lorsque vous activez la réplication logique, vous devez également ajuster certains paramètres du cluster, tels que `max_replication_slots`, `max_logical_replication_workers` et `max_worker_processes`. Pour obtenir des instructions sur l'activation de la réplication logique et le réglage de ces paramètres, consultez the section called “Configuration de la réplication logique”.

  Assurez-vous également que le paramètre `synchronous_commit` n'est pas défini sur on.

  Après avoir configuré les paramètres requis, redémarrez le cluster de base de données afin que vos modifications prennent effet. Blue/green les déploiements nécessitent que l'instance du rédacteur soit synchronisée avec le groupe de paramètres du cluster de base de données, sinon la création échoue. Pour de plus amples informations, veuillez consulter Redémarrage d'une instance de base de données au sein d'un cluster Aurora.
- Vérifiez que votre cluster de base de données exécute une version d'Aurora PostgreSQL compatible avec les déploiements. Blue/Green Pour obtenir une liste des versions compatibles, consultez the section called “Déploiements bleu/vert avec Aurora PostgreSQL”.
- Assurez-vous que toutes les tables du cluster de bases de données possèdent une clé primaire. La réplication logique PostgreSQL n'autorise pas les opérations UPDATE ou DELETE sur les tables dépourvues de clé primaire.

Préparation d'un cluster de base de données Aurora Global Database pour un blue/green déploiement

Avant de créer un blue/green déploiement pour votre cluster de base de données Aurora Global Database, tenez compte des points suivants :

- Toutes les opérations doivent être lancées depuis la même région que le cluster de rédacteurs de la base de données globale.

- Configuration du groupe de paramètres :
  - L'environnement vert utilise soit un nouveau groupe de paramètres que vous spécifiez, soit le même groupe de paramètres que le cluster bleu (par défaut).
  - Les groupes de paramètres personnalisés sont copiés dans l'environnement vert.
  - Si aucun groupe de paramètres spécifié n'existe dans la région secondaire, le groupe de paramètres par défaut de la région secondaire est utilisé pour l'environnement vert.

Spécification des modifications lors de la création d'un blue/green déploiement

Vous pouvez apporter les modifications suivantes au cluster d' de base de données dans l'environnement vert lorsque vous créez le blue/green déploiement.

Vous pouvez apporter d’autres modifications au cluster et à ses instances de base de données dans l’environnement vert après son déploiement. Par exemple, vous pouvez spécifier une version supérieure du moteur ou un autre groupe de paramètres.

Pour plus d’informations sur la modification d’un cluster de bases de données, consultez Modification d’un cluster de bases de données Amazon Aurora.

Rubriques

- Spécifier une version de moteur supérieure
- Spécifier un groupe de paramètres de base de données différent

Spécifier une version de moteur supérieure

Vous pouvez spécifier une version supérieure du moteur si vous voulez tester une mise à niveau du moteur de base de données. Lors de la bascule, la base de données est mise à niveau vers la version majeure ou mineure du moteur de base de données que vous spécifiez.

Spécifier un groupe de paramètres de base de données différent

Spécifiez un groupe de paramètres de cluster de bases de données différent de celui utilisé par le cluster de bases de données. Vous pouvez tester la manière dont les changements de paramètres affectent le cluster de bases de données dans l’environnement vert ou spécifier un groupe de paramètres pour une nouvelle version majeure du moteur de base de données dans le cas d’une mise à niveau.

Si vous spécifiez un groupe de paramètres de cluster de bases de données différent, le groupe de paramètres spécifié est associé au cluster de bases de données dans l'environnement vert. Si vous ne spécifiez aucun groupe de paramètres de cluster de bases de données différent, le cluster de bases de données dans l'environnement vert est associé au même groupe de paramètres que le cluster de bases de données bleu.

Création d'un déploiement bleu/vert

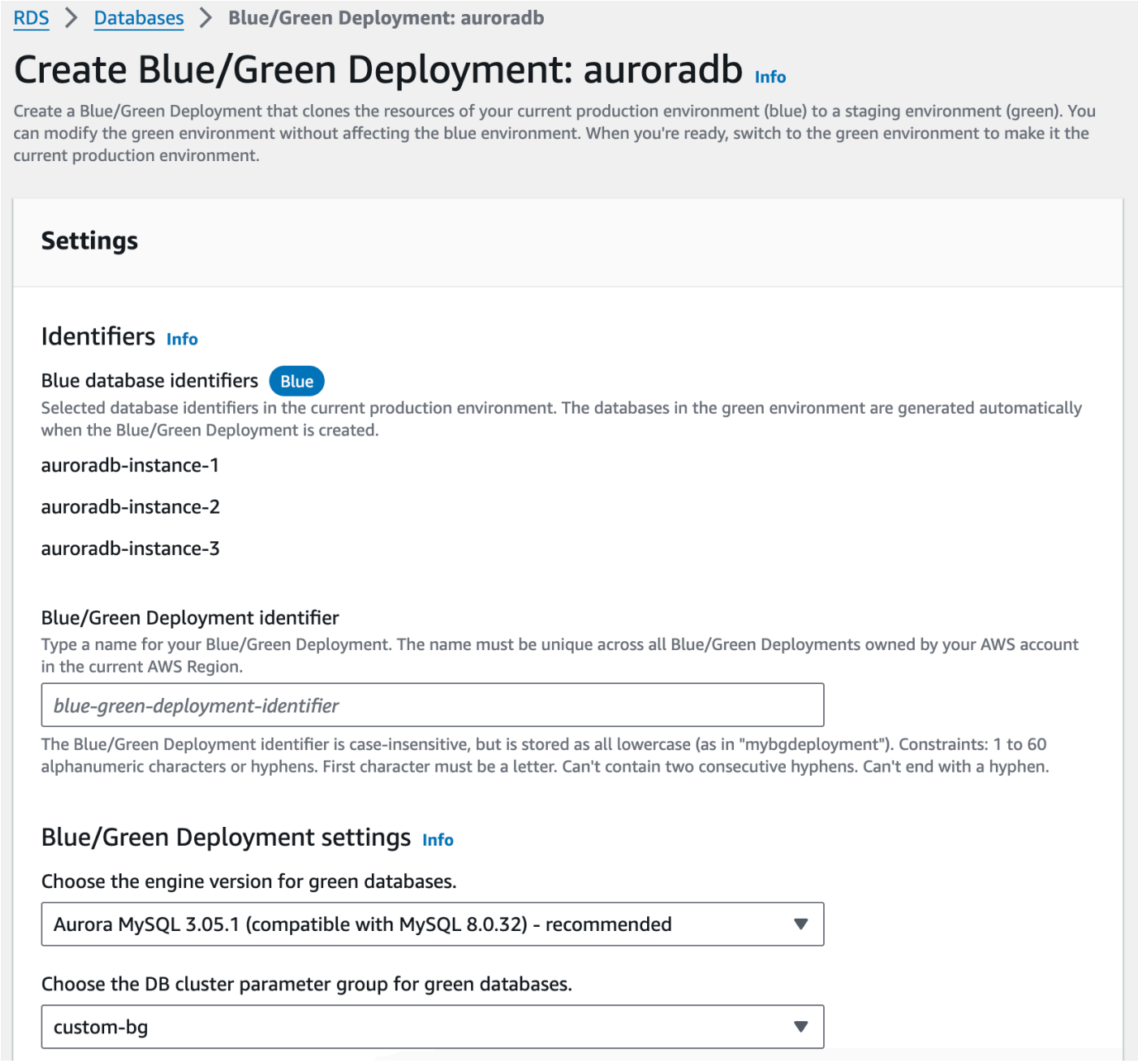
Vous pouvez créer un blue/green déploiement à l'aide de l'API AWS Management Console AWS CLI, de ou de l'API RDS.

Console

Pour créer un blue/green déploiement

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Bases de données, puis choisissez le cluster de bases de données que vous voulez copier dans un environnement vert.
3. Choisissez Actions, Créer un déploiement bleu/vert.

   La page Créer un blue/green déploiement apparaît.



4. Passez en revue les identifiants de base de données bleue. Assurez-vous qu'ils correspondent aux instances de base de données que vous attendez dans l'environnement bleu. Si ce n'est pas le cas, choisissez Annuler.
5. Pour le nom du déploiement bleu/vert, entrez un nom pour votre blue/green déploiement.
6. Dans les sections restantes, spécifiez les paramètres de l'environnement vert. Pour plus d'informations sur chaque paramètre, consultez the section called "Paramètres disponibles".

   Vous pouvez apporter d'autres modifications aux bases de données dans l'environnement vert après son déploiement.

7. Choisissez Créer.

AWS CLI

Pour créer un blue/green déploiement à l'aide de AWS CLI, utilisez la create-blue-green-deploymentcommande. Pour obtenir des informations sur toutes les options disponibles, consultez the section called “Paramètres disponibles”.

Example

Pour Linux, macOS ou Unix :

```
aws rds create-blue-green-deployment \
    --blue-green-deployment-name aurora-blue-green-deployment \
    --source arn:aws:rds:us-east-2:123456789012:cluster:auroradb \
    --target-engine-version 8.0 \
    --target-db-cluster-parameter-group-name mydbclusterparametergroup
```

Pour Windows :

```
aws rds create-blue-green-deployment ^
    --blue-green-deployment-name aurora-blue-green-deployment ^
    --source arn:aws:rds:us-east-2:123456789012:cluster:auroradb ^
    --target-engine-version 8.0 ^
    --target-db-cluster-parameter-group-name mydbclusterparametergroup
```

API RDS

Pour créer un blue/green déploiement à l'aide de l'API Amazon RDS, utilisez l'`CreateBlueGreenDeployment`opération. Pour plus d’informations sur chaque option, consultez the section called “Paramètres disponibles”.

Paramètres de création de déploiement bleu/vert

Le tableau suivant décrit les paramètres que vous pouvez choisir lorsque vous créez un blue/green déploiement. Pour plus d'informations sur les AWS CLI options, consultez create-blue-green-deployment. Pour plus d'informations sur les paramètres de l'API RDS, consultez CreateBlueGreenDeployment.

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
Identifiant de déploiement bleu/vert	Nom du blue/green déploiement.	Option de l'interface CLI : `--blue-green-deployment-name` Paramètre de l'API : `BlueGreenDeploymentName`
Identifiant de base de données bleue	Identifiant du cluster que vous souhaitez copier dans l'environnement vert. Lorsque vous utilisez l'interface de ligne de commande ou l'API, spécifiez l'Amazon Resource Name (ARN) du cluster.	Option de l'interface CLI : `--source` Paramètre de l'API : `Source`
Groupe de paramètres de cluster de bases de données pour les bases de données vertes	Groupe de paramètres à associer aux bases de données dans l'environnement vert.	Option de l'interface CLI : `--target-db-cluster-parameter-group-name` Paramètre de l'API : `TargetDBClusterParameterGroupName`
Version du moteur pour les bases de données vertes	Cette option met à niveau le cluster dans l'environnement vert vers la version de moteur de base de données spécifiée. Si vous choisissez un cluster de bases de données Aurora PostgreSQL, passez en revue et reconnaissez les limitations	Option de l'interface CLI : `--target-engine-version` Paramètre de l'API RDS : `TargetEngineVersion`

Paramètre de la console	Description du paramètre	Option de l'interface CLI et paramètre de l'API RDS
	de la réplication logique. Pour plus d'informations, consultez the section called "Limitations spécifiques à la réplication logique pour les déploiements blue/green ".	

Affichage d'un déploiement bleu/vert dans Amazon Aurora

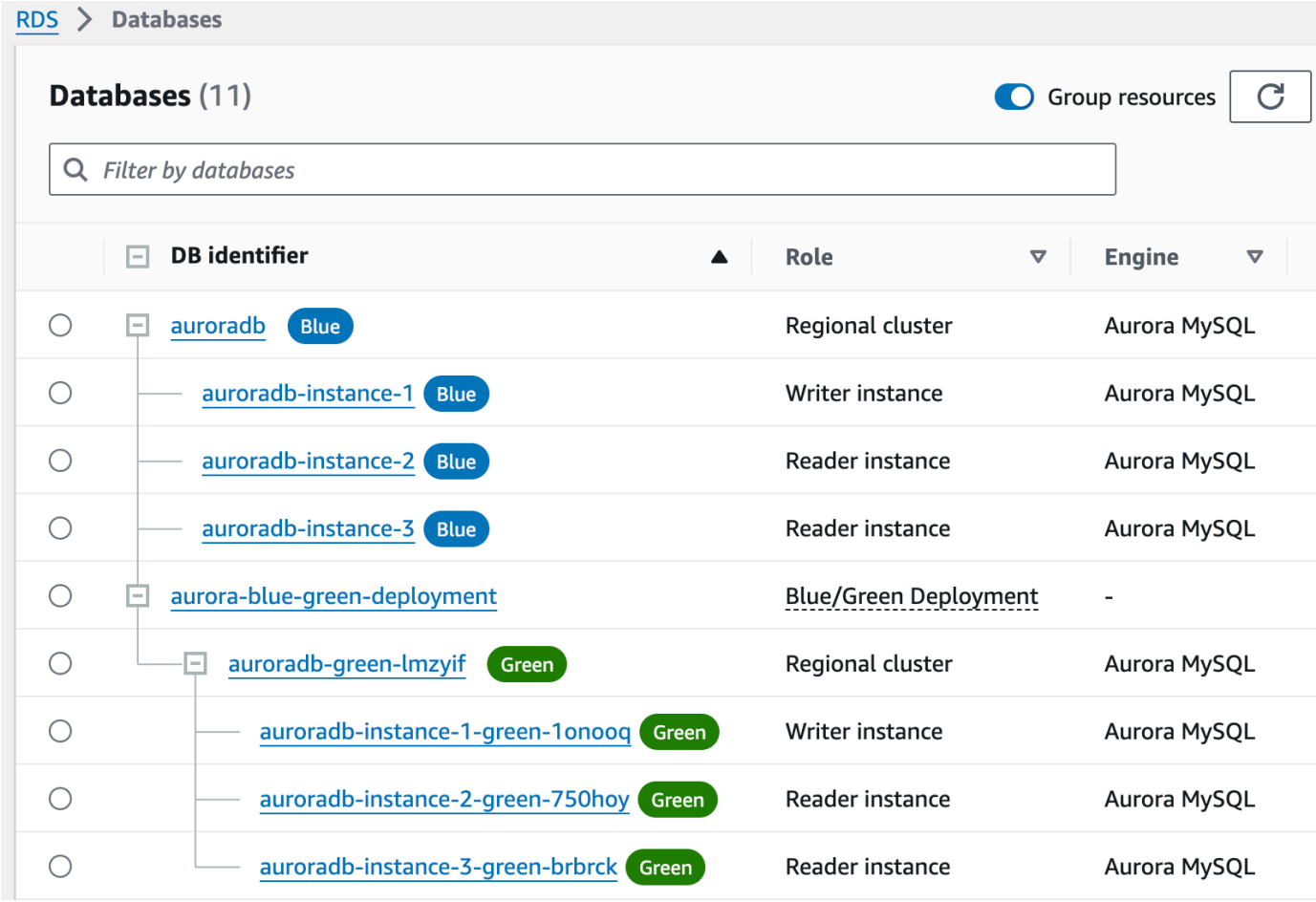
Vous pouvez afficher les détails d'un déploiement bleu/vert à l'aide de la AWS Management Console, d'AWS CLI ou de l'API RDS.

Vous pouvez également consulter des événements et vous y abonner pour obtenir des informations sur un déploiement bleu/vert. Pour plus d'informations, consultez Événements de déploiement bleu/vert.

Console

Pour afficher les détails d'un déploiement bleu/vert

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis recherchez le déploiement bleu/vert dans la liste.

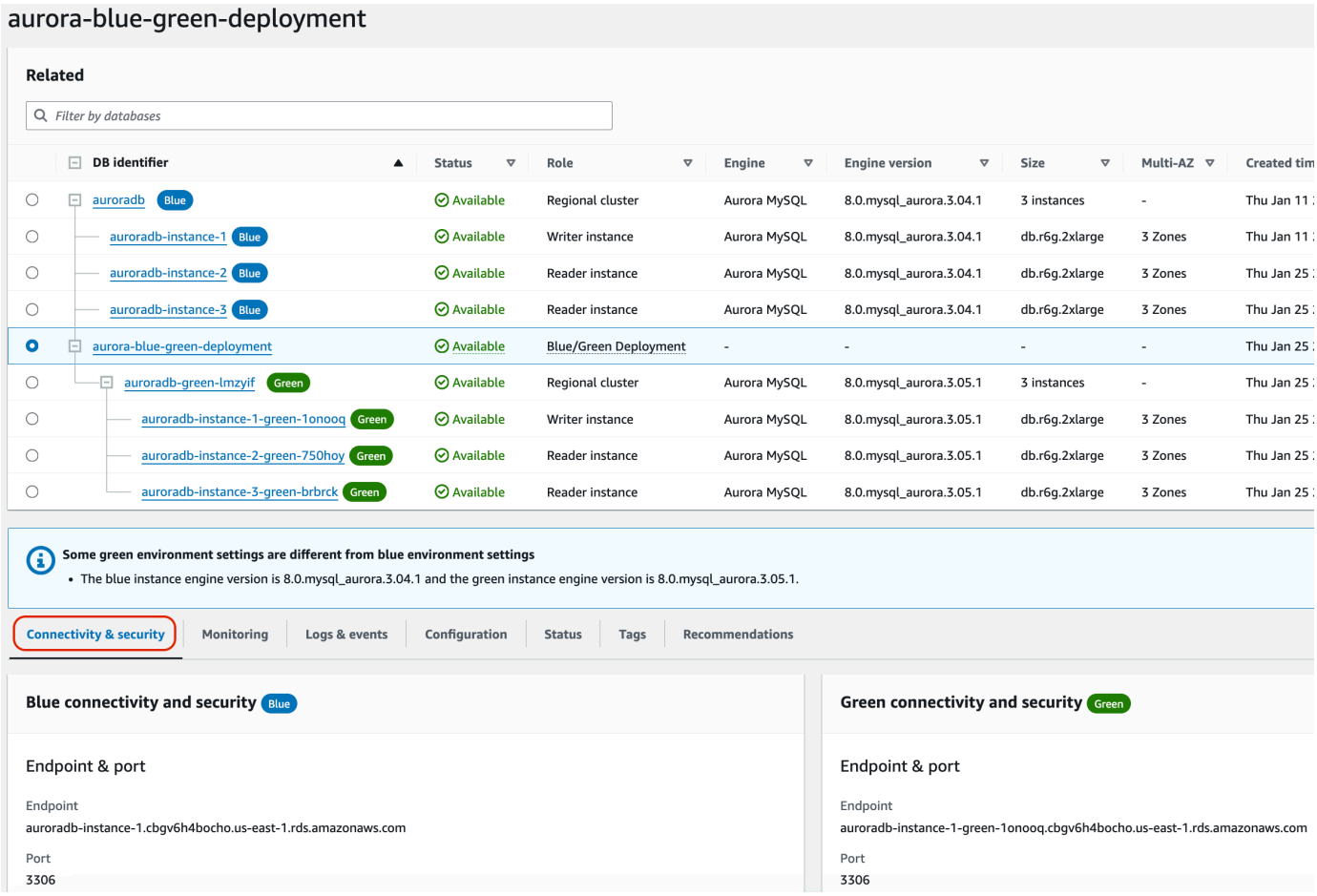


La valeur Rôle pour le déploiement bleu/vert est Déploiement bleu/vert.

3. Choisissez le nom du déploiement bleu/vert que vous souhaitez visualiser pour afficher ses détails.

   Chaque onglet comporte une section pour le déploiement bleu et une section pour le déploiement vert. Par exemple, dans l'onglet Configuration, la version du moteur de base de données peut être différente dans l'environnement bleu et dans l'environnement vert si vous mettez à niveau la version du moteur de base de données dans l'environnement vert.

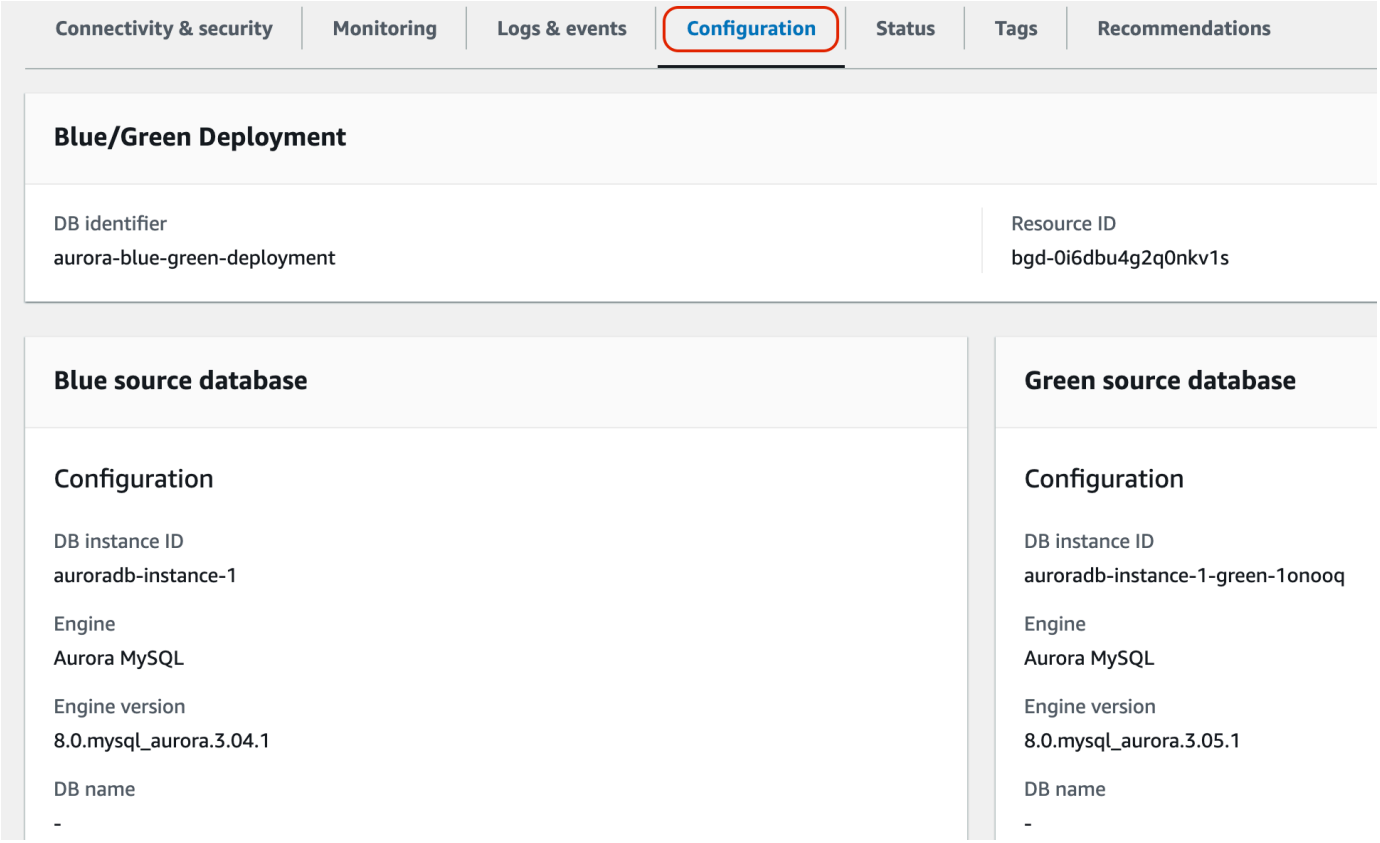
   L'image suivante montre un exemple de l'onglet Connectivité et sécurité :



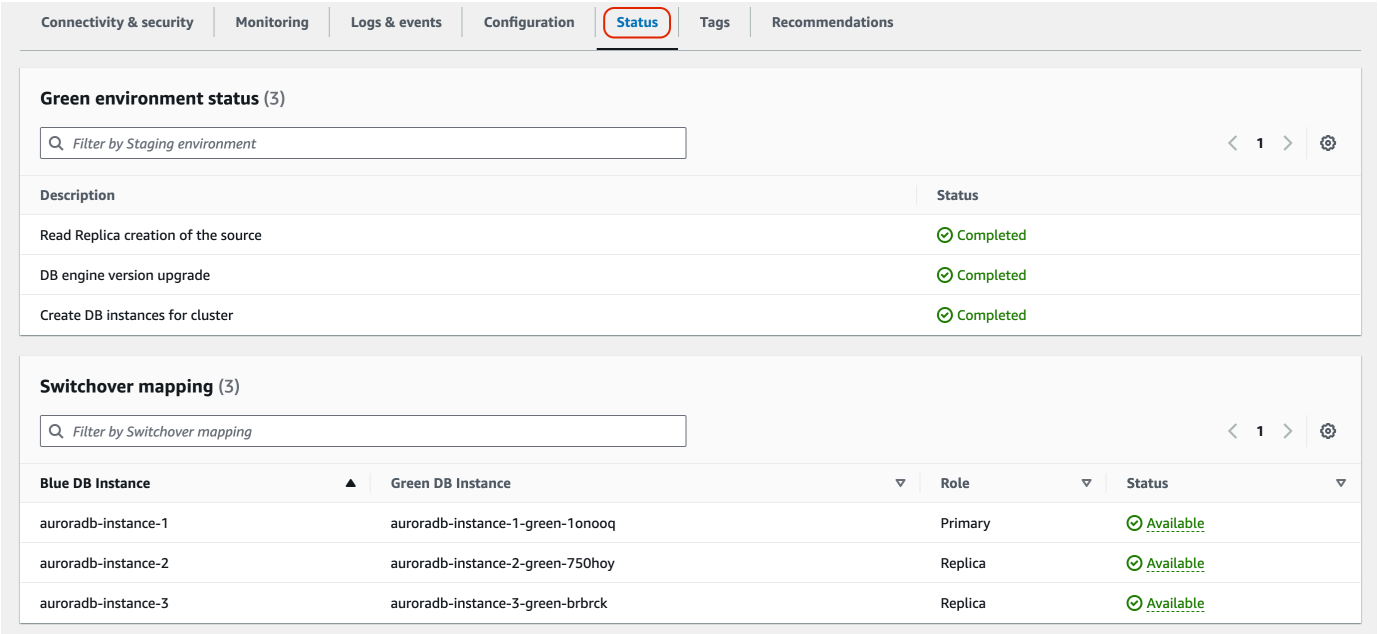
L'onglet Connectivité et sécurité comprend également une section intitulée Réplication, qui indique l'état actuel de la réplication et le retard de réplica entre les environnements bleu et vert. Si l'état de réplication est défini sur `Replicating`, le déploiement bleu/vert se réplique correctement.

Pour les déploiements bleu/vert Aurora PostgreSQL , l'état de réplication peut devenir `Replication degraded` si vous effectuez des modifications de DDL ou d'objets volumineux non prises en charge dans l'environnement bleu. Pour plus d'informations, consultez the section called "Limitations spécifiques à la réplication logique pour les déploiements blue/green ".

L'image suivante montre un exemple de l'onglet Configuration :



L'image suivante montre un exemple de l'onglet Statut :



AWS CLI

Pour afficher les détails d'un déploiement bleu/vert en utilisant AWS CLI, utilisez la commande describe-blue-green-deployments.

Example Affichage des détails d'un déploiement bleu/vert en filtrant sur son nom

Lorsque vous utilisez la commande describe-blue-green-deployments, vous pouvez filtrer sur `--blue-green-deployment-name`.

L'exemple suivant montre les détails d'un déploiement bleu/vert nommé *`my-blue-green-deployment`*.

```
aws rds describe-blue-green-deployments \
  --filters Name=blue-green-deployment-name,Values=my-blue-green-deployment
```

Example Affichage des détails d'un déploiement bleu/vert en spécifiant son identifiant

Lorsque vous utilisez la commande describe-blue-green-deployments, vous pouvez spécifier l'option `--blue-green-deployment-identifier`.

L'exemple suivant montre les détails d'un déploiement bleu/vert avec l'identifiant *`bgd-1234567890abcdef`*.

```
aws rds describe-blue-green-deployments \
  --blue-green-deployment-identifier bgd-1234567890abcdef
```

API RDS

Pour afficher les détails d'un déploiement bleu/vert à l'aide de l'API Amazon RDS, utilisez l'opération `DescribeBlueGreenDeployments` et spécifiez `BlueGreenDeploymentIdentifier`.

Changer de blue/green déploiement dans)

Une bascule transfère le cluster de bases de données, y compris ses instances de base de données, dans l'environnement vert pour qu'il devienne le cluster de bases de données de production. Avant l'opération de bascule, le trafic de production est acheminé vers le cluster dans l'environnement bleu. Après l'opération de bascule, le trafic de production est acheminé vers le cluster de bases de données dans l'environnement vert.

Le transfert d'un blue/green déploiement n'est pas la même chose que la promotion du cluster de base de données d' de base de données vert dans le cadre du blue/green déploiement. Si vous promouvez manuellement le cluster DB de base de données vert en choisissant Promouvoir dans le menu Actions, la réplication entre les environnements bleu et vert est interrompue et le blue/green déploiement passe à l'état de configuration non valide.

Rubriques

Délai de bascule

Vous pouvez spécifier un délai de bascule compris entre 30 secondes et 3 600 secondes (une heure). Si la bascule prend plus de temps que la durée spécifiée, toutes les modifications sont annulées et aucune modification n’est apportée à l’un ou l’autre des environnements. Le délai d’attente par défaut est de 300 secondes (cinq minutes).

Barrières de protection de bascule

Lorsque vous lancez une bascule, Amazon RDS effectue quelques vérifications de base pour tester la préparation des environnements bleu et vert à la bascule. Ces contrôles sont connus sous le nom de barrières de protection de bascule. Ces barrières de protection empêchent une bascule si les environnements ne sont pas prêts pour cela. Ils évitent donc une durée d’indisponibilité plus longue que prévu et empêchent la perte de données entre les environnements bleu et vert qui pourrait survenir si la bascule était lancée.

Amazon RDS exécute les contrôles de barrière de protection suivants sur l’environnement vert :

- État de la réplication : vérifiez si l’état de réplication du cluster de bases de données vert est sain. Le cluster de bases de données vert est un réplica du cluster de bases de données bleu.

- Décalage de réplication : vérifiez si le retard de réplica du cluster de base de données vert se situe dans les limites autorisées pour la bascule. Les limites autorisées sont basées sur le délai d'attente spécifié. Le retard de réplica indique dans quelle mesure le cluster de bases de données vert est en retard sur son cluster de bases de données bleu. Pour plus d'informations, consultez the section called "Surveillance du délai de réplication avant la bascule".
- Écritures actives : assurez-vous qu'aucune écriture n'est active sur le cluster de bases de données vert.

Amazon RDS exécute les contrôles de barrière de protection suivants sur l'environnement bleu :

- Réplication externe : pour Aurora PostgreSQL, assurez-vous que l'environnement bleu n'est pas une source logique autogérée (diffuseur de publication) ni un réplica (abonné). Si tel est le cas, nous vous recommandons de supprimer les emplacements de réplication autogérés et les abonnements dans toutes les bases de données de l'environnement bleu, de procéder à l'opération de bascule, puis de les recréer pour reprendre la réplication. Pour Aurora MySQL, vérifie si la base de données bleue n'est pas un réplica externe du journal binaire. Si tel est le cas, assurez-vous qu'il ne se réplique pas activement.
- Écritures actives de longue durée : assurez-vous qu'il n'y a pas d'écritures actives de longue durée sur le cluster de bases de données bleu, car elles peuvent augmenter le retard de réplica.
- Instructions DDL de longue durée : assurez-vous qu'aucune instruction DDL de longue durée ne figure sur le cluster de bases de données bleu, car elles peuvent augmenter le retard de réplica.
- Modifications PostgreSQL non prises en charge : pour les aucune modification du DDL ni aucun ajout ou modification d'objets volumineux n'ont été effectués dans l'environnement bleu. Pour de plus amples informations, veuillez consulter the section called "Limitations spécifiques à la réplication logique pour les déploiements blue/green ".

  Si Amazon RDS détecte des modifications PostgreSQL non prises en charge, il change l'état de réplication `Replication degraded` et vous indique que le basculement n'est pas disponible pour le déploiement. blue/green Pour procéder au basculement, nous vous recommandons de supprimer et de recréer le blue/green déploiement ainsi que toutes les bases de données vertes. Pour ce faire, choisissez Actions, Supprimer avec les bases de données vertes.

Actions de bascule

Lorsque vous passez d'un blue/green déploiement à un autre, RDS exécute les actions suivantes :

1. Exécute des contrôles de barrière de protection pour vérifier si les environnements bleu et vert sont prêts pour la bascule.
2. Arrête les nouvelles opérations d'écriture sur le cluster de bases de données dans les deux environnements.
3. Supprime les connexions aux instances de base de données dans les deux environnements et ne permet pas de nouvelles connexions.
4. Attend que la réplication rattrape son retard dans l'environnement vert afin que celui-ci soit synchronisé avec l'environnement bleu.
5. Renomme le cluster de bases de données et les instances de base de données dans les deux environnements.

   RDS renomme le cluster de bases de données et les instances de base de données dans l'environnement vert pour correspondre au cluster de bases de données et aux instances de base de données correspondants dans l'environnement bleu. Par exemple, supposons que le nom d'une instance de base de données dans l'environnement bleu est `mydb`. Supposons également que le nom de l'instance de base de données correspondante dans l'environnement vert est `mydb-green-abc123`. Pendant la bascule, le nom de l'instance de base de données dans l'environnement vert devient `mydb`.

   RDS renomme le cluster de bases de données et les instances de base de données dans l'environnement bleu en ajoutant `-old`*n* au nom actuel, où *n* est un nombre. Par exemple, supposons que le nom d'une instance de base de données dans l'environnement bleu est `mydb`. Après la bascule, le nom de l'instance de base de données pourrait être `mydb-old1`.

   RDS renomme également les points de terminaison dans l'environnement vert pour qu'ils correspondent aux points de terminaison correspondants dans l'environnement bleu, de sorte que les changements d'application ne sont pas nécessaires.
6. Permet les connexions aux bases de données dans les deux environnements.
7. Autorise les opérations d'écriture sur l'instance de base de données principale dans le nouvel environnement de production.

   Après la bascule, le précédent cluster de bases de données de production autorise les opérations de lecture uniquement . Même si vous activez les écritures sur le cluster de base de données, celui-ci reste en lecture seule jusqu'à ce que vous supprimiez le blue/green déploiement.

Vous pouvez surveiller l'état d'un passage au numérique à l'aide d'Amazon. EventBridge Pour de plus amples informations, veuillez consulter the section called “Événements de déploiement bleu/vert”.

Lors du passage au numérique, les balises de l'environnement bleu remplacent toutes les balises associées aux ressources de l'environnement vert. Toutes les balises que vous avez ajoutées directement aux ressources écologiques sont remplacées au cours de ce processus. Pour en savoir plus sur les identifications, consultez Marquage des ressources Amazon Aurora et Amazon RDS.

Si la bascule commence et s’arrête avant la fin pour une raison quelconque, les modifications sont annulées et aucune modification n’est apportée à l’environnement.

Bonnes pratiques de bascule

Avant de basculer, nous vous recommandons vivement de respecter les bonnes pratiques en accomplissant les tâches suivantes :

- Testez minutieusement les ressources dans l’environnement vert. Assurez-vous qu’elles fonctionnent correctement et efficacement.
- Surveillez les CloudWatch statistiques Amazon pertinentes. Pour de plus amples informations, veuillez consulter the section called “Vérification des CloudWatch métriques avant le passage au numérique”.
- Déterminez le meilleur moment pour la bascule.

  Pendant la bascule, les écritures sont interrompues dans les bases de données des deux environnements. Identifiez un moment où le trafic est le plus faible dans votre environnement de production. Les transactions de longue durée, telles que les transactions actives DDLs, peuvent augmenter le temps de transition, ce qui se traduit par des temps d'arrêt plus longs pour vos charges de travail de production.

  S'il existe un grand nombre de connexions sur vos de données, votre cluster de bases de données et vos instances de base de données, pensez à les réduire manuellement au minimum nécessaire pour votre application avant de passer au blue/green déploiement. L'un des moyens d'y parvenir consiste à créer un script qui surveille l'état du blue/green déploiement et commence à nettoyer les connexions lorsqu'il détecte que le statut est passé à`SWITCHOVER_IN_PROGRESS`.
- Assurez-vous que le cluster de bases de données et les instances de base de données dans les deux environnements sont dans l’état `Available`.
- Assurez-vous que le cluster de bases de données dans l’environnement vert est dans un état sain et qu’il se réplique.

- Assurez-vous que les configurations de votre réseau et de votre client n'augmentent pas le cache DNS Time-To-Live (TTL) au-delà de cinq secondes, ce qui est la valeur par défaut pour les zones DNS Aurora .
  Sinon, les applications continueront à envoyer du trafic d'écriture vers l'environnement bleu après la bascule.
- Pour les déploiements Aurora blue/green PostgreSQL Les déploiements suit :
  - Passez en revue les limitations relatives à la réplication logique et prenez les mesures nécessaires avant la bascule. Pour plus d'informations, consultez the section called "Limitations spécifiques à la réplication logique pour les déploiements blue/green ".
  - Exécutez l'opération `ANALYZE` pour actualiser la table `pg_statistics`. Cela réduit le risque de problèmes de performances après la bascule.

> **Note**
>
> Lors d'une bascule, vous ne pouvez modifier aucun cluster de base de données inclus dans la bascule.

Vérification des CloudWatch métriques avant le passage au numérique

Avant de passer d'un blue/green déploiement à un autre, nous vous recommandons de vérifier la valeur s des métriques suivantes sur Amazon CloudWatch.

- `DatabaseConnections`— Utilisez cette métrique pour estimer le niveau d'activité lors du blue/green déploiement et assurez-vous que la valeur est à un niveau acceptable pour votre déploiement avant de passer au mode de transfert. Si l'analyse des performances est activée, `DBLoad` est une métrique plus précise.
- `ActiveTransactions` : si `innodb_monitor_enable` a pour valeur `all` dans le groupe de paramètres de base de données de l'une de vos instances de base de données, utilisez cette métrique pour déterminer si un nombre élevé de transactions actives sont susceptibles de bloquer la bascule.

Pour plus d'informations, consultez the section called "CloudWatch métriques pour Aurora".

Surveillance du délai de réplication avant la bascule

Avant de passer d'un blue/green déploiement à un autre, assurez-vous que le délai de réplication est proche de zéro afin de réduire les temps d'arrêt.

Aurora MySQL

Pour les déploiements MySQL , vérifiez `AuroraBinlogReplicaLag` CloudWatch la métrique dans l'environnement vert pour identifier le délai de réplication actuel. Pour de plus amples informations, veuillez consulter the section called “Diagnostic et résolution du retard entre réplicas en lecture”.

Aurora PostgreSQL

Pour les , vérifiez `OldestReplicationSlotLag` CloudWatch la métrique dans l'environnement bleu pour identifier le délai de réplication actuel. Pour de plus amples informations, veuillez consulter the section called “Métriques de niveau instance pour Amazon Aurora”.

Vous pouvez également exécuter la requête SQL suivante dans l’environnement bleu :

```
SELECT slot_name,
       confirmed_flush_lsn as flushed,
       pg_current_wal_lsn(),
       (pg_current_wal_lsn() - confirmed_flush_lsn) AS lsn_distance
FROM pg_catalog.pg_replication_slots
WHERE slot_type = 'logical';

slot_name        |    flushed    | pg_current_wal_lsn | lsn_distance
-----------------+---------------+--------------------+------------
logical_replica1 | 47D97/CF32980 | 47D97/CF3BAC8      | 37192
```

`confirmed_flush_lsn` représente le dernier numéro de séquence du journal (LSN) qui a été envoyé au réplica. `pg_current_wal_lsn` représente l’emplacement actuel de la base de données. Une `lsn_distance` de 0 signifie que le réplica a rattrapé son retard.

Passer d'un blue/green déploiement à un autre

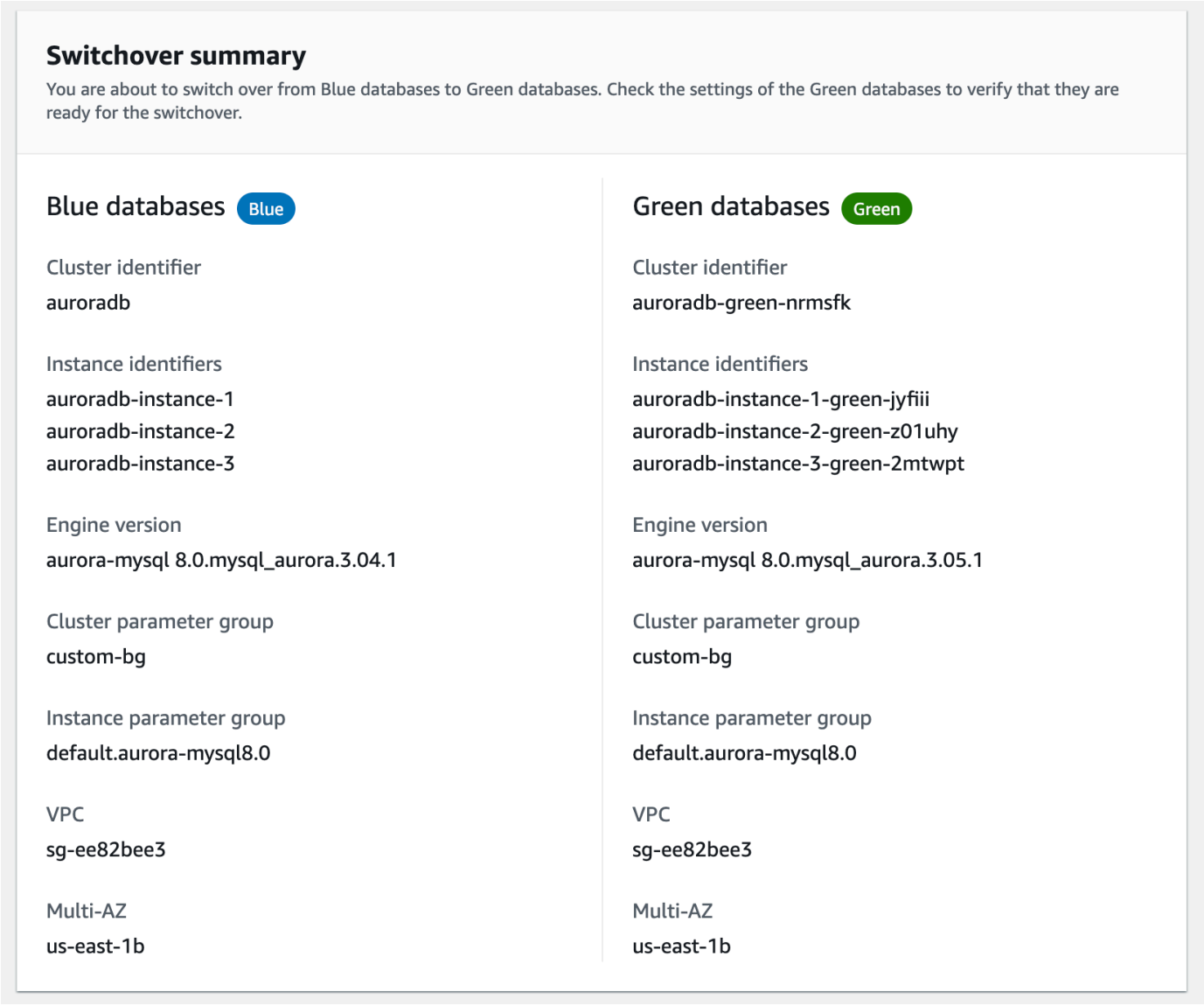
Vous pouvez passer d'un blue/green déploiement à l'aide de l'API AWS Management Console, de AWS CLI, ou de l'API RDS.

Console

Pour passer d'un blue/green déploiement à un autre

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le volet de navigation, choisissez Databases, puis choisissez le blue/green déploiement que vous souhaitez transférer.
3. Pour Actions, choisissez Basculer.

   La page Basculer apparaît.



4. Sur la page Basculer, consultez le résumé de la bascule. Assurez-vous que les ressources des deux environnements correspondent à ce que vous attendez. Si ce n'est pas le cas, choisissez Annuler.
5. Dans le champ Paramètre de délai d'attente, entrez le délai limite pour la bascule.
6. Si votre cluster exécute Aurora PostgreSQL, passez en revue et confirmez les recommandations avant la bascule. Pour plus d'informations, consultez the section called "Limitations spécifiques à la réplication logique pour les déploiements blue/green ".
7. Choisissez Basculer.

AWS CLI

Pour passer d'un blue/green déploiement à l'autre à l'aide de AWS CLI, utilisez la switchover-blue-green-deploymentcommande avec les options suivantes :

- `--blue-green-deployment-identifier`— Spécifiez l'ID de ressource du blue/green déploiement.
- `--switchover-timeout` : spécifiez la limite de temps pour la bascule, en secondes. La valeur par défaut est 300.

Example Passer d'un blue/green déploiement à un autre

Pour Linux, macOS ou Unix :

```
aws rds switchover-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --switchover-timeout 600
```

Pour Windows :

```
aws rds switchover-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --switchover-timeout 600
```

API RDS

Pour passer d'un blue/green déploiement à l'autre à l'aide de l'API Amazon RDS, utilisez l'`SwitchoverBlueGreenDeployment`opération avec les paramètres suivants :

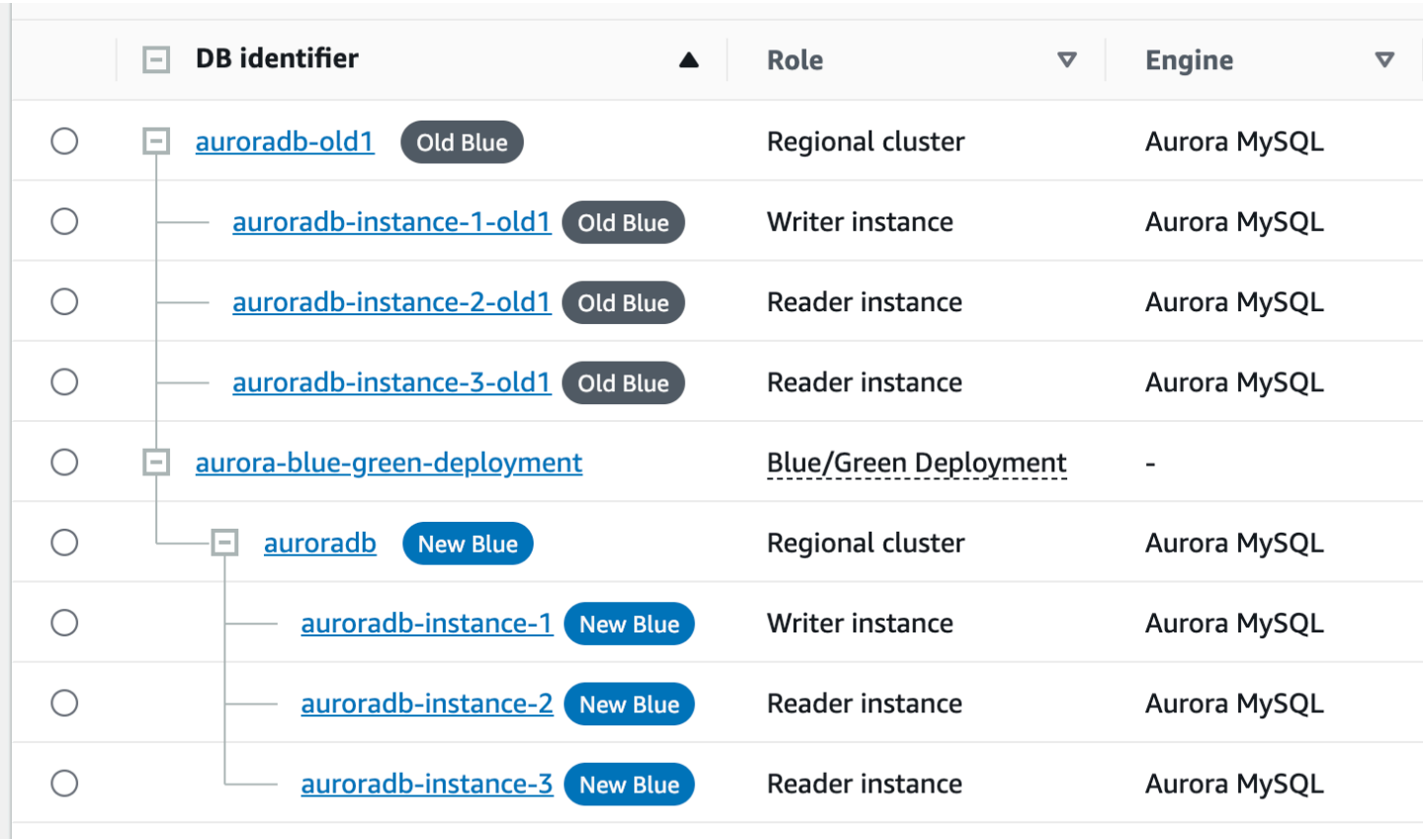
- `BlueGreenDeploymentIdentifier`— Spécifiez l'ID de ressource du blue/green déploiement.
- `SwitchoverTimeout` : spécifiez la limite de temps pour la bascule, en secondes. La valeur par défaut est 300.

Après la bascule

Après une bascule, le cluster de base de données et les instances de base de données de l'environnement bleu précédent sont conservé(e)s. Les coûts standard s'appliquent à ces ressources. La réplication et la journalisation binaire entre les environnements bleu et vert s'arrête.

RDS renomme le cluster de bases de données et les instances de base de données dans l'environnement bleu en ajoutant `-old`*n* au nom de la ressource actuelle, où *n* est un nombre. Le cluster de bases de données bleu est forcé de passer en mode lecture seule. Même si vous activez les opérations d'écriture, il reste en lecture seule jusqu'à ce que vous supprimiez le blue/green déploiement. RDS renomme le cluster de bases de données et les instances de base de données dans l'environnement vert `-new`*n*.

Si vous supprimez la ressource blue/green de déploiement, RDS conserve les `-new`*n* ressources `-old`*n* et.



Mise à jour du nœud parent pour les consommateurs

RDS propose des réplicas en lecture entièrement gérés. Cependant, il offre également la possibilité de configurer des réplicas autogérés, également appelés réplicas externes. Les réplicas externes vous permettent d'utiliser des ressources tierces comme cibles de réplication.

Après avoir transféré un déploiement , si le cluster de base de données d'instance de base de données contenait des répliques externes ou des consommateurs de journaux binaires avant le basculement, vous devez mettre à jour son nœud parent après le basculement afin de maintenir la continuité de la réplication.

Pour mettre à jour le nœud parent

1. Après la bascule, l'instance de base de données de l'enregistreur qui se trouvait auparavant dans l'environnement vert émet un événement contenant le nom et la position du fichier journal principal. Pour localiser l'événement, accédez à la console RDS et choisissez Événements dans le panneau de navigation de gauche.
2. Filtrez par événements dont la source est le nom de l'ancienne instance de base de données verte de l'enregistreur, avant la bascule.
3. Localisez l'événement qui contient les coordonnées du journal binaire. Le message de l'événement est similaire à : `Binary log coordinates in green environment after switchover: file mysql-bin-changelog.`*`000003`*` and position `*`40134574`*.
4. Assurez-vous que le consommateur ou le réplica a appliqué tous les journaux binaires de l'ancien environnement bleu. Utilisez ensuite les coordonnées du journal binaire fournies pour reprendre la réplication sur les consommateurs. Par exemple, si vous exécutez une réplique MySQL sur EC2, vous pouvez utiliser les commandes suivantes :

   MySQL 8.0.22 et versions inférieures majeures et mineures

   ```
   CHANGE MASTER TO MASTER_HOST='{new-writer-endpoint}', MASTER_LOG_FILE='mysql-bin-changelog.000003', MASTER_LOG_POS=40134574;
   ```

   MySQL 8.0.23 et versions ultérieures majeures et mineures

   ```
   CHANGE REPLICATION SOURCE TO SOURCE_HOST='{new-writer-endpoint}',
    SOURCE_LOG_FILE='mysql-bin-changelog.000003', SOURCE_LOG_POS=40134574;
   ```

Suppression d’un déploiement bleu/vert dans Amazon Aurora

Vous pouvez supprimer un déploiement bleu/vert avant ou après son basculement.

Lorsque vous supprimez un déploiement bleu/vert avant de le basculer, Amazon RDS supprime éventuellement le cluster de bases de données dans l’environnement vert :

- Si vous choisissez de supprimer le cluster de bases de données dans l’environnement vert (`--delete-target`), veillez à ce que la protection contre la suppression ne soit pas activée pour lui.
- Si vous ne supprimez pas le cluster de bases de données dans l’environnement vert (`--no-delete-target`), il est conservé, mais ne fait plus partie d’un déploiement bleu/vert. Pour Aurora MySQL, la réplication se poursuit entre les environnements. Pour Aurora PostgreSQL, l’environnement vert est promu en environnement autonome, de sorte que la réplication s’arrête.

L’option permettant de supprimer les bases de données vertes n’est pas disponible dans la console après la bascule. Lorsque vous supprimez des déploiements bleu/vert à l’aide d’AWS CLI, vous ne pouvez pas spécifier l’option `--delete-target` si le statut du déploiement est `SWITCHOVER_COMPLETED`.

> **Important**
>
> Après avoir supprimé un déploiement bleu/vert, RDS supprime les protections en lecture seule du cluster de bases de données de production précédent. Si le paramètre `read_only` est désactivé pour le cluster de bases de données, il recommence à autoriser les opérations d’écriture.

Vous pouvez supprimer un déploiement bleu/vert à l’aide de la AWS Management Console, d’AWS CLI ou de l’API RDS.

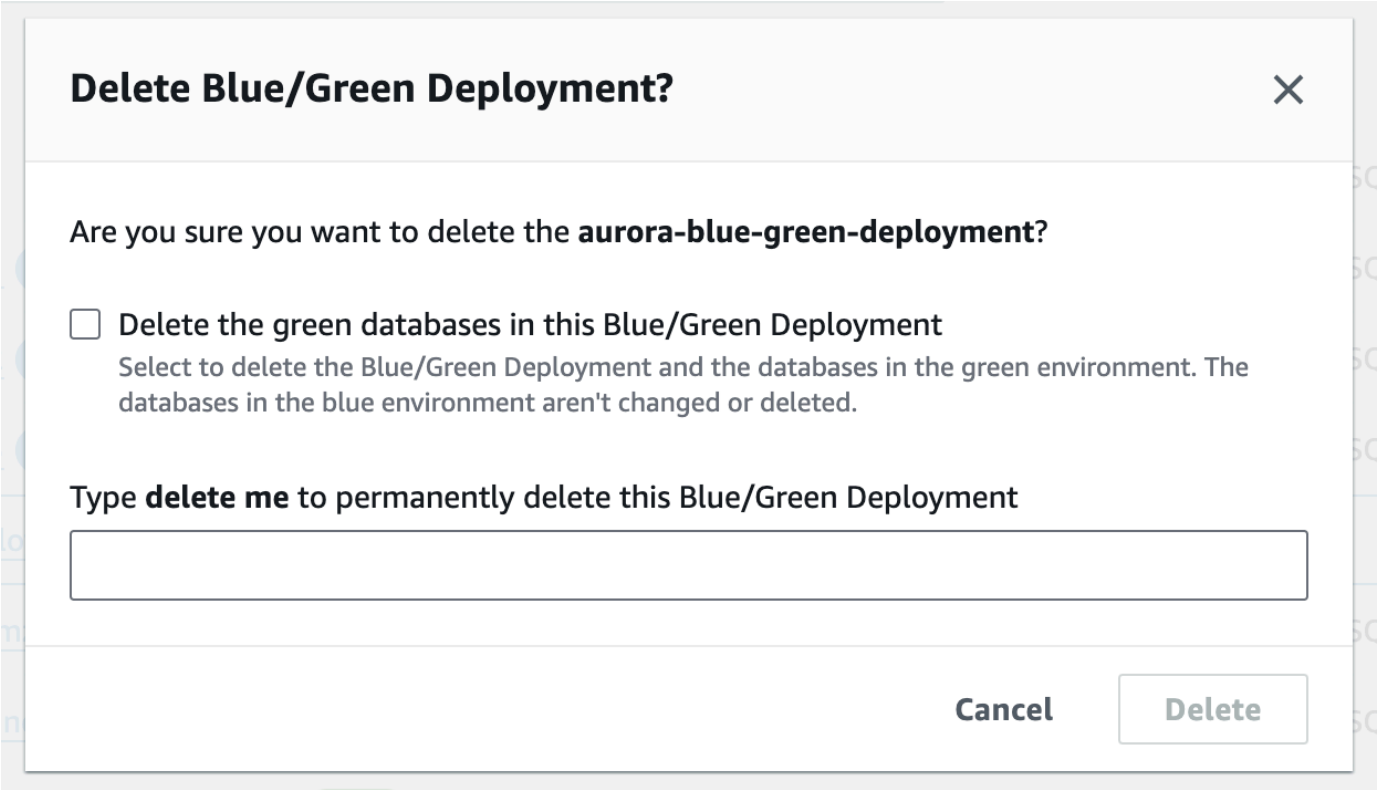
Console

Pour supprimer un déploiement bleu/vert

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Databases (Bases de données), puis choisissez le déploiement bleu/vert que vous souhaitez supprimer.

3. Pour Actions, choisissez Supprimer.

   La fenêtre Supprimer le déploiement bleu/vert ? s'affiche.



   Pour supprimer les bases de données vertes, sélectionnez Supprimer les bases de données vertes dans ce déploiement bleu/vert.

4. Saisissez **`delete me`** dans la zone.
5. Sélectionnez Delete (Supprimer).

AWS CLI

Pour supprimer un déploiement bleu/vert avec AWS CLI, utilisez la commande delete-blue-green-deployment avec les options suivantes :

- `--blue-green-deployment-identifier` : identifiant de ressource du déploiement bleu/vert à supprimer.
- `--delete-target` : spécifie que le cluster de bases de données dans l'environnement vert est supprimé. Vous ne pouvez pas spécifier cette option si le statut du déploiement bleu/vert est `SWITCHOVER_COMPLETED`.

- `--no-delete-target` :spécifie que le cluster de bases de données dans l'environnement vert est conservé.

Example Suppression d'un déploiement bleu/vert et du cluster de bases de données dans l'environnement vert

Pour Linux, macOS ou Unix :

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --delete-target
```

Pour Windows :

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --delete-target
```

Example Suppression d'un déploiement bleu/vert mais conservation du cluster de bases de données dans l'environnement vert

Pour Linux, macOS ou Unix :

```
aws rds delete-blue-green-deployment \
    --blue-green-deployment-identifier bgd-1234567890abcdef \
    --no-delete-target
```

Pour Windows :

```
aws rds delete-blue-green-deployment ^
    --blue-green-deployment-identifier bgd-1234567890abcdef ^
    --no-delete-target
```

API RDS

Pour supprimer un déploiement bleu/vert en utilisant l'API Amazon RDS, utilisez l'opération `DeleteBlueGreenDeployment` avec les paramètres suivants :

- `BlueGreenDeploymentIdentifier` : identifiant de ressource du déploiement bleu/vert à supprimer.

- `DeleteTarget` : spécifiez `TRUE` si vous souhaitez supprimer le cluster de bases de données dans l'environnement vert ou `FALSE` si vous souhaitez le conserver. Ne peut pas être `TRUE` si le statut du déploiement bleu/vert est `SWITCHOVER_COMPLETED`.

Sauvegarde et restauration d’un cluster de bases de données Amazon Aurora

Ces rubriques fournissent des informations sur la sauvegarde et la restauration des clusters de bases de données Amazon Aurora.

> **Tip**
>
> Les fonctions de haute disponibilité et les capacités de sauvegarde automatique Aurora vous aident à protéger vos données sans que vous ayez besoin de procéder à une configuration approfondie. Avant de mettre en œuvre une stratégie de sauvegarde, découvrez comment Aurora gère plusieurs copies de vos données et vous aide à y accéder sur plusieurs instances de base de données et régions AWS. Pour plus d’informations, consultez Haute disponibilité pour Amazon Aurora.

Rubriques

Présentation de la sauvegarde et de la restauration d'un cluster de bases de données Aurora

Les rubriques suivantes décrivent les sauvegardes Aurora et comment restaurer votre cluster de bases de données Aurora.

Table des matières

Sauvegardes

Aurora sauvegarde automatiquement votre volume de cluster et conserve les données de restauration pendant la totalité de la période de rétention des sauvegardes. Les sauvegardes automatisées Aurora étant continues et incrémentielles, vous pouvez rapidement opérer une restauration à un point quelconque de la période de rétention des sauvegardes. Aucun impact sur les performances ou interruption du service de base de données ne se produit lors de l'écriture des données de sauvegarde. Vous pouvez spécifier une période de rétention des sauvegardes comprise entre 1 et 35 jours lorsque vous créez ou modifiez un cluster de bases de données. Les sauvegardes automatisées Aurora sont stockées dans Amazon S3. Pour plus d'informations sur la conservation des sauvegardes automatiques, consultez Conservation des sauvegardes automatiques.

Si vous souhaitez conserver les données au-delà de la période de rétention, vous pouvez aussi réaliser un instantané des données dans votre volume de cluster. Les instantanés de cluster de bases de données Aurora n'expirent pas. Vous pouvez créer un nouveau cluster de bases de données à partir de l'instantané. Pour plus d'informations, consultez Création d'un instantané de cluster de bases de données.

> **Note**
>
> - Pour les clusters de bases de données Amazon Aurora, la période de rétention des sauvegardes par défaut est d'une journée quel que soit le mode de création du cluster de bases de données.
> - Vous ne pouvez pas désactiver les sauvegardes automatiques sur Aurora. La période de rétention des sauvegardes pour Aurora est gérée par le cluster de bases de données.

Le coût de stockage des sauvegardes dépend du volume de données d'instantanés et de sauvegardes Aurora que vous conservez et de la durée pendant laquelle vous les conservez. Pour plus d'informations sur le stockage associé aux instantanés et aux sauvegardes Aurora, consultez Comprendre l'utilisation du stockage de sauvegarde Amazon Aurora. Pour plus d'informations sur les coûts de stockage des sauvegardes Aurora, consultez Tarification d'Amazon RDS for Aurora. Après la suppression du cluster Aurora associé à un instantané, le stockage de cet instantané est soumis aux frais standard de stockage des sauvegardes pour Aurora.

En utilisant AWS Backup

Vous pouvez l'utiliser AWS Backup pour gérer les sauvegardes des clusters de bases de données Amazon Aurora.

Les instantanés gérés par AWS Backup sont considérés comme des instantanés de cluster de base de données manuels, mais ne sont pas pris en compte dans le quota de clichés de cluster de base de données pour Aurora. Les instantanés créés avec AWS Backup sont nommés avec`awsbackup:job-`*`AWS-Backup-job-number`*. Pour plus d'informations AWS Backup, consultez le AWS Backup Developer Guide.

Vous pouvez également l'utiliser AWS Backup pour gérer les sauvegardes automatisées des clusters de bases de données Amazon Aurora. Si votre cluster de base de données est associé à un plan de sauvegarde dans AWS Backup, vous pouvez utiliser ce plan de sauvegarde pour la point-in-time restauration. Les sauvegardes automatiques (continues) gérées par AWS Backup ont des noms comprenant `continuous:cluster-`*`AWS-Backup-job-number`*. Pour plus d'informations, consultez Restauration d'un cluster de bases de données à un instant spécifié en utilisant AWS Backup.

Fenêtre de sauvegarde

Les sauvegardes automatiques sont exécutées chaque jour pendant la fenêtre de sauvegarde préférée. Si la sauvegarde a besoin de plus de temps que la durée allouée par la fenêtre de sauvegarde, elle continue après la fin de la fenêtre jusqu'à ce qu'elle soit terminée. La fenêtre de sauvegarde ne peut pas chevaucher la fenêtre de maintenance hebdomadaire pour le cluster de bases de données.

Les sauvegardes automatisées Aurora sont continues et incrémentielles, mais la fenêtre de sauvegarde permet de créer une sauvegarde système quotidienne qui est conservée dans la période de rétention des sauvegardes. Vous pouvez copier la sauvegarde pour la conserver en dehors de la période de rétention.

> **Note**
>
> Lorsque vous créez un cluster de base de données à l'aide de AWS Management Console, vous ne pouvez pas spécifier de fenêtre de sauvegarde. Toutefois, vous pouvez spécifier une fenêtre de sauvegarde lorsque vous créez un cluster de bases de données à l'aide de l' AWS CLI ou l'API RDS.

Si vous ne spécifiez pas une fenêtre de sauvegarde préférée lorsque vous créez le cluster, Aurora attribue une fenêtre de sauvegarde par défaut de 30 minutes. Cette fenêtre est sélectionnée au hasard sur une période de 8 heures pour chacune Région AWS d'entre elles. Le tableau suivant répertorie les plages temporelles pour chacune des périodes Région AWS à partir desquelles les fenêtres de sauvegarde par défaut sont attribuées.

Nom de la région	Région	Bloc chronologique
USA Est (Virginie du Nord)	us-east-1	03:00–11:00 UTC
USA Est (Ohio)	us-east-2	03:00–11:00 UTC
USA Ouest (Californie du Nord)	us-west-1	06:00–14:00 UTC
US West (Oregon)	us-west-2	06:00–14:00 UTC

Nom de la région	Région	Bloc chronologique
Africa (Cape Town)	af-south-1	03:00–11:00 UTC
Asie-Pacifique (Hong Kong)	ap us-east-1	06:00–14:00 UTC
Asie-Pacifique (Hyderabad)	ap-south-2	6h30–14h30 UTC
Asie-Pacifique (Jakarta)	ap-southeast-3	08:00–16:00 UTC
Asie-Pacifique (Malaisie)	ap-southeast-5	09:00–17:00 UTC
Asie-Pacifique (Melbourne)	ap-southeast-4	11:00–19:00 UTC
Asie-Pacifique (Mumbai)	ap-south-1	16:30–00:30 UTC
Asie-Pacifique (Nouvelle-Zélande)	ap-southeast-6	13:00–21:00 UTC
Asie-Pacifique (Osaka)	ap-northeast-3	00:00–08:00 UTC
Asia Pacific (Seoul)	ap-northeast-2	13:00–21:00 UTC
Asia Pacific (Singapore)	ap-southeast-1	14:00–22:00 UTC
Asia Pacific (Sydney)	ap-southeast-2	12:00–20:00 UTC
Asie-Pacifique (Taipei)	ap-east-2	9:00–17:00 UTC
Asie-Pacifique (Thaïlande)	ap-southeast-7	8:00–16:00 UTC

Nom de la région	Région	Bloc chronologique
Asie-Pacifique (Tokyo)	ap-northeast-1	13:00–21:00 UTC
Canada (Centre)	ca-central-1	03:00–11:00 UTC
Canada-Ouest (Calgary)	ca-west-1	18:00–02:00 UTC
Chine (Pékin)	cn-north-1	06:00–14:00 UTC
China (Ningxia)	cn-northwest-1	06:00–14:00 UTC
Europe (Frankfurt)	eu-central-1	20:00–04:00 UTC
Europe (Ireland)	eu-west-1	22:00–06:00 UTC
Europe (London)	eu-west-2	22:00–06:00 UTC
Europe (Milan)	eu-south-1	02:00–10:00 UTC
Europe (Paris)	eu-west-3	07:29–14:29 UTC
Europe (Espagne)	eu-south-2	02:00–10:00 UTC
Europe (Stockholm)	eu-north-1	23:00–07:00 UTC
Europe (Zurich)	eu-central-2	02:00–10:00 UTC
Israël (Tel Aviv)	il-central-1	03:00–11:00 UTC
Mexique (Centre)	mx-central-1	19:00–03:00 UTC
Moyen-Orient (Bahreïn)	me-south-1	06:00–14:00 UTC
Moyen-Orient (EAU)	me-central-1	05:00–13:00 UTC
Amérique du Sud (São Paulo)	sa-east-1	23:00–07:00 UTC

Nom de la région	Région	Bloc chronologique
AWS GovCloud (USA Est)	us-gov-east-1	17:00–01:00 UTC
AWS GovCloud (US-Ouest)	us-gov-west-1	06:00–14:00 UTC

Restauration des données

Vous pouvez récupérer vos données en créant un nouveau cluster de bases de données Aurora à partir des données de sauvegarde qu'Aurora conserve, à partir d'un instantané de cluster de bases de données que vous avez enregistré ou à partir d'une sauvegarde automatique conservée. Vous pouvez restaurer rapidement une nouvelle copie d'un cluster de bases de données créé à partir des données de sauvegarde à un point quelconque de la période de rétention des sauvegardes. Comme les sauvegardes Aurora sont continues et incrémentielles pendant la période de conservation des sauvegardes, vous n'avez pas besoin de prendre fréquemment des instantanés de vos données pour améliorer les temps de restauration.

La dernière heure de restauration possible d'un cluster de bases de données est le point le plus récent auquel vous pouvez restaurer votre cluster de bases de données. Elle se trouve généralement à moins de 5 minutes de l'heure actuelle pour un cluster de bases de données actif, ou à 5 minutes avant l'heure de suppression du cluster pour une sauvegarde automatique conservée.

L'heure de restauration la plus ancienne spécifie jusqu'à quelle date vous pouvez remonter au sein de la période de conservation des sauvegardes pour restaurer votre volume de cluster.

Pour déterminer la date de restauration la plus ancienne ou la plus récente d'un cluster de bases de données, recherchez les valeurs `Latest restorable time` ou `Earliest restorable time` sur la console RDS. Pour obtenir des informations sur l'affichage de ces valeurs, consultez Affichage des sauvegardes automatiques conservées pour Amazon Aurora.

Vous pouvez déterminer à quel moment la restauration d'un cluster de bases de données est complète à l'aide des valeurs `Latest restorable time` et `Earliest restorable time`. Les valeurs retournent NULL tant que l'opération de restauration n'est pas terminée. Vous ne pouvez pas demander une opération de sauvegarde ou de restauration si `Latest restorable time` ou `Earliest restorable time` retourne NULL.

Pour plus d'informations sur la restauration d'un cluster de bases de données à une date spécifiée, consultez Restauration d'un cluster de bases de données à une date définie.

Clonage de bases de données pour Aurora

Vous pouvez aussi utiliser le clonage de base de données pour cloner les bases de données de votre cluster de bases de données Aurora dans un nouveau cluster de bases de données au lieu de restaurer un instantané de cluster de bases de données. Les bases de données clones n'utilisent qu'un espace supplémentaire minime lorsqu'elles sont créées la première fois. Les données sont copiées uniquement lorsqu'elles changent, sur les bases de données sources ou sur les bases de données clones. Vous pouvez créer plusieurs clones à partir du même cluster de bases de données ou créer des clones supplémentaires même à partir d'autres clones. Pour plus d'informations, consultez Clonage d'un volume pour un cluster de bases de données Amazon Aurora.

Retour sur trace

Aurora MySQL permet désormais d'effectuer un retour en arrière d'un cluster de bases de données à une heure spécifique, sans restaurer les données à partir d'une sauvegarde. Pour plus d'informations, consultez Retour en arrière d'un cluster de bases de données Aurora.

Conservation des sauvegardes automatiques

Lorsque vous supprimez un cluster de bases de données provisionné ou Aurora Serverless v2, vous pouvez conserver les sauvegardes automatiques. Cela vous permet de restaurer un cluster de bases de données à un instant spécifique dans le passé au cours de la période de conservation des sauvegardes, même après la suppression du cluster.

Les sauvegardes automatiques conservées contiennent des instantanés du système et des journaux de transactions d'un cluster de bases de données. Elles incluent également les propriétés du cluster de bases de données, telles que la classe d'instance de base de données, qui sont requises pour le restaurer en un cluster actif.

Vous pouvez restaurer ou supprimer des sauvegardes automatiques conservées à l'aide de la AWS Management Console, de l'API RDS et de l'AWS CLI.

> **Note**
>
> Vous ne pouvez pas conserver les sauvegardes automatiques pour les clusters de bases de données Aurora Serverless v1.

Rubriques

Période de conservation

Les instantanés du système et les journaux de transactions contenus dans une sauvegarde automatique conservée expirent de la même façon que pour le cluster de bases de données source. Les paramètres de la période de conservation du cluster source s'appliquent également aux sauvegardes automatiques. Dans la mesure où aucun nouvel instantané ni journal n'est créé pour ce cluster, les sauvegardes automatiques conservées finissent par expirer complètement. Une fois la

période de conservation terminée, vous continuez à conserver les instantanés manuels du cluster de bases de données, mais toutes les sauvegardes automatiques expirent.

Vous pouvez supprimer des sauvegardes automatiques conservées à l'aide de la console, de l'interface AWS CLI ou de l'API RDS. Pour plus d'informations, consultez Suppression des sauvegardes automatiques conservées pour Amazon Aurora.

Contrairement à une sauvegarde automatique conservée, un instantané final n'expire pas. Nous vous recommandons vivement de réaliser un instantané final même si vous conservez les sauvegardes automatiques car celles-ci finissent par expirer.

Coûts de conservation

Il n'y a pas de frais supplémentaires pour le stockage de sauvegarde jusqu'à 100 % du stockage total de votre base de données Aurora pour chaque cluster de bases de données Aurora. Il n'y a pas non plus de frais supplémentaires jusqu'à un jour lorsque vous conservez des sauvegardes automatiques après la suppression d'un cluster de bases de données. Les sauvegardes que vous conservez pendant plus d'une journée sont facturées.

Il n'y a pas de frais supplémentaires pour les journaux de transactions ou les métadonnées de l'instance. Toutes les autres règles de tarification des sauvegardes s'appliquent aux clusters restaurables. Pour en savoir plus, consultez la page Tarification d'Amazon Aurora.

Empêcher la suppression automatique des sauvegardes

Amazon RDS supprime les sauvegardes automatisées dans plusieurs situations :

- A la fin de leur période de conservation.
- Lorsque vous supprimez un cluster de bases de données.

Si vous souhaitez conserver une sauvegarde automatique à plus long terme, copiez-la pour créer un instantané de base de données manuel qui sera conservé jusqu'à ce que vous le supprimiez. Des coûts de stockage Amazon RDS peuvent s'appliquer aux instantanés manuels si ces derniers dépassent votre espace de stockage par défaut.

Pour plus d'informations sur la copie d'un instantané de cluster de bases de données, consultez Copie d'un instantané de cluster de bases de données.

Pour plus d'informations sur les coûts de stockage des sauvegardes, consultez Tarification d'Amazon RDS.

Limitations

Les limitations suivantes s'appliquent aux sauvegardes automatiques conservées :

- Le nombre maximum de sauvegardes automatiques conservées dans une Région AWS est de 40. Il n'est pas inclus dans le quota pour les clusters de bases de données. Vous pouvez avoir simultanément jusqu'à 40 clusters de bases de données en cours d'exécution, 40 instances de base de données en cours d'exécution et 40 sauvegardes automatiques conservées pour des clusters de bases de données.

  Pour plus d'informations, consultez Quotas dans Amazon Aurora.
- Les sauvegardes automatiques conservées ne contiennent pas d'informations sur les paramètres ou les groupes d'options.
- Vous pouvez restaurer un cluster supprimé jusqu'à un instant dans le passé compris dans la période de conservation au moment de la suppression.
- Vous ne pouvez pas modifier une sauvegarde automatique conservée, car elle est composée des sauvegardes du système, des journaux de transactions et des propriétés de cluster de bases de données qui existaient au moment où vous avez supprimé le cluster source.
- La réplication de sauvegarde automatique entre régions n'est pas prise en charge pour les clusters de bases de données Aurora. Aurora ne prend pas en charge la réplication automatique des instantanés et des journaux de transactions vers une autre Région AWS. Pour une reprise après sinistre entre régions, vous devez copier manuellement les instantanés Aurora dans la région de destination souhaitée.

Affichage des sauvegardes automatiques conservées pour Amazon Aurora

Pour afficher vos sauvegardes automatisées conservées dans la console RDS, choisissez Sauvegardes automatiques dans le panneau de navigation, puis choisissez Rétention. Pour afficher des instantanés individuels associés à une sauvegarde automatisée conservée, choisissez Snapshots (Instantanés) dans le panneau de navigation. Vous pouvez également décrire les instantanés individuels associés à une sauvegarde automatique conservée. À partir de là, vous pouvez restaurer une instance de base de données directement à partir d'un de ces instantanés.

Pour décrire vos sauvegardes automatiques conservées avec la AWS CLI, utilisez l'une des commandes suivantes :

```
aws rds describe-db-cluster-automated-backups --db-cluster-resource-
id DB_cluster_resource_ID
```

Pour décrire vos sauvegardes automatiques conservées à l'aide de l'API RDS, appelez l'action `DescribeDBClusterAutomatedBackups` avec le paramètre `DbClusterResourceId` :

Suppression des sauvegardes automatiques conservées pour Amazon Aurora

Vous pouvez supprimer les sauvegardes automatiques conservées quand elles ne sont plus nécessaires. Pour supprimer une sauvegarde automatique conservée à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API Amazon RDS, utilisez les procédures suivantes.

Console

Pour supprimer une sauvegarde automatisée conservée

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Automated backups (Sauvegardes automatisées).
3. Choisissez l'onglet Rétention.

4. Choisissez la sauvegarde automatisée conservée que vous souhaitez supprimer.
5. Pour Actions, choisissez Supprimer.
6. Dans la page de confirmation, entrez **`delete me`** et choisissez Delete (Supprimer).

AWS CLI

Vous pouvez supprimer une sauvegarde automatique conservée à l'aide de la commande AWS CLI delete-db-cluster-automated-backup avec l'option suivante :

- `--db-cluster-resource-id` : identifiant de ressource pour le cluster de bases de données source.

  Vous pouvez rechercher l'identifiant de ressource pour le cluster de bases de données source d'une sauvegarde automatique conservée en exécutant la commande AWS CLI describe-db-cluster-automated-backups.

Example

Cet exemple supprime la sauvegarde automatique conservée pour le cluster de bases de données source qui possède l'ID de ressource `cluster-123ABCEXAMPLE`.

Pour Linux, macOS ou Unix :

```
aws rds delete-db-cluster-automated-backup \
    --db-cluster-resource-id cluster-123ABCEXAMPLE
```

Pour Windows :

```
aws rds delete-db-cluster-automated-backup ^
    --db-cluster-resource-id cluster-123ABCEXAMPLE
```

API RDS

Vous pouvez supprimer une sauvegarde automatique conservée en utilisant l'opération d'API Amazon RDS DeleteDBClusterAutomatedBackup avec le paramètre suivant :

- `DbClusterResourceId` : identifiant de ressource pour le cluster de bases de données source.

  Vous pouvez rechercher l'identifiant de ressource pour l'instance de base de données source d'une sauvegarde automatique conservée à l'aide de l'opération d'API Amazon RDS DescribeDBClusterAutomatedBackups.

Comprendre l’utilisation du stockage de sauvegarde Amazon Aurora

Amazon Aurora gère deux types de sauvegardes : les sauvegardes automatiques (continues) et les instantanés.

Stockage de sauvegarde automatique

La sauvegarde automatique (continue) d’un cluster stocke de manière incrémentielle toutes les modifications apportées aux bases de données au cours d’une période de conservation spécifiée, afin de pouvoir les restaurer telles qu’elles étaient à un instant quelconque de cette période de conservation. Les périodes de conservation peuvent aller de 1 à 35 jours. Les sauvegardes automatiques sont incrémentielles et facturées en fonction de la quantité de stockage requis pour pouvoir restaurer à un instant quelconque de la période de conservation.

Aurora fournit également une quantité d’utilisation gratuite des sauvegardes. Cette quantité d’utilisation gratuite est égale à la taille du dernier volume du cluster (telle que représentée par la métrique Amazon CloudWatch `VolumeBytesUsed`). Cette quantité est soustraite de l’utilisation calculée des sauvegardes automatiques. Aucun frais ne s’applique pour une sauvegarde automatique dont la durée de conservation n’est que d’un jour.

Par exemple, votre sauvegarde automatique a une période de conservation de 7 jours et vous souhaitez restaurer votre cluster dans l’état où il était quatre jours auparavant. Aurora utilise les données incrémentielles stockées dans la sauvegarde automatique pour recréer l’état du cluster à la même heure, quatre jours plus tôt.

La sauvegarde automatique stocke toutes les informations requises pour pouvoir restaurer le cluster à un instant quelconque de la fenêtre de conservation. Cela signifie qu’elle stocke toutes les modifications effectuées pendant cette fenêtre de conservation, y compris les écritures de nouvelles informations ou la suppression d’informations existantes. Pour les bases de données soumises à de nombreuses modifications, la taille de la sauvegarde automatique augmente au fil du temps. Une fois qu’une base de données ne fait plus l’objet de modifications, vous pouvez vous attendre à ce que la taille de la sauvegarde automatique diminue, car des modifications précédemment stockées quittent la fenêtre de conservation.

L’utilisation totale facturée pour la sauvegarde automatique ne dépasse jamais la taille cumulée du volume du cluster au cours de la période de conservation. Par exemple, si votre période de

conservation est de 7 jours et que le volume de votre cluster était de 100 Go par jour, l'utilisation de la sauvegarde automatique facturée ne dépasse jamais 700 Go (100 Go * 7).

Stockage d'instantanés

Les instantanés de clusters de bases de données sont toujours des sauvegardes complètes qui capturent la taille du volume du cluster lorsque vous les créez. Que vous créiez des instantanés manuellement ou via un plan AWS Backups, Aurora les traite comme des instantanés manuels. Aurora fournit un espace de stockage gratuit illimité pour les instantanés compris dans la période de conservation des sauvegardes automatiques. Lorsqu'un instantané manuel est en dehors de cette période, il est facturé en fonction du nombre de giga-octets par mois. Les instantanés automatiques du système restent gratuits, sauf si vous les copiez. Comme les sauvegardes automatiques ne couvrent pas les copies d'instantanés, AWS les facture toujours.

Pour obtenir des informations générales sur les sauvegardes Aurora, consultez Sauvegardes. Pour plus d'informations sur les coûts de stockage des sauvegardes Aurora, consultez la page Tarification d'Amazon Aurora.

Métriques Amazon CloudWatch pour le stockage de sauvegarde Aurora

Vous pouvez surveiller vos clusters Aurora et créer des rapports à l'aide de métriques Amazon CloudWatch dans la console CloudWatch. Vous pouvez utiliser les métriques CloudWatch suivantes pour consulter et surveiller la quantité de stockage utilisée par vos sauvegardes Aurora. Ces métriques sont calculées de manière indépendante pour chaque cluster de bases de données Aurora.

- `BackupRetentionPeriodStorageUsed` représente la quantité de stockage de sauvegarde utilisée, en octets, pour stocker les sauvegardes automatiques à l'heure actuelle.
  - La valeur dépend de la taille du volume du cluster et du nombre de modifications (écritures et mises à jour) apportées au cluster de bases de données pendant la période de conservation. Cela est dû au fait que la sauvegarde automatique doit stocker toutes les modifications incrémentielles apportées au cluster pour permettre une restauration à tout instant donné.
  - Cette métrique ne soustrait pas le niveau gratuit d'utilisation de la sauvegarde fourni par Aurora.
  - Cette métrique émet un seul point de données quotidien pour l'utilisation des sauvegardes automatiques enregistrée ce jour-là.
- `SnapshotStorageUsed` : représente la quantité de stockage de sauvegarde utilisée, en octets, pour stocker les instantanés manuels au-delà de la période de conservation de la sauvegarde automatisée.

- Cette valeur dépend du nombre d'instantanés que vous conservez au-delà de la période de conservation de la sauvegarde automatisée et de la taille de chaque instantané.
- La taille de chaque instantané correspond à la taille du volume de cluster au moment où vous avez pris l'instantané.
- Les instantanés sont des sauvegardes complètes, non incrémentielles.
- Cette métrique émet un point de données par jour pour chaque instantané facturé. Pour récupérer votre utilisation quotidienne totale d'instantanés, effectuez la somme de cette métrique sur une période d'un jour.

- `TotalBackupStorageBilled` : représente les métriques relatives à l'ensemble de l'utilisation facturée des sauvegardes, en octets, pour le cluster donné :

  ```
  BackupRetentionPeriodStorageUsed + SnapshotStorageUsed - free tier
  ```

  - Cette métrique émet un point de données par jour pour la valeur `BackupRetentionPeriodStorageUsed`, moins le niveau gratuit d'utilisation des sauvegardes fourni par Aurora. Ce niveau gratuit est égal à la dernière taille enregistrée du volume du cluster de bases de données. Ce point de données représente l'utilisation facturée réelle pour la sauvegarde automatique.
  - Cette métrique émet des points de données quotidiens individuels pour toutes les valeurs `SnapshotStorageUsed`.
  - Pour récupérer votre utilisation quotidienne totale facturée des sauvegardes, effectuez la somme de cette métrique sur une période d'un jour. Elle cumule toute l'utilisation facturée d'instantanés avec l'utilisation facturée de sauvegardes automatiques pour fournir votre utilisation totale facturée de sauvegardes.

Pour plus d'informations sur l'utilisation des métriques CloudWatch, consultez Disponibilité des métriques Aurora dans la console Amazon RDS.

Calcul de l'utilisation du stockage de sauvegarde

L'utilisation d'une sauvegarde automatique est calculée en examinant tous les enregistrements incrémentiels qui doivent être stockés, pour permettre une restauration à tout instant donné au cours de la période de conservation de la sauvegarde. Ces modifications concernent non seulement le nombre d'opérations d'écriture, mais également la taille et l'étendue des modifications de données. Chaque type d'opération (INSERT, UPDATE, DELETE) crée des enregistrements de modifications qui doivent être conservés pour une reprise ponctuelle. Par conséquent, bien que deux bases de

données puissent avoir le même nombre d'opérations d'écriture (IOPS), leurs exigences en matière de stockage de sauvegarde peuvent différer considérablement en fonction du volume de données modifié lors de chaque transaction.

Par exemple, vous disposez d'une sauvegarde automatique avec une période de conservation de 7 jours. La taille du volume de votre cluster juste avant la période de conservation était de 100 Go. Il s'agit donc de la quantité minimale dont Aurora a besoin pour stocker. Vous avez ensuite l'activité suivante pour les 7 jours suivants, où la taille incrémentielle des enregistrements correspond à la quantité de stockage nécessaire pour stocker les enregistrements de modifications provenant des écritures et des mises à jour de votre base de données.

jour	Taille d'enregistrement incrémentielle (Go)
1	10
2	15
3	25
4	20
5	10
6	25
7	30
Total	135

Ces données indiquent que l'utilisation calculée de sauvegarde automatique pour votre sauvegarde est la suivante :

```
100 GB (volume size before retention period) + 135 GB (size of incremental records) =
 235 GB total backup usage
```

L'utilisation facturée soustrait ensuite le niveau d'utilisation gratuit. Supposons que la taille la plus récente de votre volume est de 200 Go :

```
235 GB total backup usage - 200 GB (latest volume size) = 35 GB billed backup usage
```

FAQ

Quand suis-je facturé pour les instantanés ?

Vous êtes facturé pour les instantanés manuels qui se situent en dehors de (plus anciens que) la période de conservation de la sauvegarde automatique.

Qu’est-ce qu’un instantané manuel ?

Un instantané manuel est un instantané auquel l’une des conditions suivantes s’applique :

- Il a été manuellement demandé par vous.
- Il a été capturé par un service de sauvegarde automatique tel qu’AWS Backup.
- Il est copié à partir d’un instantané automatique du système afin d’être conservé en dehors de la période de conservation

Qu’arrive-t-il à mes instantanés manuels si je supprime mon cluster de bases de données ?

Les instantanés manuels n’expirent pas tant que vous ne les supprimez pas.

Lorsque vous supprimez votre cluster de bases de données, les instantanés manuels que vous avez capturés précédemment continuent d’exister. Si ces instantanés n’étaient pas facturés auparavant parce qu’ils se trouvaient dans la période de conservation des sauvegardes automatiques, ils ne sont désormais plus couverts et commencent tous à être facturés à leur taille complète pour leur utilisation.

Comment puis-je réduire mes coûts de stockage de sauvegarde ?

Il existe plusieurs moyens de réduire les coûts liés à l’utilisation des sauvegardes :

- Supprimez les instantanés manuels qui se situent en dehors de la période de conservation de votre sauvegarde automatique. Cela inclut les instantanés que vous avez capturés, ainsi que les instantanés que votre plan de sauvegarde AWS Backup a pu capturer. Veillez à vérifier votre plan AWS Backup pour vous assurer qu’il ne conserve pas les instantanés en dehors de la période de conservation à laquelle vous ne vous attendez pas.
- Évaluez les écritures et les mises à jour apportées à votre base de données pour voir si vous pouvez réduire le nombre de modifications que vous effectuez. Comme notre sauvegarde automatique stocke toutes les modifications incrémentielles au cours de la période de conservation, la réduction du nombre de mises à jour que vous effectuez réduit également vos frais de sauvegarde automatique.
- Déterminez s’il est judicieux de réduire la période de conservation de votre sauvegarde automatique. La réduction de la période de conservation signifie que la sauvegarde stocke

moins de jours de données incrémentielles, ce qui peut réduire le coût global de la sauvegarde. Toutefois, la réduction de cette période de conservation peut également entraîner la facturation de certains instantanés, qui se trouvent désormais hors de la période de conservation. Veillez à vérifier tous les coûts supplémentaires que vous pourriez encourir pour les instantanés avant de décider s'il s'agit d'une solution adéquate pour vous.

Comment le stockage de sauvegarde est-il facturé ?

Le stockage de sauvegarde est facturé au Go par mois.

Cela signifie que l'utilisation du stockage de sauvegarde est facturée comme la moyenne pondérée de l'utilisation sur le mois donné. Voici quelques exemples pour un mois de 30 jours :

- L'utilisation facturée des sauvegardes est de 100 Go pour les 30 jours du mois. Vos frais sont les suivants :

```
(100 GB * 30) / 30 = 100 GB-month
```

- L'utilisation facturée des sauvegardes est de 100 Go pour les 15 premiers jours du mois, puis de 0 Go pour les 15 derniers. Vos frais sont les suivants :

```
(100 GB * 15 + 0 GB * 15) / 30 = 50 GB-month
```

- L'utilisation facturée des sauvegardes est de 50 Go pour les 10 premiers jours du mois, de 100 Go pour les 10 suivants, puis de 150 Go pour les 10 derniers. Vos frais sont les suivants :

```
(50 GB * 10 + 100 GB * 10 + 150 GB * 10) / 30 = 100 GB-month
```

Comment le paramètre de retour en arrière de mon cluster de bases de données affecte-t-il l'utilisation du stockage de sauvegarde ?

Le paramètre de retour en arrière d'un cluster de bases de données Aurora n'affecte pas le volume de données de sauvegarde de ce cluster. Amazon facture séparément le stockage des données de retour sur trace. Pour obtenir des informations sur les prix du retour en arrière d'Aurora, consultez la page Amazon Aurora pricing (Tarification d'Amazon Aurora).

Comment les coûts de stockage s'appliquent-ils aux instantanés partagés ?

Si vous partagez un instantané avec un autre utilisateur, vous continuez d'en être le propriétaire. Les coûts du stockage s'appliquent au propriétaire de l'instantané. Si vous supprimez un instantané partagé dont vous êtes propriétaire, personne ne peut y accéder.

Pour conserver l’accès à un instantané partagé mais détenu par une autre personne, vous pouvez copier cet instantané. En agissant ainsi, vous devenez le propriétaire du nouvel instantané. Les coûts de stockage associés à l’instantané copié s’appliquent à votre compte.

Pour plus d’informations sur le partage d’instantanés, consultez Partage d’un instantané de cluster de bases de données. Pour plus d’informations sur la copie d’instantanés, consultez Copie d’un instantané de cluster de bases de données.

Création d’un instantané de cluster de bases de données

Amazon RDS crée un instantané du volume de stockage de votre cluster de bases de données en sauvegardant l’intégralité de ce dernier, et pas seulement les bases de données. Lorsque vous créez un instantané de cluster DB, vous devez identifier le cluster DB que vous allez sauvegarder, puis nommer votre instantané de cluster DB afin de pouvoir effectuer une restauration à partir de ce dernier ultérieurement. Le temps nécessaire à la création d’un instantané de cluster DB varie en fonction de la taille de vos bases de données. Étant donné que l’instantané inclut l’intégralité du volume de stockage, la taille des fichiers, comme les fichiers temporaires, a également une incidence sur le temps nécessaire à la création de l’instantané.

> **Note**
>
> Votre cluster de bases de données doit être dans l’état `available` pour prendre un instantané du cluster de bases de données.

Contrairement aux sauvegardes automatisées, les instantanés manuels ne sont pas soumis à la période de rétention des sauvegardes. Les instantanés n’expirent pas.

Pour les sauvegardes à très long terme, nous vous recommandons d’exporter les données d’instantané vers Amazon S3. Si la version majeure de votre moteur de base de données n’est plus prise en charge, vous ne pouvez pas restaurer cette version à partir d’un instantané. Pour de plus amples informations, veuillez consulter Exportation de données d’instantanés de cluster de bases de données vers Amazon S3.

Vous pouvez créer un instantané de cluster de base de données à l'aide de l'APIAWS Management Console, deAWS CLI, ou de l'API RDS.

Console

Pour créer un instantané de cluster de bases de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Snapshots.

   La liste des instantanés manuels s’affiche.
3. Choisissez Prendre un instantané.

La fenêtre Capture d'un instantané de base de données apparaît.

4. Pour Type d'instantané, sélectionnez Cluster de bases de données.
   Réalisez un instantané de base de données.
5. Choisissez le Cluster de bases de données dont vous voulez prendre un instantané.
6. Entrez le Nom de l'instantané.
7. Choisissez Prendre un instantané.

   La liste Instantanés manuels s'affiche avec le nouvel instantané de cluster de bases de données dont l'état est `Creating`. Une fois que l'état de l'instantané est `Available`, vous pouvez voir son heure de création.

AWS CLI

Lorsque vous créez un instantané de cluster de base de données à l'aide deAWS CLI, vous devez identifier le cluster de base de données que vous allez sauvegarder, puis donner un nom à votre instantané de cluster de base de données afin de pouvoir le restaurer ultérieurement. Pour ce faire, utilisez la AWS CLI `create-db-cluster-snapshot`commande avec les paramètres suivants :

- `--db-cluster-identifier`
- `--db-cluster-snapshot-identifier`

Dans cet exemple, vous créez un instantané de cluster de base de données nommé *`mydbclustersnapshot`* d'après un cluster de base de données appelé*`mydbcluster`*.

Example

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier mydbcluster \
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

Pour Windows :

```
aws rds create-db-cluster-snapshot ^
    --db-cluster-identifier mydbcluster ^
```

```
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

API RDS

Lorsque vous créez un instantané de cluster de bases de données par l'intermédiaire de l'API Amazon RDS, vous devez identifier le cluster de bases de données que vous allez sauvegarder, puis nommer votre instantané de cluster de bases de données afin de pouvoir effectuer une restauration à partir de ce dernier ultérieurement. Pour ce faire, vous pouvez utiliser la commande de l'API Amazon RDS `CreateDBClusterSnapshot` avec les paramètres suivants :

- DBClusterIdentifiant
- DBClusterSnapshotIdentifier

Vérification de la disponibilité de l'instantané de cluster de bases de données

Vous pouvez vérifier que l'instantané du cluster de base de données est disponible en consultant la section Instantanés de l'onglet Maintenance et sauvegardes de la page détaillée du cluster dans leAWS Management Console, en utilisant la commande `describe-db-cluster-snapshots`CLI ou en utilisant l'action `DescribeDBClusterSnapshots`API.

Vous pouvez également utiliser la commande de l'interface de commande `wait db-cluster-snapshot-available` pour interroger l'API toutes les 30 secondes jusqu'à ce que l'instantané soit disponible.

Restauration à partir d'un instantané de cluster de bases de données

Amazon RDS crée un instantané du volume de stockage de votre cluster de bases de données en sauvegardant l'intégralité de ce dernier, et pas seulement les bases de données. Vous pouvez créer un cluster de bases de données en effectuant une restauration à partir de cet instantané de base de données. Vous indiquez le nom de l'instantané de cluster de bases de données à partir duquel opérer la restauration, puis un nom pour le nouveau cluster de bases de données résultant de l'opération de restauration. Vous ne pouvez pas restaurer un cluster de bases de données existant à partir d'un instantané de cluster de bases de données. Un nouveau cluster de bases de données est généré lors de la restauration.

Important

Vous ne pouvez pas restaurer un instantané sur une version du moteur de base de données qui a dépassé sa date de fin de support standard Aurora. Vous ne pouvez accéder à une base de données qu'après sa mise à niveau réussie vers une version prise en charge. Pour plus d'informations sur les versions du moteur de base de données Aurora prises en charge, consultezSupport étendu Amazon RDS avec Amazon Aurora.
Si la mise à niveau vers une version prise en charge de votre cluster échoue, l'état du cluster `upgrade_failed` devient et Aurora crée un instantané final avec le préfixe`rds-final`.
Pour accéder à votre base de données restaurée sur la version obsolète après un échec de mise à niveau, contactez le Support AWS .

Après restauration du cluster de bases de données, vous pouvez l'utiliser dès que son statut est `available`.

Vous pouvez l'utiliser CloudFormation pour restaurer un cluster de base de données à partir d'un instantané de cluster de base de données. Pour plus d'informations, consultez AWS : :RDS : : DBCluster dans le guide de l'AWS CloudFormation utilisateur.

Note

Le partage manuel d'un instantané de cluster de base de données, qu'il soit chiffré ou non, permet aux AWS comptes autorisés de restaurer directement un cluster de base de données à partir de l'instantané au lieu d'en prendre une copie et de le restaurer à partir de celui-ci.

Pour de plus amples informations, veuillez consulter Partage d'un instantané de cluster de bases de données.

Pour plus d'informations sur la restauration d'un cluster de bases de données Aurora ou d'un cluster global avec une version de support étendu RDS, consultez Restauration d'un cluster de bases de données Aurora ou d'un cluster global avec le support étendu Amazon RDS.

Considérations relatives au groupe de paramètres

Nous vous recommandons de conserver le groupe de paramètres de base de données et le groupe de paramètres de cluster de bases de données pour tous les instantanés de cluster de bases de données que vous créez, de manière à pouvoir associer votre cluster de bases de données restauré aux groupes de paramètres appropriés.

Le groupe de paramètres de base de données par défaut et le groupe de paramètres de cluster de bases de données sont associés au cluster restauré, sauf si vous en choisissez des autres. Aucun paramètre personnalisé n'est disponible dans les groupes de paramètres par défaut.

Vous pouvez spécifier les groupes de paramètres lorsque vous restaurez l'instancele cluster de bases de données.

Pour plus d'informations sur les groupes de paramètres de base de données et les groupes de paramètres de cluster de bases de données, consultez Groupes de paramètres pour Amazon Aurora.

Considérations relatives aux groupes de sécurité

Lorsque vous restaurez un cluster de bases de données, le cloud privé virtuel (VPC) par défaut, le groupe de sous-réseaux de base de données et le groupe de sécurité du VPC sont associés à l'instance restaurée, sauf si vous en choisissez d'autres.

- Si vous utilisez la console Amazon RDS, vous pouvez spécifier un groupe de sécurité de VPC personnalisé à associer au cluster ou créer un nouveau groupe de sécurité de VPC.
- Si vous utilisez le AWS CLI, vous pouvez spécifier un groupe de sécurité VPC personnalisé à associer au cluster en incluant l'`--vpc-security-group-ids`option dans la `restore-db-cluster-from-snapshot` commande.
- Si vous utilisez l'API Amazon RDS, vous pouvez inclure le paramètre `VpcSecurityGroupIds.VpcSecurityGroupId.N` dans l'action `RestoreDBClusterFromSnapshot`.

Dès que la restauration est terminée et que votre nouveau cluster de bases de données est disponible, vous pouvez également changer les paramètres de VPC en modifiant le cluster de bases de données. Pour plus d'informations, consultez Modification d'un cluster de bases de données Amazon Aurora.

Considérations relatives à Amazon Aurora

Avec Aurora, vous restaurez un instantané de cluster de bases de données dans un cluster de bases de données.

Aurora MySQL et Aurora PostgreSQL vous permettent également de restaurer un instantané de cluster de bases de données dans un cluster de bases de données Aurora Serverless. Pour plus d'informations, consultez Restauration d'un cluster de bases de données Aurora Serverless v1.

Aurora MySQL vous permet de restaurer un instantané de cluster de bases de données à partir d'un cluster sans requête parallèle à un cluster avec une requête parallèle. Le mécanisme d'instantané est la manière la plus rapide d'ingérer de grands volumes de données à un cluster Aurora MySQL à requête parallèle activée, car la requête parallèle est généralement utilisée avec de très grands tableaux. Pour plus d'informations, consultez Requêtes parallèles pour Amazon Aurora MySQL.

Restaurer à partir d'un instantané

Vous pouvez restaurer un cluster de bases de données à partir d'un instantané de cluster de bases de données à l'aide d' AWS Management Console, d' AWS CLI ou de l'API RDS.

Console

Pour restaurer un cluster DB à partir d'un instantané de cluster DB

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Snapshots.
3. Choisissez l'instantané de cluster de bases de données à partir duquel vous voulez restaurer.
4. Pour Actions, choisissez Restaurer l'instantané.

   La page Restaurer un instantané s'affiche.
5. Choisissez la version du moteur de base de données dans laquelle vous souhaitez restaurer le cluster de bases de données.

Par défaut, l'instantané est restauré dans la version du moteur de base de données du cluster de bases de données source, si cette version est disponible.

6. Pour DB instance identifier (Identifiant de l'instance de base de données), saisissez le nom de l'instance de base de données restaurée. Notez qu'Amazon RDS déduit l'identifiant du cluster de bases de données à partir de l'identifiant d'instance de base de données que vous spécifiez.
7. Spécifiez d'autres paramètres, tels que la configuration de stockage du cluster de bases de données.

   Pour plus d'informations sur chaque paramètre, consultez Paramètres pour les clusters de bases de données Aurora.
8. Choisissez Restore DB Cluster (Restaurer un cluster de bases de données).

AWS CLI

Pour restaurer un cluster de base de données à partir d'un instantané de cluster de base de données, utilisez la AWS CLI commande restore-db-cluster-from-snapshot.

Dans cet exemple, vous effectuez la restauration à partir d'un instantané de cluster de bases de données précédemment créé, nommé `mydbclustersnapshot`. Vous effectuez la restauration à un nouveau cluster de bases de données nommé `mynewdbcluster`.

Vous pouvez spécifier d'autres paramètres, tels que la version du moteur de base de données. Si vous ne spécifiez pas de version de moteur, le cluster de bases de données est restauré dans la version de moteur par défaut.

Pour plus d'informations sur chaque paramètre, consultez Paramètres pour les clusters de bases de données Aurora.

Example

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine aurora-mysql|aurora-postgresql
```

Pour Windows :

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mynewdbcluster ^
    --snapshot-identifier mydbclustersnapshot ^
    --engine aurora-mysql|aurora-postgresql
```

Lorsque le cluster de bases de données a été restauré, vous devez ajouter le cluster de bases de données au groupe de sécurité utilisé par le cluster de bases de données utilisée pour créer l'instantané de base de données si vous souhaitez profiter de la même fonctionnalité que celle du cluster de bases de données précédent.

> **Important**
>
> Si vous utilisez la console pour restaurer un cluster de bases de données, Amazon RDS crée automatiquement l'instance de base de données principale (auteur) pour votre cluster de bases de données. Si vous utilisez l' AWS CLI pour restaurer un cluster de bases de données, vous devez créer explicitement l'instance principale pour votre cluster de bases de données. L'instance principale est la première instance créée dans un cluster de bases de données. Si vous ne créez pas l'instance de base de données principale, les points de terminaison du cluster de bases de données conservent le statut `creating`.
> Appelez la create-db-instance AWS CLI commande pour créer l'instance principale de votre cluster de base de données. Incluez le nom du cluster de bases de données comme valeur de l'option `--db-cluster-identifier`.

API RDS

Pour restaurer un cluster de base de données à partir d'un instantané de cluster de base de données, appelez l'opération Restore de l'API RDS DBCluster FromSnapshot avec les paramètres suivants :

- `DBClusterIdentifier`
- `SnapshotIdentifier`

> **Important**
>
> Si vous utilisez la console pour restaurer un cluster de bases de données, Amazon RDS crée automatiquement l'instance de base de données principale (auteur) pour votre cluster de bases de données. Si vous utilisez l'API RDS pour restaurer un cluster de bases de données,

vous devez créer explicitement l’instance principale pour votre cluster de bases de données. L’instance principale est la première instance créée dans un cluster de bases de données. Si vous ne créez pas l’instance de base de données principale, les points de terminaison du cluster de bases de données conservent le statut `creating`.
Appelez l'opération Create de l'API RDS DBInstance pour créer l'instance principale de votre cluster de base de données. Incluez le nom du cluster de bases de données comme valeur de paramètre `DBClusterIdentifier`.

Copie d’un instantané de cluster de bases de données

Avec Amazon Aurora, vous pouvez copier des sauvegardes automatisées ou des instantanés de cluster de bases de données manuels. Après avoir copié un instantané, la copie est un instantané manuel. Vous pouvez effectuer plusieurs copies d’une sauvegarde automatisée ou d’un instantané manuel, mais chaque copie doit avoir un identifiant unique.

Vous pouvez copier un instantané à l'intérieur de celui-ci Région AWS, vous pouvez copier un instantané par-dessus Régions AWS et vous pouvez copier des instantanés partagés. Vous pouvez copier des instantanés vers un autre compte Région AWS ou vers un autre compte en une seule étape.

> **Note**
>
> Amazon facture en fonction du nombre de données de sauvegarde et d’instantané Amazon Aurora que vous conservez et de la durée de rétention. Pour plus d’informations sur le stockage associé aux instantanés et aux sauvegardes Aurora, consultez Comprendre l’utilisation du stockage de sauvegarde Amazon Aurora. Pour plus d’informations sur la tarification du stockage Aurora, consultez Tarification de Amazon RDS pour Aurora.

Passez en revue les limites et les considérations relatives à la copie des instantanés de cluster de bases de données. Pour copier des instantanés de clusters de bases de données, consultez l’une des rubriques suivantes.

- Copie d’un instantané de cluster de bases de données à l’aide de la AWS Management Console
- Copie d’un instantané de cluster de bases de données non chiffré à l’aide de l’AWS CLI ou de l’API Amazon RDS
- Copie d’un instantané de cluster de bases de données chiffré à l’aide de l’AWS CLI ou de l’API Amazon RDS
- Copie d’un instantané de cluster de bases de données entre des comptes

Copie d’un instantané de cluster de bases de données à l’aide de la AWS Management Console

Utilisez les procédures de cette rubrique pour copier un instantané de cluster de bases de données. Si votre moteur de base de données source est Aurora, votre instantané est un instantané de cluster de bases de données.

Pour chaque compte AWS, vous pouvez copier jusqu’à cinq instantanés de cluster de bases de données à la fois d’une Région AWS vers une autre. La copie des instantanés de cluster de bases de données chiffrés et non chiffrés est prise en charge. Si vous copiez un instantané de cluster de bases de données dans une autre Région AWS, vous créez un instantané de cluster de bases de données manuel qui est conservé dans cette Région AWS. La copie d’un instantané de cluster de bases de données hors de la Région AWS source entraîne des frais de transfert de données Amazon RDS.

Pour plus d’informations sur la tarification du transfert des données, consultez Tarification d’Amazon RDS.

Une fois que la copie de l’instantané de cluster de bases de données a été créée dans la nouvelle Région AWS, elle se comporte de la même façon que tous les autres instantanés de cluster de bases de données dans cette Région AWS.

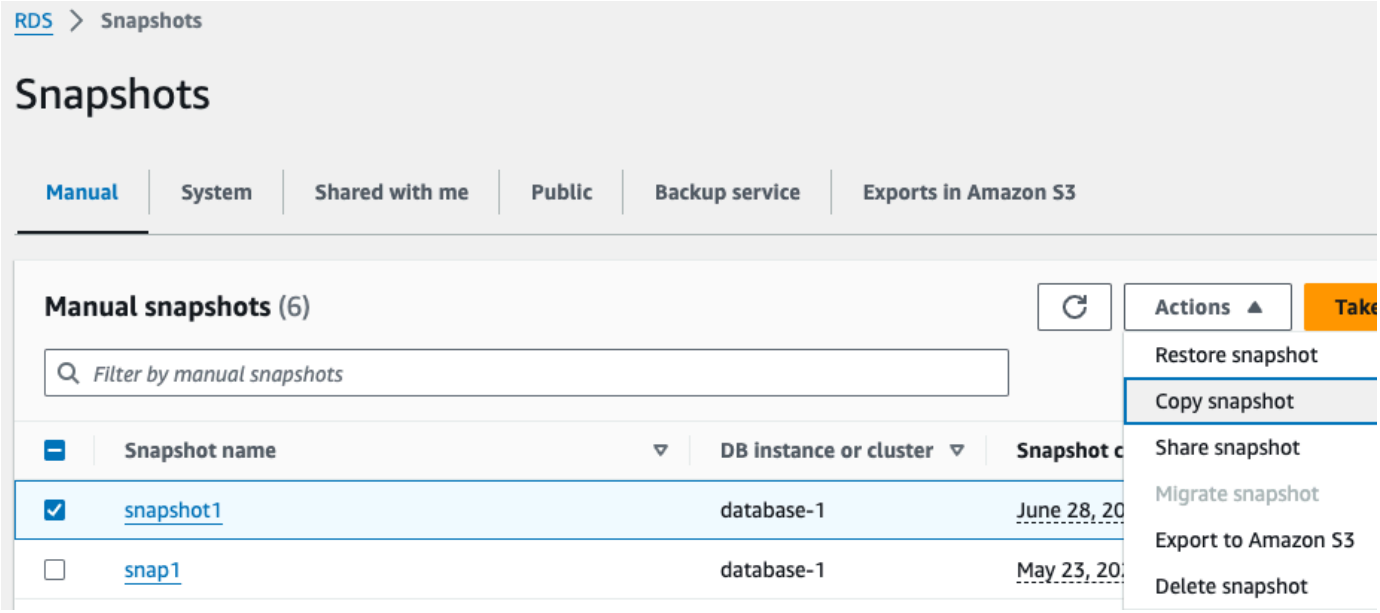
Cette procédure permet de copier des instantanés de cluster de bases de données chiffrés ou non chiffrés dans la même Région AWS ou entre régions.

Pour annuler une opération de copie une fois qu’elle est en cours, supprimez l’instantané de cluster de bases de données cible pendant que cet instantané de cluster de bases de données a le statut copying (copie).

Avant de copier un instantané de cluster de bases de données, consultez Limitations et Considérations relatives à la copie d’instantanés.

Pour copier un instantané de cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Snapshots.
3. Sélectionnez l’instantané de cluster de bases de données que vous voulez copier.
4. Choisissez Actions, puis Copier un instantané.



5. (Facultatif) Pour copier l'instantané de cluster de bases de données dans une autre Région AWS, sélectionnez cette Région AWS dans Région de destination.
6. Saisissez le nom de la copie de l'instantané de cluster de bases de données dans Nouvel identifiant d'instantané de base de données.
7. Pour copier les balises et les valeurs de l'instantané vers la copie de cet instantané, choisissez Copy Tags (Copier les balises).
8. Choisissez Copier un instantané.

Limitations

Vous trouverez ci-dessous certaines limites qui s'appliquent lorsque vous copiez des instantanés :

- Vous ne pouvez pas copier un instantané depuis ou vers les sites suivants Régions AWS :
  - Chine (Pékin)
  - Chine (Ningxia)
- Vous pouvez copier un instantané entre AWS GovCloud (US-East) et AWS GovCloud (US-West). Toutefois, vous ne pouvez pas copier un instantané entre ces AWS GovCloud (US) régions et ces zones commerciales Régions AWS.
- Si vous supprimez un instantané source avant que l'instantané cible ne soit disponible, la copie d'instantané peut échouer. Vérifiez que l'instantané cible a le statut `AVAILABLE` avant de supprimer un instantané source.

- Vous pouvez avoir jusqu'à cinq demandes de copie d'instantanés en cours vers une même région de destination par compte.
- Lorsque vous demandez plusieurs copies d'instantanés pour la même instance de base de données source, elles sont mises en file d'attente en interne. Les copies demandées ultérieurement ne démarreront pas tant que les copies de l'instantané précédent ne seront pas terminées. Pour plus d'informations, voir Pourquoi la création de mon instantané EC2 AMI ou EBS est-elle lente ? dans le AWS Knowledge Center.
- En fonction du type Régions AWS concerné et de la quantité de données à copier, la réalisation d'une copie instantanée entre régions peut prendre des heures. Dans certains cas, il peut y avoir un grand nombre de demandes de copie d'instantanés d'une région à une autre à partir d'une région source donnée. Dans de tels cas, Amazon RDS peut mettre en file d'attente les nouvelles demandes de copie entre régions provenant de cette région source en attendant que certaines copies en cours se terminent. Aucune information d'avancement n'est affichée sur les demandes de copie quand elles sont en file d'attente. Les informations d'avancement sont affichées lorsque la copie commence.
- Aurora ne prend pas en charge les instantanés incrémentiels. Les copies d'instantanés de clusters de bases de données Aurora sont toujours stockées sous forme de copies complètes. Une copie complète d'un instantané conserve toutes les données et métadonnées requises pour restaurer le cluster de bases de données.

Considérations relatives à la copie d'instantanés

Voici quelques points à prendre en compte lors de la copie d'instantanés.

Rubriques

- Considérations relatives à la copie d'instantanés partagés
- Considérations relatives à la copie d'instantanés de clusters de bases de données
- Considérations relatives à la copie d'instantanés entre régions
- Considérations relatives aux groupes de paramètres

Considérations relatives à la copie d'instantanés partagés

Vous pouvez copier des instantanés qui vous ont été partagés par d'autres AWS comptes. Dans certains cas, vous pouvez copier un instantané chiffré qui a été partagé depuis un autre AWS

compte. Dans ces cas, vous devez avoir accès à celui AWS KMS key qui a été utilisé pour chiffrer l'instantané. Pour de plus amples informations, veuillez consulter Partage d’instantanés chiffrés.

Copie entre régions et entre comptes en une seule étape

Pour copier un instantané entre régions et entre comptes en une seule action, vous devez d'abord partager l'instantané avec le compte cible AWS . Si l'instantané est chiffré, vous devez également partager la AWS KMS clé avec le AWS compte cible. Si l'instantané est chiffré avec la AWS KMS clé par défaut, vous devez d'abord le copier pour le rechiffrer avec une clé gérée par le client avant de le partager avec le compte cible. Une fois le partage effectué, vous pouvez créer une copie sur ce compte (régional ou interrégional) à partir du compte cible.

Considérations relatives à la copie d’instantanés de clusters de bases de données

Vous pouvez copier un instantané qui a été chiffré à l’aide d’une clé KMS. Si vous copiez un instantané chiffré, la copie de l’instantané doit également être chiffrée. Si vous copiez un instantané chiffré à l'intérieur de Région AWS celui-ci, vous pouvez chiffrer la copie avec la même clé KMS que l'instantané d'origine. Ou vous pouvez spécifier une clé KMS différente.

Si vous copiez un instantané chiffré entre régions, vous devez spécifier une clé KMS valide dans la Région AWS de destination. Il peut s’agir d’une clé KMS spécifique à une Région ou d’une clé multi-Régions. Pour plus d’informations sur les clés KMS multi-régions, consultez Utilisation de clés multi-régions dans AWS KMS.

Pour plus d'informations sur la gestion des AWS KMS clés pour Amazon RDS, consultezGestion AWS KMS key.

L’instantané source reste chiffré pendant tout le processus de copie. Pour plus d’informations, consultez Limitations des clusters de bases de données chiffrées Amazon Aurora.

> **Note**
>
> Pour les instantanés de cluster de bases de données Amazon Aurora, vous ne pouvez pas chiffrer un instantané de cluster de bases de données non chiffré lorsque vous copiez l’instantané.

Pour copier des instantanés de clusters de bases de données chiffrés, consultez les rubriques suivantes.

- Copie d’un instantané de cluster de bases de données chiffré à l’aide de l’AWS CLI ou de l’API Amazon RDS
- Copie d’un instantané de cluster de bases de données entre des comptes

Considérations relatives à la copie d’instantanés entre régions

Vous pouvez copier les instantanés de cluster de bases de données entre Régions AWS. Toutefois, il existe certaines contraintes et considérations en ce qui concerne la copie d’instantanés entre régions.

En fonction du type Régions AWS concerné et de la quantité de données à copier, la réalisation d'une copie instantanée entre régions peut prendre des heures.

Dans certains cas, il peut y avoir un grand nombre de demandes de copie d’instantanés d’une région à une autre à partir d’une Région AWS source donnée. Dans de tels cas, Amazon RDS peut placer les nouvelles demandes de copie interrégionales provenant de cette source Région AWS dans une file d'attente jusqu'à ce que certaines copies en cours soient terminées. Aucune information d’avancement n’est affichée sur les demandes de copie quand elles sont en file d’attente. Les informations d’avancement sont affichées lorsque la copie commence.

Des frais de transfert de données s'appliquent pour les copies instantanées interrégionales. La copie d’instantanés entre régions crée des copies complètes des données cibles, mais les frais de transfert de données sont incrémentiels. Les données incrémentielles incluent à la fois les nouvelles données ajoutées à la base de données d’un client depuis la dernière copie, ainsi que les modifications apportées aux données existantes. Pour plus d’informations, consultez Création de copies de sauvegarde entre Régions AWS dans le Guide du développeur AWS Backup .

> **Note**
>
> Aurora copie le minimum de données requis pour créer une copie complète d’un instantané dans la région de destination. Des frais de transfert de données s’appliquent lors de la copie d’instantanés entre régions.

Considérations relatives aux groupes de paramètres

Lorsque vous copiez un instantané d’une région à une autre, la copie n’inclut pas le groupe de paramètres utilisé par le cluster de bases de données d’origine. Lorsque vous restaurez un instantané pour créer un nouveau cluster de base de données, ce cluster de base de données obtient le groupe de paramètres par défaut pour celui dans Région AWS lequel il a été créé. Pour fournir

au nouveau cluster de bases de données les mêmes paramètres que la version d'origine, procédez comme suit :

1. Dans la destination Région AWS, créez un groupe de paramètres de cluster de base de données avec les mêmes paramètres que le cluster de base de données d'origine. S'il en existe déjà un dans le nouveau Région AWS, vous pouvez l'utiliser.
2. Après avoir restauré le cliché dans la destination Région AWS, modifiez le nouveau cluster de base de données et ajoutez le groupe de paramètres nouveau ou existant de l'étape précédente.

Copie d'un instantané de cluster de bases de données non chiffré à l'aide de l'AWS CLI ou de l'API Amazon RDS

Utilisez les procédures décrites dans les sections suivantes pour copier un instantané de cluster de bases de données non chiffré à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API Amazon RDS.

Pour annuler une opération de copie une fois qu'elle est en cours, supprimez l'instantané de cluster de bases de données cible identifiée par `--target-db-cluster-snapshot-identifier` ou `TargetDBClusterSnapshotIdentifier` tandis que le statut de l'instantané de cluster de bases de données dispose du statut copying (copie en cours).

Console

Pour copier un instantané de cluster de bases de données à l'aide de la AWS Management Console, consultez Copie d'un instantané de cluster de bases de données à l'aide de la AWS Management Console.

AWS CLI

Pour copier un instantané de cluster de bases de données, utilisez la commande AWS CLI copy-db-cluster-snapshot. Si vous copiez l'instantané dans une autre Région AWS, exécutez la commande dans la Région AWS dans laquelle l'instantané sera copié.

Les options suivantes sont utilisées pour copier un instantané de cluster de bases de données non chiffré :

- `--source-db-cluster-snapshot-identifier` – Identifiant de l'instantané de cluster de bases de données à copier. Si vous copiez l'instantané vers une autre Région AWS, cet identifiant doit être indiqué au format ARN de la Région AWS source.

- `--target-db-cluster-snapshot-identifier` – Identifiant de la nouvelle copie de l'instantané de cluster de bases de données.

Le code suivant crée une copie d'instantané de cluster de bases de données `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` appelée `myclustersnapshotcopy` dans la Région AWS dans laquelle la commande est exécutée. Lorsque la copie est réalisée, toutes les balises de l'instantané d'origine sont copiées dans la copie de l'instantané.

Example

Pour Linux, macOS ou Unix :

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --copy-tags
```

Pour Windows :

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --copy-tags
```

API RDS

Pour copier un instantané de cluster de bases de données, utilisez l'opération d'API Amazon RDS CopyDBClusterSnapshot. Si vous copiez l'instantané dans une autre Région AWS, exécutez l'action dans la Région AWS dans laquelle l'instantané sera copié.

Les paramètres suivants sont utilisés pour copier un instantané de cluster de bases de données non chiffré :

- `SourceDBClusterSnapshotIdentifier` – Identifiant de l'instantané de cluster de bases de données à copier. Si vous copiez l'instantané vers une autre Région AWS, cet identifiant doit être indiqué au format ARN de la Région AWS source.

- `TargetDBClusterSnapshotIdentifier` – Identifiant de la nouvelle copie de l'instantané de cluster de bases de données.

Le code suivant crée une copie d'un instantané `arn:aws:rds:us-east-1:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20130805` appelée `myclustersnapshotcopy` dans la région USA Ouest (Californie du Nord). Lorsque la copie est réalisée, toutes les balises de l'instantané d'origine sont copiées dans la copie de l'instantané.

Example

```
https://rds.us-west-1.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-east-1%3A123456789012%3Acluster-snapshot%3Aaurora-cluster1-snapshot-20130805
   &TargetDBSnapshotIdentifier=myclustersnapshotcopy
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20140429/us-west-1/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
   &X-Amz-Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copie d'un instantané de cluster de bases de données chiffré à l'aide de l'AWS CLI ou de l'API Amazon RDS

Utilisez les procédures décrites dans les sections suivantes pour copier un instantané de cluster de bases de données chiffré à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API Amazon RDS.

Pour annuler une opération de copie une fois qu'elle est en cours, supprimez l'instantané de cluster de bases de données cible identifiée par `--target-db-cluster-snapshot-identifier` ou `TargetDBClusterSnapshotIdentifier` tandis que le statut de l'instantané de cluster de bases de données dispose du statut copying (copie en cours).

Console

Pour copier un instantané de cluster de bases de données à l'aide de la AWS Management Console, consultez Copie d'un instantané de cluster de bases de données à l'aide de la AWS Management Console.

AWS CLI

Pour copier un instantané de cluster de bases de données, utilisez la commande AWS CLI copy-db-cluster-snapshot. Si vous copiez l'instantané dans une autre Région AWS, exécutez la commande dans la Région AWS dans laquelle l'instantané sera copié.

Les options suivantes sont utilisées pour copier un instantané de cluster de bases de données chiffré :

- `--source-db-cluster-snapshot-identifier` – Identifiant de l'instantané de cluster de bases de données chiffré à copier. Si vous copiez l'instantané vers une autre Région AWS, cet identifiant doit être indiqué au format ARN de la Région AWS source.
- `--target-db-cluster-snapshot-identifier` – Identifiant de la nouvelle copie de l'instantané de cluster de bases de données chiffré.
- `--kms-key-id` – Identifiant de clé KMS de la clé à utiliser pour chiffrer la copie de l'instantané de cluster de bases de données.

  Vous pouvez également utiliser cette option si l'instantané de cluster de bases de données est chiffré, que vous copiez l'instantané dans la même Région AWS et que vous voulez spécifier une nouvelle clé KMS pour chiffrer la copie. Sinon, la copie de l'instantané de cluster de bases de données est chiffrée avec la même clé KMS que l'instantané de cluster de bases de données source.

  Vous devez utiliser cette option si l'instantané de cluster de bases de données est chiffré et que vous copiez l'instantané dans une autre Région AWS. Dans ce cas, vous devez indiquer une clé KMS pour la Région AWS de destination.

L'exemple de code suivant copie l'instantané de cluster de bases de données chiffré de la région USA Ouest (Oregon) vers la région USA Est (Virginie du Nord). La commande est appelée dans la région USA Est (Virginie du Nord).

Example

Pour Linux, macOS ou Unix :

```
aws rds copy-db-cluster-snapshot \
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 \
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy \
  --kms-key-id my-us-east-1-key
```

Pour Windows :

```
aws rds copy-db-cluster-snapshot ^
  --source-db-cluster-snapshot-identifier arn:aws:rds:us-west-2:123456789012:cluster-snapshot:aurora-cluster1-snapshot-20161115 ^
  --target-db-cluster-snapshot-identifier myclustersnapshotcopy ^
  --kms-key-id my-us-east-1-key
```

Le paramètre `--source-region` est obligatoire lorsque vous copiez un instantané de cluster de bases de données chiffré entre la région AWS GovCloud (US-East) et la région AWS GovCloud (US-West). Pour `--source-region`, spécifiez la Région AWS de l'instance de base de données source. La Région AWS spécifiée dans `source-db-cluster-snapshot-identifier` doit correspondre à la Région AWS spécifiée dans `--source-region`.

Si `--source-region` n'est pas spécifié, spécifiez une valeur `--pre-signed-url`. Une URL présignée est une URL qui contient une demande signée via Signature Version 4 pour la commande `copy-db-cluster-snapshot` qui est appelée dans la Région AWS source. Pour en savoir plus sur l'option `pre-signed-url`, consultez copy-db-cluster-snapshot dans la Référence des commandes AWS CLI.

API RDS

Pour copier un instantané de cluster de bases de données, utilisez l'opération d'API Amazon RDS CopyDBClusterSnapshot. Si vous copiez l'instantané dans une autre Région AWS, exécutez l'action dans la Région AWS dans laquelle l'instantané sera copié.

Les paramètres suivants sont utilisés pour copier un instantané de cluster de bases de données chiffré :

- `SourceDBClusterSnapshotIdentifier` – Identifiant de l'instantané de cluster de bases de données chiffré à copier. Si vous copiez l'instantané vers une autre Région AWS, cet identifiant doit être indiqué au format ARN de la Région AWS source.
- `TargetDBClusterSnapshotIdentifier` – Identifiant de la nouvelle copie de l'instantané de cluster de bases de données chiffré.

- `KmsKeyId` – Identifiant de clé KMS de la clé à utiliser pour chiffrer la copie de l'instantané de cluster de bases de données.

  Vous pouvez utiliser ce paramètre si l'instantané de cluster de bases de données est chiffré, que vous copiez l'instantané dans la même Région AWS et que vous spécifiez une nouvelle clé KMS à utiliser pour chiffrer la copie. Sinon, la copie de l'instantané de cluster de bases de données est chiffrée avec la même clé KMS que l'instantané de cluster de bases de données source.

  Vous devez utiliser ce paramètre si l'instantané de cluster de bases de données est chiffré et que vous copiez l'instantané dans une autre Région AWS. Dans ce cas, vous devez indiquer une clé KMS pour la Région AWS de destination.
- `PreSignedUrl` – Si vous copiez l'instantané dans une autre Région AWS, vous devez spécifier le paramètre `PreSignedUrl`. La valeur `PreSignedUrl` doit être une URL qui contient une demande signée via Signature Version 4 pour appeler l'action `CopyDBClusterSnapshot` dans la Région AWS source à partir de laquelle l'instantané de cluster de bases de données est copié. Pour en savoir plus sur l'utilisation d'une URL présignée, consultez CopyDBClusterSnapshot.

L'exemple de code suivant copie l'instantané de cluster de bases de données chiffré de la région USA Ouest (Oregon) vers la région USA Est (Virginie du Nord). L'action est appelée dans la région USA Est (Virginie du Nord).

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CopyDBClusterSnapshot
    &KmsKeyId=my-us-east-1-key
    &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCopyDBClusterSnapshot
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526SourceDBClusterSnapshotIdentifier%253Darn%25253Aaws%25253Ards
%25253Aus-west-2%25253A123456789012%25253Acluster-snapshot%25253Aaurora-cluster1-
snapshot-20161115
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-west-2%252Frds
%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
```

```
        %2526X-Amz-Expires%253D3600
        %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-amz-
content-sha256%253Bx-amz-date
        %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SourceDBClusterSnapshotIdentifier=arn%3Aaws%3Ards%3Aus-
west-2%3A123456789012%3Acluster-snapshot%3Aaurora-cluster1-snapshot-20161115
    &TargetDBClusterSnapshotIdentifier=myclustersnapshotcopy
    &Version=2014-10-31
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
    &X-Amz-Date=20161117T221704Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=da4f2da66739d2e722c85fcfd225dc27bba7e2b8dbea8d8612434378e52adccf
```

Le paramètre `PreSignedUrl` est obligatoire lorsque vous copiez un instantané de cluster de bases de données chiffré entre la région AWS GovCloud (US-East) et la région AWS GovCloud (US-West). La valeur `PreSignedUrl` doit être une URL qui contient une demande signée via Signature Version 4 pour appeler l'opération `CopyDBClusterSnapshot` dans la Région AWS source à partir de laquelle l'instantané de cluster de bases de données est copié. Pour en savoir plus sur l'utilisation d'une URL présignée, consultez CopyDBClusterSnapshot dans la Référence d'API Amazon RDS.

Pour générer automatiquement plutôt que manuellement une URL présignée, utilisez plutôt la commande AWS CLI copy-db-cluster-snapshot avec l'option `--source-region`.

Copie d'un instantané de cluster de bases de données entre des comptes

Vous pouvez activer d'autres comptes AWS pour copier les instantanés de cluster de bases de données que vous avez spécifiés à l'aide des actions `ModifyDBClusterSnapshotAttribute` et `CopyDBClusterSnapshot` de l'API Amazon RDS. Vous pouvez uniquement copier des instantanés de cluster de bases de données entre les comptes d'une même Région AWS. Le processus de copie entre comptes fonctionne de la façon suivante : le compte A configure l'instantané pour qu'il puisse être copié, tandis que le compte B le copie.

1. A l'aide du compte A, appelez `ModifyDBClusterSnapshotAttribute`, en spécifiant **`restore`** pour le paramètre `AttributeName`, ainsi que l'ID du compte B du paramètre `ValuesToAdd`.

2. (Si l'instantané est chiffré) A l'aide du compte A, mettez à jour la politique de clé pour la clé KMS, en ajoutant d'abord l'ARN du compte B comme un `Principal`, puis autorisez l'action `kms:CreateGrant`.
3. (Si l'instantané est chiffré) A l'aide du compte B, choisissez ou créez un utilisateur et attachez une politique IAM à cet utilisateur afin de permettre à ce dernier de copier un instantané de cluster de bases de données chiffré à l'aide de votre clé KMS.
4. A l'aide du compte B, appelez la commande `CopyDBClusterSnapshot` et utilisez le paramètre `SourceDBClusterSnapshotIdentifier` pour spécifier l'ARN de l'instantané de cluster de bases de données à copier, qui doit comprendre l'ID du compte A.

Pour répertorier tous les comptes AWS autorisés à restaurer un instantané de cluster de bases de données, utilisez l'opération d'API DescribeDBSnapshotAttributes ou DescribeDBClusterSnapshotAttributes.

Pour supprimer l'autorisation de partage d'un compte AWS, utilisez l'action `ModifyDBSnapshotAttribute` ou `ModifyDBClusterSnapshotAttribute` avec `AttributeName` configuré sur `restore` et l'ID du compte à supprimer dans le paramètre `ValuesToRemove`.

Copie d'un instantané de cluster de bases de données non chiffré dans un autre compte

Utilisez la procédure suivante pour copier un instantané de cluster de bases de données non chiffré dans un autre compte dans la même Région AWS.

1. Dans le compte source de l'instantané de cluster de bases de données, appelez `ModifyDBClusterSnapshotAttribute`, en spécifiant **`restore`** pour le paramètre `AttributeName`, ainsi que l'ID du compte cible du paramètre `ValuesToAdd`.

   Exécuter l'exemple suivant à l'aide du compte `987654321` permet d'autoriser deux identifiants de compte AWS `123451234512` et `123456789012` à restaurer l'instantané de cluster de bases de données appelé `manual-snapshot1`.

   ```
   https://rds.us-west-2.amazonaws.com/
    ?Action=ModifyDBClusterSnapshotAttribute
    &AttributeName=restore
    &DBClusterSnapshotIdentifier=manual-snapshot1
    &SignatureMethod=HmacSHA256&SignatureVersion=4
    &ValuesToAdd.member.1=123451234512
    &ValuesToAdd.member.2=123456789012
   ```

```
 &Version=2014-10-31
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
 &X-Amz-Date=20150922T220515Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

2. Dans le compte cible, appelez la commande `CopyDBClusterSnapshot` et utilisez le paramètre `SourceDBClusterSnapshotIdentifier` pour spécifier l'ARN de l'instantané de cluster de bases de données à copier, qui doit comprendre l'ID du compte source.

   Exécuter l'exemple suivant à l'aide du compte `123451234512` permet de copier l'instantané de cluster de bases de données `aurora-cluster1-snapshot-20130805` à partir du compte `987654321` et de créer un instantané de cluster de bases de données appelé `dbclustersnapshot1`.

```
https://rds.us-west-2.amazonaws.com/
   ?Action=CopyDBClusterSnapshot
   &CopyTags=true
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
   &X-Amz-
Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Copie d'un instantané de cluster de bases de données chiffré dans un autre compte

Utilisez la procédure suivante pour copier un instantané de cluster de bases de données chiffré dans un autre compte dans la même Région AWS.

1. Dans le compte source de l'instantané de cluster de bases de données, appelez `ModifyDBClusterSnapshotAttribute`, en spécifiant **`restore`** pour le paramètre `AttributeName`, ainsi que l'ID du compte cible du paramètre `ValuesToAdd`.

Exécuter l'exemple suivant à l'aide du compte `987654321` permet d'autoriser deux identifiants de compte AWS `123451234512` et `123456789012` à restaurer l'instantané de cluster de bases de données appelé `manual-snapshot1`.

```
https://rds.us-west-2.amazonaws.com/
 ?Action=ModifyDBClusterSnapshotAttribute
 &AttributeName=restore
 &DBClusterSnapshotIdentifier=manual-snapshot1
 &SignatureMethod=HmacSHA256&SignatureVersion=4
 &ValuesToAdd.member.1=123451234512
 &ValuesToAdd.member.2=123456789012
 &Version=2014-10-31
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
 &X-Amz-Date=20150922T220515Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=ef38f1ce3dab4e1dbf113d8d2a265c67d17ece1999ffd36be85714ed36dddbb3
```

2. Dans le compte source de l'instantané de cluster de bases de données, créez une clé KMS personnalisée dans la même Région AWS que l'instantané de cluster de bases de données crypté. Lors de la création de la clé gérée par le client, vous donnez accès à celle-ci au Compte AWS cible. Pour plus d'informations, consultez Création d'une clé gérée par le client et octroi d'un accès à celle-ci..
3. Copiez et partagez l'instantané avec le Compte AWS cible. Pour plus d'informations, consultez Copie et partage d'instantané depuis le compte source.
4. Dans le compte cible, appelez la commande `CopyDBClusterSnapshot` et utilisez le paramètre `SourceDBClusterSnapshotIdentifier` pour spécifier l'ARN de l'instantané de cluster de bases de données à copier, qui doit comprendre l'ID du compte source.

   Exécuter l'exemple suivant à l'aide du compte `123451234512` permet de copier l'instantané de cluster de bases de données `aurora-cluster1-snapshot-20130805` à partir du compte `987654321` et de créer un instantané de cluster de bases de données appelé `dbclustersnapshot1`.

   ```
   https://rds.us-west-2.amazonaws.com/
      ?Action=CopyDBClusterSnapshot
      &CopyTags=true
      &SignatureMethod=HmacSHA256
      &SignatureVersion=4
   ```

```
   &SourceDBClusterSnapshotIdentifier=arn:aws:rds:us-west-2:987654321:cluster-
snapshot:aurora-cluster1-snapshot-20130805
   &TargetDBClusterSnapshotIdentifier=dbclustersnapshot1
   &Version=2013-09-09
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Credential=AKIADQKE4SARGYLE/20150922/us-west-2/rds/aws4_request
   &X-Amz-Date=20140429T175351Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-
date
   &X-Amz-
Signature=9164337efa99caf850e874a1cb7ef62f3cea29d0b448b9e0e7c53b288ddffed2
```

Partage d’un instantané de cluster de bases de données

Amazon RDS vous permet de partager un instantané de cluster de bases de données manuel des façons suivantes :

- Le partage d’un instantané de cluster de bases de données manuel chiffré ou non chiffré permet aux comptes AWS autorisés de copier l’instantané.
- Le partage d’un instantané de cluster de bases de données manuel chiffré ou non chiffré permet aux comptes AWS autorisés de restaurer directement un cluster de bases de données à partir l’instantané plutôt que d’effectuer une copie et de la restaurer.

> **Note**
>
> Pour partager un instantané de cluster de bases de données automatisé, créez un instantané de cluster de bases de données manuel en copiant l’instantané automatisé, puis partagez cette copie. Ce processus s’applique également aux ressources générées par AWS Backup.

Pour plus d’informations sur la copie d’un instantané, consultez Copie d’un instantané de cluster de bases de données. Pour plus d’informations sur la restauration d’une instance de base de données à partir d’un instantané de cluster de bases de données, consultez Restauration à partir d’un instantané de cluster de bases de données.

Pour plus d’informations sur la restauration d’un cluster de bases de données à partir d’un instantané de cluster de bases de données, consultez Présentation de la sauvegarde et de la restauration d’un cluster de bases de données Aurora.

Vous pouvez partager un instantané manuel avec jusqu’à 20 autres Comptes AWS.

Les limites suivantes s’appliquent lors du partage d’instantanés manuels avec d’autres Comptes AWS :

- Lorsque vous restaurez un cluster de bases de données à partir d’un instantané partagé à l’aide d’AWS Command Line Interface (AWS CLI) ou de l’API Amazon RDS, vous devez spécifier l’Amazon Resource Name (ARN) de l’instantané partagé en tant qu’identifiant de l’instantané.

Découvrez comment partager des instantanés, des instantanés publics et des instantanés chiffrés dans les sections suivantes. Vous apprendrez également à arrêter de partager des instantanés.

Rubriques

Partage d’un instantané

Vous pouvez partager un instantané de cluster de bases de données à l’aide d’AWS Management Console, d’AWS CLI ou de l’API RDS.

Console

En utilisant la console Amazon RDS, vous pouvez partager un instantané de cluster de bases de données manuel avec jusqu’à 20 Comptes AWS. Vous pouvez également utiliser la console pour arrêter le partage d’un instantané manuel avec un ou plusieurs comptes.

Pour partager un instantané de cluster de bases de données manuel à l’aide de la console Amazon RDS

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Snapshots.
3. Sélectionnez l’instantané manuel que vous voulez partager.
4. Pour Actions, choisissez Share snapshot (Partager l’instantané).
5. Choisissez l’une des options suivantes pour DB snapshot visibility (Visibilité d’instantané de base de données).

    - Si la source n’est pas chiffrée, choisissez Public pour permettre à tous les Comptes AWS de restaurer un cluster de bases de données à partir de votre instantané de cluster de bases de données manuel, ou choisissez Privé pour autoriser uniquement les Comptes AWS que vous spécifiez à restaurer un cluster de bases de données à partir de votre instantané de cluster de bases de données manuel.

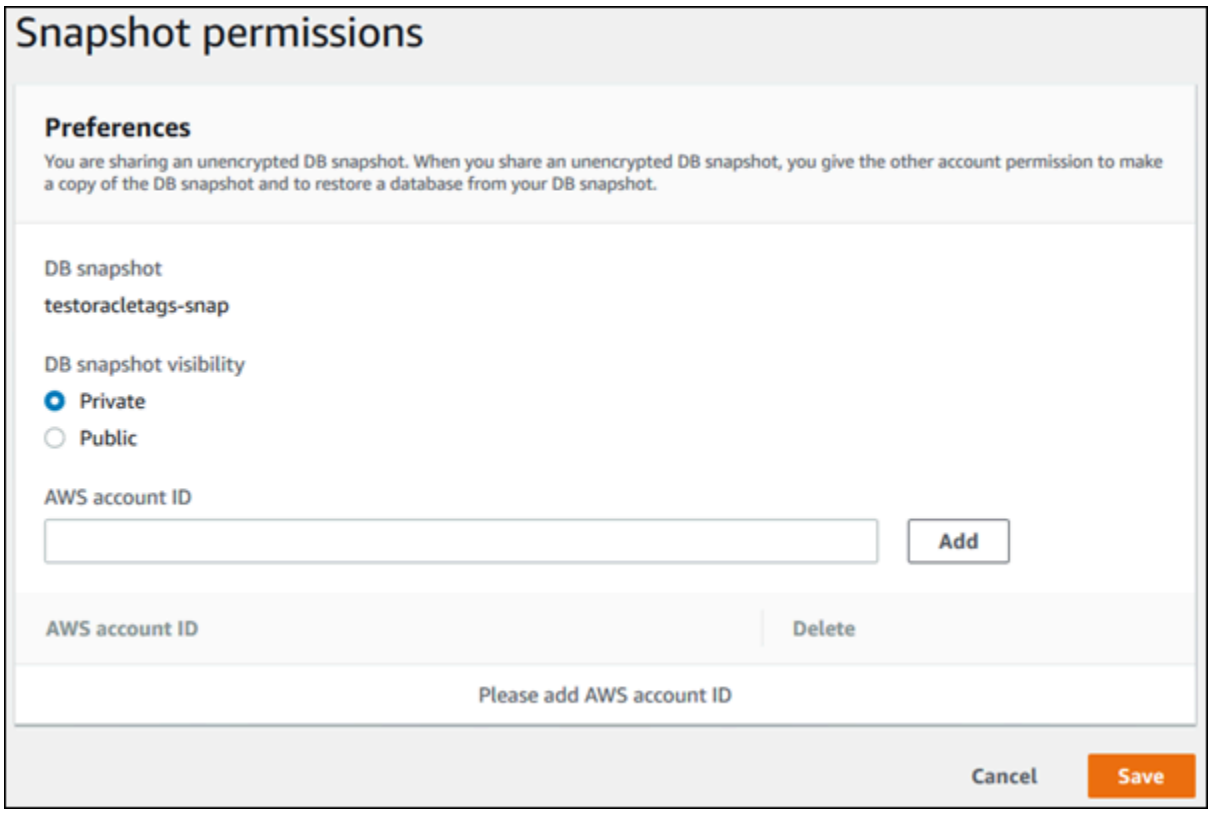
> **Warning**
>
> Si vous définissez Visibilité d'instantané de bases de données sur Public, tous les Comptes AWS pourront restaurer un cluster de bases de données à partir de votre instantané de cluster de bases de données manuel, et pourront accéder à vos données. Ne partagez pas en Public un instantané manuel de cluster DB contenant des informations privées.
> Pour plus d'informations, consultez Partage d'instantanés publics.

- Si la source est chiffrée, la Visibilité d'instantané de base de données est définie sur Privé, car les instantanés chiffrés ne peuvent pas être partagés s'ils sont marqués comme étant publics.

> **Note**
>
> Les instantanés chiffrés avec la AWS KMS key par défaut ne peuvent pas être partagés. Pour en savoir plus sur la manière de contourner ce problème, consultez Partage d'instantanés chiffrés.

6. Dans ID de compte AWS, saisissez l'identifiant du Compte AWS que vous souhaitez autoriser à restaurer un cluster de bases de données à partir de votre instantané manuel, puis choisissez Ajouter. Faites de même pour inclure des identifiants de Compte AWS supplémentaires, jusqu'à 20 Comptes AWS.

   Si vous faites une erreur en ajoutant un identifiant de Compte AWS à la liste des comptes autorisés, vous pouvez le supprimer de la liste en choisissant Supprimer à droite de l'identifiant de Compte AWS incorrect.



7. Après avoir ajouté des identifiants pour tous les Comptes AWS que vous souhaitez autoriser à restaurer l'instantané manuel, choisissez Enregistrer pour enregistrer vos modifications.

AWS CLI

Pour partager un instantané de cluster de bases de données, utilisez la commande `aws rds modify-db-cluster-snapshot-attribute`. Utilisez le paramètre `--values-to-add` pour ajouter une liste des identifiants de Comptes AWS autorisés à restaurer l'instantané manuel.

Example de partager un instantané avec un seul compte

L'exemple suivant active un identifiant de Compte AWS `123456789012` pour restaurer l'instantané de cluster de bases de données nommé `cluster-3-snapshot`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier cluster-3-snapshot \
--attribute-name restore \
--values-to-add 123456789012
```

Pour Windows :

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier cluster-3-snapshot ^
--attribute-name restore ^
--values-to-add 123456789012
```

Example de partager un instantané avec plusieurs comptes

L'exemple suivant active deux identifiants de Compte AWS, `111122223333` et `444455556666`, pour restaurer l'instantané de cluster de bases de données appelé `manual-cluster-snapshot1`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier manual-cluster-snapshot1 \
--attribute-name restore \
--values-to-add {"111122223333","444455556666"}
```

Pour Windows :

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier manual-cluster-snapshot1 ^
--attribute-name restore ^
--values-to-add "[\"111122223333\",\"444455556666\"]"
```

> **Note**
>
> Lorsque vous utilisez l'invite de commandes Windows, vous devez utiliser des guillemets doubles (") d'échappement dans le code JSON en les préfixant d'une barre oblique inverse (\).

Pour répertorier les Comptes AWS activés pour restaurer un instantané, utilisez la commande `describe-db-cluster-snapshot-attributes` de l'AWS CLI.

API RDS

Vous pouvez également partager un instantané de cluster de bases de données manuel avec d'autres Comptes AWS à l'aide de l'API Amazon RDS. Pour ce faire, appelez l'opération

`ModifyDBClusterSnapshotAttribute`. Indiquez `restore` pour `AttributeName`, et utilisez le paramètre `ValuesToAdd` pour ajouter une liste des identifiants de Comptes AWS pour ceux autorisés à restaurer l'instantané manuel.

Pour qu'un instantané manuel soit défini comme étant public et qu'il puisse être restauré par tous les Comptes AWS, utilisez la valeur `all`. Toutefois, n'ajoutez pas la valeur `all` pour tous les instantanés manuels contenant des informations privées que vous ne souhaitez pas mettre à la disposition de tous les Comptes AWS. De même, ne spécifiez pas la valeur `all` pour les instantanés chiffrés, car il est impossible de rendre tous ces instantanés publics.

Pour répertorier tous les Comptes AWS autorisés à restaurer un instantané, utilisez l'opération d'API `DescribeDBClusterSnapshotAttributes`.

Partage d'instantanés publics

Vous pouvez partager un instantané manuel non chiffré en mode public pour qu'il soit accessible à tous les Comptes AWS. Lors du partage d'un instantané marqué comme public, assurez-vous de n'inclure aucune information privée dans l'instantané public.

Lorsqu'un instantané est partagé publiquement, il donne à tous les Comptes AWS l'autorisation de copier l'instantané et de créer des clusters de bases de données à partir de cet instantané.

Le stockage de sauvegarde des snapshots publics appartenant à d'autres comptes n'est pas facturé. Seuls les instantanés que vous possédez vous sont facturés.

Si vous copiez un instantané public, vous êtes propriétaire de la copie. Le stockage de sauvegarde de votre copie d'instantané vous est facturé. Si vous créez un cluster de bases de données à partir d'un instantané public, ce cluster de bases de données vous est facturé. Pour obtenir des informations sur la tarification Amazon Aurora, consultez la page de tarification Aurora.

Vous ne pouvez supprimer que les instantanés publics que vous possédez. Pour supprimer un instantané partagé ou public, assurez-vous de vous connecter au Compte AWS propriétaire de l'instantané.

Affichage des instantanés publics appartenant à d'autres Comptes AWS

Vous pouvez afficher les instantanés publics appartenant à d'autres comptes dans une Région AWS spécifique dans l'onglet Public de la page Instantanés dans la console Amazon RDS. Vos instantanés (ceux appartenant à votre compte) n'apparaissent pas dans cet onglet.

Pour afficher des instantanés publics

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Snapshots.
3. Choisissez l'onglet Public.

   Les instantanés publics s'affichent. Vous pouvez voir quel compte possède un instantané public dans la colonne Owner (Propriétaire).

   > **Note**
   >
   > Pour voir cette colonne, vous devrez peut-être modifier les préférences de la page en sélectionnant l'icône en forme d'engrenage en haut à droite de la liste Public snapshots (Instantanés publics).

Affichage de vos propres Instantanés publics

Vous pouvez utiliser la commande AWS CLI suivante (Unix uniquement) pour afficher les instantanés publics appartenant à votre Compte AWS dans une Région AWS particulière.

```
aws rds describe-db-cluster-snapshots --snapshot-type public --include-public |
 grep account_number
```

La sortie renvoyée est semblable à l'exemple suivant si vous avez des instantanés publics.

```
"DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:myclustersnapshot1",
"DBClusterSnapshotArn": "arn:aws:rds:us-west-2:123456789012:cluster-
snapshot:myclustersnapshot2",
```

Partage d'instantanés publics à partir de versions obsolètes du moteur de base de données

La restauration ou la copie d'instantanés publics à partir de versions obsolètes du moteur de base de données n'est pas prise en charge. Pour permettre la restauration ou la copie de votre instantané public non pris en charge, effectuez les opérations suivantes :

1. Marquez l'instantané comme privé.

2. Restaurez l'instantané.
3. Mettez à niveau le cluster de bases de données restauré vers une version du moteur prise en charge.
4. Créez un nouvel instantané.
5. Partagez à nouveau l'instantané publiquement.

Partage d'instantanés chiffrés

Vous pouvez partager des instantanés de cluster DB qui ont été chiffrés « au repos » en utilisant l'algorithme de chiffrement AES-256, comme décrit dans Chiffrement des ressources Amazon Aurora.

Les restrictions suivantes s'appliquent au partage d'instantanés chiffrés :

- Vous ne pouvez pas partager des instantanés chiffrés marqués comme publics.
- Vous ne pouvez pas partager un instantané chiffré à l'aide de la clé de chiffrement KMS par défaut du Compte AWS qui a partagé l'instantané.

  Pour plus d'informations sur la gestion des clés AWS KMS pour Amazon RDS, consultez Gestion AWS KMS key.

Pour contourner le problème de clé KMS par défaut, effectuez les tâches suivantes :

1. Création d'une clé gérée par le client et octroi d'un accès à celle-ci..
2. Copie et partage d'instantané depuis le compte source.
3. Copiez l'instantané partagé dans le compte cible.

Création d'une clé gérée par le client et octroi d'un accès à celle-ci.

Vous devez d'abord créer une clé KMS personnalisée dans la même Région AWS que le cluster de bases de données chiffré. Lors de la création de la clé gérée par le client, vous lui permettez d'accéder à un autre Compte AWS.

Création d'une clé gérée par le client et octroi d'un accès à celle-ci.

1. Connectez-vous à la AWS Management Console depuis le Compte AWS source.
2. Ouvrez la console AWS KMS à l'adresse https://console.aws.amazon.com/kms.

3. Pour changer de Région AWS, utilisez le sélecteur de région dans l’angle supérieur droit de la page.
4. Dans le panneau de navigation, choisissez Clés gérées par le client.
5. Choisissez Create key.
6. Sur la page Configurer la clé :
    a. Pour Type de clé, choisissez Symétrique.
    b. Pour Utilisation de la clé, choisissez Chiffrer et déchiffrer.
    c. (Facultatif) Développez Options avancées.
    d. Pour Origine des clés, choisissez KMS.
    e. Pour Régionalité, choisissez Clé de région unique.
    f. Choisissez Suivant.
7. Sur la page Ajouter des étiquettes :
    a. Pour Alias, entrez un nom d’affichage pour votre clé KMS, par exemple **`share-snapshot`**.
    b. (Facultatif) Saisissez une description pour votre clé KMS.
    c. (Facultatif) Ajoutez des identifications à votre règle KMS.
    d. Choisissez Suivant.
8. Sur la page Définir des autorisations d’administration de clé, choisissez Suivant.
9. Sur la page Définir des autorisations d’utilisation de clé :
    a. Pour Autre Comptes AWS, choisissez Ajouter une autre Compte AWS.
    b. Entrez l’ID du Compte AWS auquel vous souhaitez donner accès.

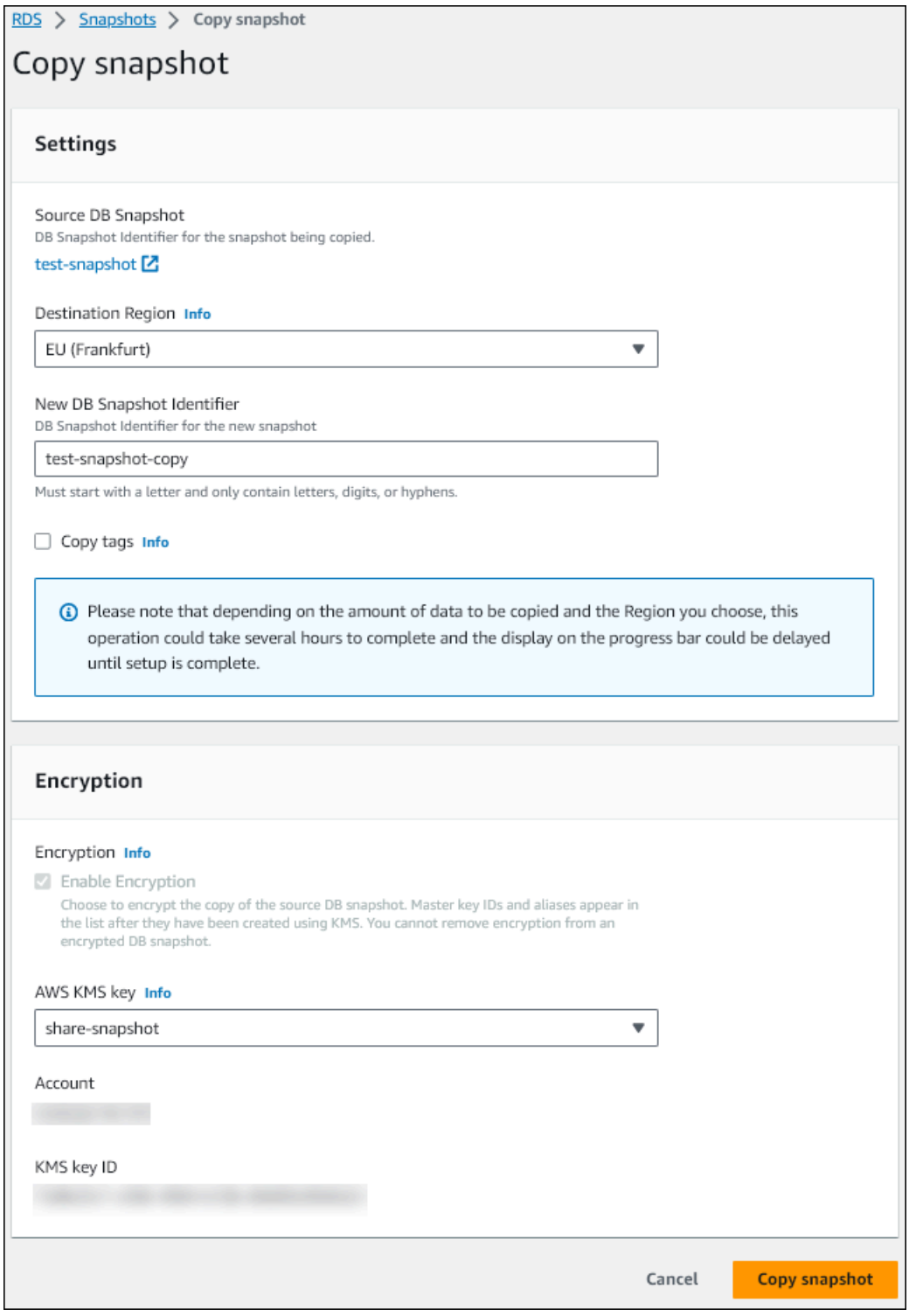
        Vous pouvez donner accès à plusieurs Comptes AWS.
    c. Choisissez Suivant.
10. Vérifiez votre clé KMS, puis choisissez Terminer.

Copie et partage d’instantané depuis le compte source

Ensuite, vous copiez l’instantané du cluster de bases de données source vers un nouvel instantané à l’aide de la clé gérée par le client. Ensuite, vous le partagez avec le Compte AWS cible.

Pour copier et partager l’instantané

1. Connectez-vous à la AWS Management Console depuis le Compte AWS source.
2. Ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
3. Dans le panneau de navigation, choisissez Snapshots.
4. Sélectionnez l’instantané de cluster de bases de données que vous voulez copier.
5. Sous Actions, choisissez Copier un instantané.
6. Sur la page Copier un instantané :
    a. Pour Région de destination, choisissez la Région AWS où vous avez créé la clé gérée par le client lors de la procédure précédente.
    b. Saisissez le nom de la copie de l’instantané de cluster de bases de données dans Nouvel identifiant d’instantané de base de données.
    c. Pour AWS KMS key, choisissez la clé gérée par le client que vous avez créée.



d. Choisissez Copy snapshot (Copier un instantané).

7. Lorsque la copie d'instantanée est disponible, sélectionnez-la.
8. Pour Actions, choisissez Share snapshot (Partager l'instantané).
9. Sur la page des Autorisations relatives aux instantanés :

a. Entrez l'ID du Compte AWS avec lequel vous partagez la copie instantanée, puis choisissez Ajouter.
b. Choisissez Enregistrer.

L'instantané est partagé.

Copiez l'instantané partagé dans le compte cible

Vous pouvez maintenant copier l'instantané partagé dans le Compte AWS cible.

Pour copier l'instantané partagé

1. Connectez-vous à la AWS Management Console depuis le Compte AWS cible.
2. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
3. Dans le panneau de navigation, choisissez Snapshots.
4. Choisissez l'onglet Partagé avec moi.
5. Sélectionnez l'instantané partagé.
6. Sous Actions, choisissez Copier un instantané.
7. Choisissez vos paramètres pour copier l'instantané comme dans la procédure précédente, mais utilisez une AWS KMS key appartenant au compte cible.

   Choisissez Copy snapshot (Copier un instantané).

Arrêt du partage d'un instantané

Pour arrêter de partager un instantané de cluster de bases de données, vous supprimez les autorisations depuis le Compte AWS cible.

Console

Pour arrêter le partage d'un instantané de cluster de bases de données manuel avec un Compte AWS

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Snapshots.

3. Sélectionnez l'instantané manuel que vous voulez cesser de partager.
4. Choisissez Actions, puis Share snapshot (Partager l'instantané).
5. Pour supprimer une autorisation pour un Compte AWS, choisissez Delete pour l'identifiant du compte AWS dans la liste des comptes autorisés.
6. Choisissez Save pour enregistrer les changements.

Interface de ligne de commande (CLI)

Pour supprimer de la liste un identifiant de Compte AWS, utilisez le paramètre `--values-to-remove`.

Example de l'arrêt du partage d'instantanés

L'exemple suivant empêche l'ID de Compte AWS 444455556666 de restaurer l'instantané.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-snapshot-attribute \
--db-cluster-snapshot-identifier manual-cluster-snapshot1 \
--attribute-name restore \
--values-to-remove 444455556666
```

Pour Windows :

```
aws rds modify-db-cluster-snapshot-attribute ^
--db-cluster-snapshot-identifier manual-cluster-snapshot1 ^
--attribute-name restore ^
--values-to-remove 444455556666
```

API RDS

Pour supprimer l'autorisation de partage d'un Compte AWS, utilisez l'opération `ModifyDBClusterSnapshotAttribute` avec `AttributeName` défini sur `restore` et le paramètre `ValuesToRemove`. Pour marquer un instantané manuel comme privé, supprimez la valeur `all` de la liste des valeurs pour l'attribut `restore`.

Exportation des données du cluster de bases de données vers Amazon S3

Vous pouvez exporter des données d'un cluster de bases de données Amazon Aurora en service vers un compartiment Amazon S3. Le processus d'exportation s'exécute en arrière-plan et n'affecte pas les performances de votre cluster de bases de données actif.

Par défaut, toutes les données du cluster de bases de données sont exportées. Toutefois, vous pouvez choisir d'exporter des ensembles spécifiques de bases de données, de schémas ou de tables.

Amazon Aurora clone le cluster de bases de données, extrait les données du clone et les stocke dans un compartiment Amazon S3. Les données sont stockées dans un format Apache Parquet qui est compressé et cohérent. En règle générale, la taille d'un fichier Parquet est comprise entre 1 et 10 Mo.

Les performances plus rapides que vous pouvez obtenir avec l'exportation de données d'instantanés pour les versions 2 et 3 de Aurora MySQL ne s'appliquent pas à l'exportation de données de cluster de bases de données. Pour plus d'informations, consultez Exportation de données d'instantanés de cluster de bases de données vers Amazon S3.

L'exportation de l'intégralité du cluster de bases de données vous est facturée, que vous exportiez des données complètes ou partielles. Pour en savoir plus, consultez la page Tarification d'Amazon Aurora.

Une fois les données exportées, vous pouvez les analyser directement via des outils tels que Amazon Athena ou Amazon Redshift Spectrum. Pour plus d'informations sur l'utilisation Athena de la lecture des données de parquet, consultez Parquet SerDe dans le Amazon Athena Guide de l'utilisateur. Pour plus d'informations sur l'utilisation de Redshift Spectrum pour lire des données Parquet, consultez COPY depuis les formats de données en colonnes dans le Guide du développeur de base de données Amazon Redshift.

La disponibilité et la prise en charge des fonctionnalités varient selon les versions spécifiques de chaque moteur de base de données, et selon les Régions AWS. Pour plus d'informations sur la disponibilité des versions et des régions de l'exportation des données de cluster de bases de données vers S3, consultez Régions et moteurs de bases de données Aurora pris en charge pour l'exportation des données du cluster vers Amazon S3.

Vous utilisez le processus suivant pour exporter des données d'un cluster de bases de données vers un compartiment Amazon S3. Pour plus de détails, consultez les sections suivantes.

Présentation de l'exportation des données depuis un cluster de bases de données

1. Identifiez le cluster de bases de données dont vous voulez exporter les données.
2. Configurez l'accès au compartiment Amazon S3.

   Un compartiment est un conteneur d'objets ou de fichiers Amazon S3. Pour fournir les informations permettant d'accéder à un compartiment, procédez comme suit :

   a. Identifiez le compartiment S3 où les données du cluster de bases de données doivent être exportées. Le compartiment S3 doit être situé dans la même région AWS que le cluster de bases de données. Pour plus d'informations, consultez Identification du compartiment Amazon S3 pour l'exportation.

   b. Créez un rôle Gestion des identités et des accès AWS (IAM) qui accorde à la tâche d'exportation du cluster de bases de données l'accès au compartiment S3. Pour plus d'informations, consultez Fournir l'accès à un compartiment Amazon S3 à l'aide d'un rôle IAM.
3. Créez un chiffrement symétrique AWS KMS key pour le chiffrement côté serveur. La clé KMS est utilisée par la tâche d'exportation du cluster pour configurer le chiffrement côté serveur AWS KMS lors de l'écriture des données d'exportation dans S3.

   La politique de clés KMS doit inclure à la fois les autorisations `kms:CreateGrant` et `kms:DescribeKey`. Pour plus d'informations sur l'utilisation des clés KMS dans Amazon Aurora, consultez Gestion AWS KMS key.

   Si vous avez une instruction de refus dans votre politique de clés KMS, veillez à exclure explicitement le principal de service AWS `export.rds.amazonaws.com`.

   Vous pouvez utiliser une clé KMS dans votre compte AWS ou une clé KMS entre comptes. Pour plus d'informations, consultez Utiliser un compte croisé AWS KMS key.
4. Exportez le cluster de bases de données vers Amazon S3 à l'aide de la console ou de la commande CLI `start-export-task`. Pour plus d'informations, consultez Création de tâches d'exportation du cluster de bases de données.
5. Pour accéder aux données exportées dans le compartiment Amazon S3, consultez Chargement, téléchargement et gestion d'objets dans le Guide de l'utilisateur Amazon Simple Storage Service.

Découvrez dans les sections suivantes comment configurer, exporter, surveiller, annuler et résoudre les problèmes liés aux tâches d'exportation de clusters de bases de données.

Rubriques

Considérations relatives aux exportations du cluster de bases de données

Pour en savoir plus sur les limites, les conventions de dénomination des fichiers, ainsi que sur la conversion et le stockage des données lors de l’exportation de clusters de bases de données vers Amazon S3, consultez les sections suivantes.

Rubriques

Limitations

L’exportation de données du cluster de bases de données vers Amazon S3 présente les limites suivantes :

- Vous ne pouvez pas exécuter simultanément plusieurs tâches d’exportation pour le même cluster de bases de données. Cette règle s’applique aux exportations complètes et partielles.
- Vous pouvez avoir jusqu’à cinq tâches d’exportation d’instantané de base de données simultanées en cours par Compte AWS.
- Les clusters de bases de données Aurora Serverless v1 ne prennent pas en charge les exportations vers S3.
- Aurora MySQL et Aurora PostgreSQL prennent en charge les exportations vers S3 uniquement pour le mode moteur provisionné.
- Les exportations vers S3 ne prennent pas en charge les préfixes S3 contenant le signe des deux-points (:).

- Les caractères suivants du chemin d'accès au fichier S3 sont convertis en traits de soulignement (_) lors de l'exportation :

```
\ ` " (space)
```

- Si une base de données, un schéma ou une table comporte des caractères autres que les suivants, l'exportation partielle n'est pas prise en charge. Toutefois, vous pouvez exporter l'ensemble du cluster de bases de données.
  - Lettres latines (A–Z)
  - Chiffres (0–9)
  - Symbole dollar ($)
  - Trait de soulignement (_)
- Les espaces ( ) et certains caractères ne sont pas pris en charge dans les noms de colonnes des tables de base de données. Les tables dont les noms de colonnes contiennent les caractères suivants sont ignorées lors de l'exportation :

```
, ; { } ( ) \n \t = (space)
```

- Les tables dont les noms contiennent des barres obliques (/) sont ignorées lors de l'exportation.
- Les tables temporaires et non journalisées d'Aurora PostgreSQL sont ignorées lors de l'exportation.
- Si les données contiennent un objet volumineux tel qu'un objet BLOB ou CLOB proche de ou supérieur à 500 Mo, l'exportation échoue.
- Si une table contient une grande ligne proche de ou supérieure à 2 Go, la table est ignorée lors de l'exportation.
- Pour les exportations partielles, la taille maximale de la liste `ExportOnly` est de 200 Ko.
- Nous vous recommandons vivement d'utiliser un nom unique pour chaque tâche d'exportation. Si vous n'utilisez pas un nom de tâche unique, vous risquez de recevoir le message d'erreur suivant :

  ExportTaskAlreadyExistsFault: An error occurred (ExportTaskAlreadyExists) when calling the StartExportTask operation: The export task with the ID *xxxxx* already exists (ExportTaskAlreadyExistsFault : une erreur s'est produite (ExportTaskAlreadyExists) lors de l'appel de l'opération StartExportTask : la tâche d'exportation avec l'ID xxxxx existe déjà).
- Certaines tables pouvant être ignorées, nous vous recommandons de vérifier les nombres de lignes et de tables dans les données après l'exportation.

Convention de dénomination de fichiers

Les données exportées pour des tables spécifiques sont stockées au format *base_prefix*/*files*, qui utilise le préfixe de base suivant :

```
export_identifier/database_name/schema_name.table_name/
```

Par exemple :

```
export-1234567890123-459/rdststcluster/mycluster.DataInsert_7ADB5D19965123A2/
```

Les fichiers de sortie utilisent la convention d'appellation suivante, où *partition_index* est alphanumérique :

```
partition_index/part-00000-random_uuid.format-based_extension
```

Par exemple :

```
1/part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
    a/part-00000-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
```

La convention de dénomination de fichiers est sujette à modification. Par conséquent, lors de la lecture des tables cibles, nous vous conseillons de lire tout ce qui se trouve à l'intérieur du préfixe de base de la table.

Conversion des données et format de stockage

Lorsque vous exportez un cluster de bases de données vers un compartiment Amazon S3, Amazon Aurora convertit les données, les exporte et les stocke au format Parquet. Pour plus d'informations, consultez Conversion des données lors de l'exportation vers un compartiment Amazon S3.

Configuration de l'accès à un compartiment Amazon S3

Vous identifiez le compartiment Amazon S3, puis vous donnez à la tâche d'exportation du cluster de bases de données l'autorisation d'y accéder.

Rubriques

Identification du compartiment Amazon S3 pour l’exportation

Identifiez le compartiment Amazon S3 vers lequel exporter les données du cluster de bases de données. Utilisez un compartiment S3 existant ou créez un nouveau compartiment S3.

> **Note**
>
> Le compartiment S3 doit se trouver dans la même AWS région que le cluster de base de données.

Pour plus d’informations sur l’utilisation des Amazon S3 compartiments, consultez les points suivants dans le Guide de l’utilisateur Amazon Simple Storage Service :

- Comment afficher les propriétés d’un compartiment S3 ?
- Comment activer le chiffrement par défaut pour un compartiment Amazon S3 ?
- Comment créer un compartiment S3 ?

Fournir l’accès à un compartiment Amazon S3 à l’aide d’un rôle IAM

Avant d’exporter les données du cluster de bases de données vers Amazon S3, donnez aux tâches d’exportation les autorisations d’accès en écriture au compartiment Amazon S3.

Pour accorder cette autorisation, créez une politique IAM qui donne accès au compartiment, puis créez un rôle IAM et attachez la politique au rôle. Plus tard, vous pourrez affecter le rôle IAM à la tâche d’exportation de votre cluster de bases de données.

> **Important**
>
> Si vous prévoyez d'utiliser le AWS Management Console pour exporter votre cluster de base de données, vous pouvez choisir de créer la politique IAM et le rôle automatiquement lorsque

vous exportez le cluster de base de données. Pour obtenir des instructions, veuillez consulter Création de tâches d'exportation du cluster de bases de données.

Pour donner aux tâches l'accès à Amazon S3

1. Créez une politique IAM. Cette politique fournit les autorisations relatives au compartiment et aux objets qui permettent à votre tâche d'exportation de cluster de données d'accéder à Amazon S3.

   Dans la politique, incluez les actions obligatoires suivantes pour permettre le transfert de fichiers depuis Amazon Aurora vers un compartiment S3 :

   - `s3:PutObject*`
   - `s3:GetObject*`
   - `s3:ListBucket`
   - `s3:DeleteObject*`
   - `s3:GetBucketLocation`

   Dans la politique, incluez les ressources suivantes pour identifier le compartiment S3 et les objets qu'il contient. La liste de ressources suivante indique le format Amazon Resource Name (ARN) pour l'accès à Amazon S3.

   - `arn:aws:s3:::amzn-s3-demo-bucket`
   - `arn:aws:s3:::amzn-s3-demo-bucket/*`

   Pour plus d'informations concernant la création d'une politique IAM pour Amazon Aurora, consultez Création et utilisation d'une politique IAM pour l'accès à une base de données IAM. Consultez également Didacticiel : création et attachement de votre première politique gérée par le client dans le Guide de l'utilisateur IAM.

   La AWS CLI commande suivante crée une politique IAM nommée `ExportPolicy` avec ces options. Elle accorde un accès à un compartiment nommé *amzn-s3-demo-bucket*.

> **Note**
>
> Après avoir créé la politique, notez son ARN. Vous en aurez besoin par la suite pour attacher la politique à un rôle IAM.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Créez un rôle IAM, afin qu'Aurora puisse endosser ce rôle IAM en votre nom pour accéder à vos compartiments Amazon S3. Pour plus d'informations, consultez Création d'un rôle pour déléguer des autorisations à un utilisateur IAM dans le Guide de l'utilisateur IAM.

   L'exemple suivant montre comment utiliser la AWS CLI commande pour créer un rôle nommé`rds-s3-export-role`.

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
```

```
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. Attachez la politique IAM que vous avez créée au rôle IAM que vous venez de créer.

   La AWS CLI commande suivante associe la politique créée précédemment au rôle nommé`rds-s3-export-role`. Remplacez *`your-policy-arn`* par l'ARN de stratégie que vous avez noté lors d'une étape précédente.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-export-role
```

Utilisation d'un compartiment Amazon S3 entre comptes

Vous pouvez utiliser des compartiments S3 sur plusieurs AWS comptes. Pour de plus amples informations, veuillez consulter Utilisation d'un compartiment Amazon S3 entre comptes.

Création de tâches d'exportation du cluster de bases de données

Créez des tâches d'exportation pour exporter les données de votre cluster de bases de données Aurora vers un compartiment Amazon S3. Vous pouvez avoir jusqu'à cinq tâches simultanées d'exportation de cluster de bases de données en cours par Compte AWS.

> **Note**
>
> L'exportation des données d'un cluster de bases de données peut prendre un certain temps selon le type et la taille de votre base de données. La tâche d'exportation commence par cloner et mettre à l'échelle l'ensemble de la base de données avant d'extraire les données vers Amazon S3. La progression de la tâche au cours de cette phase s'affiche sous l'intitulé Starting. Lorsque la tâche passe à l'exportation de données vers S3, la progression affiche l'intitulé En cours.
>
> La durée nécessaire à l'exportation dépend des données stockées dans la base de données. Par exemple, l'exportation des tables comportant des colonnes numériques d'index ou de clé primaire bien distribuées est la plus rapide. L'opération prend plus de temps pour les tables

qui ne contiennent pas de colonne adaptée au partitionnement et les tables avec un seul index sur une colonne basée sur une chaîne, car l'exportation utilise un processus à thread unique plus lent.

Vous pouvez exporter les données du cluster de bases de données vers Amazon S3 en utilisant la AWS Management Console, l'AWS CLI, ou l'API RDS.

Si vous utilisez une fonction Lambda pour exporter les données du cluster de bases de données, ajoutez l'action `kms:DescribeKey` à la politique de la fonction Lambda. Pour plus d'informations, consultez Autorisations AWS Lambda.

Console

L'option de console Export to Amazon S3 (Exporter vers Amazon S3) n'apparaît que pour les clusters de bases de données qui peuvent être exportés vers Amazon S3. Un cluster de bases de données peut ne pas être disponible pour l'exportation pour les raisons suivantes :

- Le moteur de base de données n'est pas pris en charge pour l'exportation S3.
- La version du cluster de bases de données n'est pas prise en charge pour l'exportation S3.
- L'exportation S3 n'est pas prise en charge dans la région AWS où le cluster de bases de données a été créé.

Pour exporter les données du cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Choisissez le cluster de bases de données dont vous voulez exporter les données.
4. Pour actions, choisissez Export to Amazon S3 (Exporter vers Amazon S3).

   La fenêtre Export to Amazon S3 (Exporter vers Amazon S3) apparaît.
5. Dans Export Identifier (Identifiant d'exportation), entrez un nom pour identifier la tâche d'exportation. Cette valeur est également utilisée pour le nom du fichier créé dans le compartiment S3.
6. Choisissez les données à exporter :

   - Choisissez All (Tout) pour exporter toutes les données du cluster de bases de données.

- Choisissez Partial (Partiel) pour exporter des parties spécifiques du cluster de bases de données. Pour identifier les parties du cluster à exporter, entrez un(e) ou plusieurs bases de données, schémas ou tables pour Identifiers (Identifiants), séparés par des espaces.

  Utilisez le format suivant :

  ```
  database[.schema][.table] database2[.schema2][.table2] ... databasen[.scheman]
  [.tablen]
  ```

  Exemples :

  ```
  mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
   mydatabase2.myschema2.mytable2
  ```

7. Pour S3 bucket (Compartiment S3), choisissez le compartiment vers lequel exporter.

   Pour affecter les données exportées à un chemin d'accès de dossier dans le compartiment S3, entrez le chemin d'accès facultatif pour S3 prefix (Préfixe S3).

8. Pour Rôle IAM, choisissez un rôle qui vous accorde un accès en écriture au compartiment S3 choisi, ou créez un nouveau rôle.

   - Si vous avez créé un rôle en suivant les étapes décrites dans Fournir l'accès à un compartiment Amazon S3 à l'aide d'un rôle IAM, choisissez ce rôle.
   - Si vous n'avez pas créé un rôle qui vous accorde un accès en écriture au compartiment S3 que vous avez choisi, choisissez Create a new role (Créer un nouveau rôle) pour créer le rôle automatiquement. Ensuite, saisissez un nom pour le rôle dans Nom du rôle IAM.

9. Pour KMS key (Clé KMS), entrez l'ARN de la clé à utiliser pour chiffrer les données exportées.
10. Choisissez Export to Amazon S3 (Exporter vers Amazon S3).

AWS CLI

Pour exporter les données d'un cluster de bases de données vers Amazon S3 à l'aide de AWS CLI, utilisez la commande start-export-task avec les options requises suivantes :

- `--export-task-identifier`
- `--source-arn` : Amazon Resource Name (ARN) du cluster de bases de données
- `--s3-bucket-name`

- `--iam-role-arn`
- `--kms-key-id`

Dans les exemples suivants, la tâche d'exportation est nommée *`my-cluster-export`*, qui exporte les données vers un compartiment S3 nommé *`amzn-s3-demo-destination-bucket`*.

Example

Pour Linux, macOS ou Unix :

```
aws rds start-export-task \
    --export-task-identifier my-cluster-export \
    --source-arn arn:aws:rds:us-west-2:123456789012:cluster:my-cluster \
    --s3-bucket-name amzn-s3-demo-destination-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

Pour Windows :

```
aws rds start-export-task ^
    --export-task-identifier my-DB-cluster-export ^
    --source-arn arn:aws:rds:us-west-2:123456789012:cluster:my-cluster ^
    --s3-bucket-name amzn-s3-demo-destination-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

Vous trouverez ci-après un exemple de sortie.

```
{
    "ExportTaskIdentifier": "my-cluster-export",
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:my-cluster",
    "S3Bucket": "amzn-s3-demo-destination-bucket",
    "IamRoleArn": "arn:aws:iam:123456789012:role/ExportTest",
    "KmsKeyId": "my-key",
    "Status": "STARTING",
    "PercentProgress": 0,
    "TotalExtractedDataInGB": 0,
}
```

Pour fournir un chemin de dossier dans le compartiment S3 pour l'exportation du cluster de bases de données, incluez l'option `--s3-prefix` dans la commande start-export-task.

API RDS

Pour exporter des données du cluster de bases de données vers Amazon S3 à l'aide de l'API Amazon RDS, utilisez l'opération StartExportTask avec les paramètres requis suivants :

- `ExportTaskIdentifier`
- `SourceArn` : ARN du cluster de bases de données
- `S3BucketName`
- `IamRoleArn`
- `KmsKeyId`

Surveillance des tâches d'exportation du cluster de bases de données

Vous pouvez surveiller les exportations du cluster de bases de données à l'aide de l'AWS Management Console, de l'AWS CLI ou de l'API RDS.

Console

Pour surveiller les exportations du cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Exports in Amazon S3 (Exportations dans Amazon S3).

   Les exportations de cluster de bases de données sont indiquées dans la colonne Source type (Type de source). L'état de l'exportation est affiché dans la colonne Status (État).
3. Pour afficher des informations détaillées sur l'exportation d'un cluster de bases de données spécifique, sélectionnez la tâche d'exportation.

AWS CLI

Pour surveiller les tâches d'exportation d'un cluster de bases de données à l'aide de AWS CLI, utilisez la commande describe-export-tasks.

L'exemple suivant montre comment afficher les informations actuelles sur toutes vos exportations de cluster de bases de données.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2022-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:parameter-groups-
test"
        },
{
            "Status": "COMPLETE",
            "TaskStartTime": "2022-10-31T20:58:06.998Z",
            "TaskEndTime": "2022-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "SourceArn": "arn:aws:rds:us-
west-2:123456789012:cluster:example-1-2019-10-31-06-44"
        },
        {
            "Status": "FAILED",
            "TaskEndTime": "2022-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket amzn-s3-demo-bucket2 isn't located in the
 current AWS Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "S3Bucket": "amzn-s3-demo-bucket2",
```

```
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "SourceArn": "arn:aws:rds:us-
west-2:123456789012:cluster:example-1-2019-10-30-06-45"
        }
    ]
}
```

Pour afficher des informations sur une tâche d'exportation spécifique, incluez l'option `--export-task-identifier` avec la commande `describe-export-tasks`. Pour filtrer la sortie, incluez l'option `--Filters`. Pour plus d'options, consultez la commande describe-export-tasks .

API RDS

Pour afficher des informations sur les exportations de cluster de bases de données à l'aide de l'API Amazon RDS, utilisez l'opération DescribeExportTasks.

Pour suivre l'achèvement du workflow d'exportation ou pour initier un autre workflow, vous pouvez vous abonner à des rubriques Amazon Simple Notification Service. Pour plus d'informations sur Amazon SNS, consultez Utiliser la notification d'événements d'Amazon RDS.

Annulation d'une tâche d'exportation d'un cluster de bases de données

Vous pouvez annuler une tâche d'exportation de cluster de bases de données à l'aide de la AWS Management Console, d'AWS CLI ou de l'API RDS.

> **Note**
>
> L'annulation d'une tâche d'exportation ne supprime pas les données qui ont été exportées vers Amazon S3. Pour plus d'informations sur la suppression des données à l'aide de la console, consultez Comment supprimer des objets d'un compartiment S3 ?. Pour supprimer les données à l'aide de la CLI, utilisez la commande delete-object.

Console

Pour annuler une tâche d'exportation de cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Exports in Amazon S3 (Exportations dans Amazon S3).

   Les exportations de cluster de bases de données sont indiquées dans la colonne Source type (Type de source). L'état de l'exportation est affiché dans la colonne Status (État).
3. Sélectionnez la tâche d'exportation que vous souhaitez annuler.
4. Choisissez Cancel (Annuler).
5. Choisissez Cancel export task (Annuler la tâche d'exportation) sur la page de confirmation.

AWS CLI

Pour annuler une tâche d'exportation à l'aide de AWS CLI, utilisez la commande cancel-export-task. La commande requiert l'option `--export-task-identifier`.

Example

```
aws rds cancel-export-task --export-task-identifier my-export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my-export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "SourceArn": "arn:aws:rds:us-west-2:123456789012:cluster:export-example-1"
}
```

API RDS

Pour annuler une tâche d'exportation à l'aide de l'API Amazon RDS, utilisez l'opération CancelExportTask avec le paramètre `ExportTaskIdentifier`.

Résolution des problèmes liés aux exportations de clusters de bases de données

Consultez les sections suivantes afin de résoudre les messages d'échec et les erreurs d'autorisation PostgreSQL dans le cadre des exportations de clusters de bases de données vers Amazon S3.

Messages d'échec relatifs aux tâches d'exportation Amazon S3

Le tableau suivant décrit les messages renvoyés en cas d'échec des tâches d'exportation Amazon S3.

Message d'échec	Description
Impossible de trouver ou d'accéder au cluster de bases de données source : [nom du cluster]	Le cluster de bases de données source ne peut pas être cloné.
Une erreur interne inconnue s'est produite.	La tâche a échoué en raison d'une erreur inconnue, d'une exception ou d'un échec.
Une erreur interne inconnue s'est produite lors de l'écriture des métadonnées de la tâche d'exporta tion dans le compartiment S3 [nom du compartiment].	La tâche a échoué en raison d'une erreur inconnue, d'une exception ou d'un échec.
L'exportation RDS n'a pas réussi à écrire les métadonnées de la tâche d'exportation, car elle ne peut pas assumer le rôle IAM [ARN du rôle].	La tâche d'exportation assume votre rôle IAM pour vérifier si elle est autorisée à écrire des métadonnées dans votre compartiment S3. Si la tâche ne peut pas assumer votre rôle IAM, elle échoue.
L'exportation RDS n'a pas réussi à écrire les métadonnées de la tâche d'exportation dans le compartiment S3 [nom du compartiment] à l'aide du rôle IAM [ARN du rôle] avec la clé KMS [ID de la clé]. Code d'erreur : [code d'erreur]	Une ou plusieurs autorisations sont manquantes et dès lors, la tâche d'exportation ne peut pas accéder au compartiment S3. Ce message d'échec est généré lors de la réception de l'un des codes d'erreur suivants : • `AWSSecurityTokenServiceException` avec le code d'erreur `AccessDenied`

Message d’échec	Description
	• `AmazonS3Exception` avec le code d’erreur `NoSuchBucket` , `AccessDenied` , `KMS.KMSInvalidStateException` , `403 Forbidden` ou `KMS.DisabledException` Ces codes d’erreur indiquent que les paramètres sont mal configurés pour le rôle IAM, le compartiment S3 ou la clé KMS.
Le rôle IAM [ARN du rôle] n’est pas autorisé à appeler [action S3] sur le compartiment S3 [nom du compartiment]. Examinez vos autorisations et retentez l’exportation.	La politique IAM est mal configurée. L’autorisation pour l’action S3 spécifique sur le compartiment S3 est manquante, ce qui entraîne l’échec de la tâche d’exportation.
La vérification de la clé KMS a échoué. Vérifiez les informations d’identification de votre clé KMS et réessayez.	La vérification des informations d’identification de clé KMS a échoué.
La vérification des informations d’identification S3 a échoué. Vérifiez les autorisations de votre compartiment S3 et de la politique IAM.	La vérification des informations d’identification S3 a échoué.
Le compartiment S3 [nom du compartiment] n’est pas valide. Il n’est peut-être pas situé dans la Région AWS actuelle ou il n’existe pas. Vérifiez le nom de votre compartiment S3 et retentez l’exportation.	Le compartiment S3 n’est pas valide.
Le compartiment S3 [nom du compartiment] ne se trouve pas dans la Région AWS actuelle. Vérifiez le nom de votre compartiment S3 et retentez l’exportation.	Le compartiment S3 ne se trouve pas dans la bonne Région AWS.

Dépannage des erreurs d’autorisations PostgreSQL

Lors de l’exportation de bases de données PostgreSQL vers Amazon S3, vous pouvez voir une erreur `PERMISSIONS_DO_NOT_EXIST` indiquant que certaines tables ont été ignorées. Cette erreur se produit généralement lorsque le superutilisateur, que vous avez spécifié lors de la création du cluster de bases de données, ne dispose pas des autorisations nécessaires pour accéder à ces tables.

Pour corriger cette erreur, exécutez la commande suivante :

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

Pour plus d’informations sur les privilèges des superutilisateurs, consultez Privilèges du compte utilisateur principal.

Exportation de données d'instantanés de cluster de bases de données vers Amazon S3

Vous pouvez exporter des données d'instantanés de cluster de bases de données vers un compartiment Amazon S3. Le processus d'exportation s'exécute en arrière-plan et n'affecte pas les performances de votre cluster de bases de données actif.

Lorsque vous exportez un instantané de cluster de bases de données, Amazon Aurora extrait les données de l'instantané et les stocke dans un compartiment Amazon S3. Vous pouvez exporter des instantanés manuels et des instantanés système automatisés. Par défaut, toutes les données de l'instantané sont exportées. Toutefois, vous pouvez choisir d'exporter des ensembles spécifiques de bases de données, de schémas ou de tables.

> **Note**
>
> L'exportation de données à partir d'un instantané de cluster de bases de données nécessite la restauration de l'instantané. Les temps de restauration sont influencés par divers facteurs, tels que la quantité de trafic réseau reçu par la Région AWS par rapport à sa bande passante disponible. En cas d'augmentation soudaine du trafic, les délais d'exécution peuvent être plus longs que prévu.
>
> Une alternative pour réduire le délai d'exportation S3 pour les bases de données Aurora est l'exportation de clusters de bases de données en direct vers S3. L'exportation d'un cluster de bases de données implique des temps de démarrage plus courts que l'exportation d'un instantané de base de données, car il n'est pas nécessaire de restaurer un instantané. Pour plus d'informations, consultez Exportation des données du cluster de bases de données vers Amazon S3.

Les données sont stockées dans un format Apache Parquet qui est compressé et cohérent. En règle générale, la taille d'un fichier Parquet est comprise entre 1 et 10 Mo.

Une fois les données exportées, vous pouvez les analyser directement via des outils tels que Amazon Athena ou Amazon Redshift Spectrum. Pour plus d'informations sur l'utilisation Athena de la lecture des données de parquet, consultez Parquet SerDe dans le Amazon Athena Guide de l'utilisateur. Pour plus d'informations sur l'utilisation de Redshift Spectrum pour lire des données Parquet, consultez COPY depuis les formats de données en colonnes dans le Guide du développeur de base de données Amazon Redshift.

La disponibilité et la prise en charge des fonctionnalités varient selon les versions spécifiques de chaque moteur de base de données, et selon les Régions AWS. Pour plus d'informations sur la disponibilité des versions et des régions de l'exportation des données d'instantané de cluster de bases de données vers S3, consultez Régions et moteurs de bases de données Aurora pris en charge pour l'exportation des données d'instantanés vers Amazon S3.

Vous utilisez le processus suivant pour exporter des données d'instantané de base de données vers un compartiment Amazon S3. Pour plus de détails, consultez les sections suivantes.

Présentation de l'exportation des données d'instantané

1. Identifiez l'instantané à exporter.

   Utilisez un instantané automatisé ou manuel existant ou créez un instantané manuel d'une instance de base de données.

2. Configurez l'accès au compartiment Amazon S3.

   Un compartiment est un conteneur d'objets ou de fichiers Amazon S3. Pour fournir les informations permettant d'accéder à un compartiment, procédez comme suit :

   a. Identifiez le compartiment S3 vers lequel l'instantané doit être exporté. Le compartiment S3 doit se situer dans la même région AWS que l'instantané. Pour plus d'informations, consultez Identification du compartiment Amazon S3 pour l'exportation.

   b. Créez un rôle Gestion des identités et des accès AWS (IAM) qui accorde à la tâche d'exportation d'instantané l'accès au compartiment S3. Pour plus d'informations, consultez Fournir l'accès à un compartiment Amazon S3 à l'aide d'un rôle IAM.

3. Créez un chiffrement symétrique AWS KMS key pour le chiffrement côté serveur. La clé KMS est utilisée par la tâche d'exportation de l'instantané pour configurer le chiffrement côté serveur AWS KMS lors de l'écriture des données d'exportation dans S3.

   La politique de clés KMS doit inclure à la fois les autorisations `kms:CreateGrant` et `kms:DescribeKey`. Pour plus d'informations sur l'utilisation des clés KMS dans Amazon Aurora, consultez Gestion AWS KMS key.

   Si vous avez une instruction de refus dans votre politique de clés KMS, veillez à exclure explicitement le principal de service AWS `export.rds.amazonaws.com`.

   Vous pouvez utiliser une clé KMS dans votre compte AWS ou une clé KMS entre comptes. Pour plus d'informations, consultez Utiliser un compte croisé AWS KMS key.

4. Exportez l'instantané vers Amazon S3 à l'aide de la console ou de la commande de CLI `start-export-task`. Pour plus d'informations, consultez Création de tâches d'exportation d'instantanés.
5. Pour accéder aux données exportées dans le compartiment Amazon S3, consultez Chargement, téléchargement et gestion d'objets dans le Guide de l'utilisateur Amazon Simple Storage Service.

Découvrez dans les sections suivantes comment configurer, exporter, surveiller et annuler les tâches d'exportation d'instantanés de cluster de bases de données et comment résoudre les problèmes associés.

Rubriques

Considérations relatives aux exportations d'instantanés de base de données

Limitations

L'exportation de données d'instantané de bases de données vers Amazon S3 présente les limites suivantes :

- Vous ne pouvez pas exécuter simultanément plusieurs tâches d'exportation pour le même instantané de cluster de bases de données. Cette règle s'applique aux exportations complètes et partielles.
- Vous pouvez avoir jusqu'à cinq tâches d'exportation d'instantané de base de données simultanées en cours par Compte AWS.
- Vous ne pouvez pas exporter les données d'instantanés des clusters de bases de données Aurora Serverless v1 vers S3.

- Les exportations vers S3 ne prennent pas en charge les préfixes S3 contenant le signe des deux-points (:).
- Les caractères suivants du chemin d'accès au fichier S3 sont convertis en traits de soulignement (_) lors de l'exportation :

```
\ ` " (space)
```

- Si une base de données, un schéma ou une table comporte des caractères autres que les suivants, l'exportation partielle n'est pas prise en charge. Toutefois, vous pouvez exporter l'intégralité de l'instantané de base de données.
  - Lettres latines (A–Z)
  - Chiffres (0–9)
  - Symbole dollar ($)
  - Trait de soulignement (_)
- Les espaces ( ) et certains caractères ne sont pas pris en charge dans les noms de colonnes des tables de base de données. Les tables dont les noms de colonnes contiennent les caractères suivants sont ignorées lors de l'exportation :

```
, ; { } ( ) \n \t = (space)
```

- Les tables dont les noms contiennent des barres obliques (/) sont ignorées lors de l'exportation.
- Les tables temporaires et non journalisées d'Aurora PostgreSQL sont ignorées lors de l'exportation.
- Si les données contiennent un objet volumineux tel qu'un objet BLOB ou CLOB proche de ou supérieur à 500 Mo, l'exportation échoue.
- Si une table contient une grande ligne proche de ou supérieure à 2 Go, la table est ignorée lors de l'exportation.
- Pour les exportations partielles, la taille maximale de la liste `ExportOnly` est de 200 Ko.
- Nous vous recommandons vivement d'utiliser un nom unique pour chaque tâche d'exportation. Si vous n'utilisez pas un nom de tâche unique, vous risquez de recevoir le message d'erreur suivant :

  ExportTaskAlreadyExistsFault: An error occurred (ExportTaskAlreadyExists) when calling the StartExportTask operation: The export task with the ID *xxxxx* already exists (ExportTaskAlreadyExistsFault : une erreur s'est produite (ExportTaskAlreadyExists) lors de l'appel de l'opération StartExportTask : la tâche d'exportation avec l'ID xxxxx existe déjà).

- Vous pouvez supprimer un instantané lors de l'exportation de ses données vers S3, mais les coûts de stockage de cet instantané vous sont tout de même facturés tant que la tâche d'exportation n'est pas terminée.
- Vous ne pouvez pas restaurer les données des instantanés exportées de S3 vers un nouveau cluster de bases de données.

Convention de dénomination de fichiers

Les données exportées pour des tables spécifiques sont stockées au format *base_prefix*/*files*, qui utilise le préfixe de base suivant :

```
export_identifier/database_name/schema_name.table_name/
```

Par exemple :

```
export-1234567890123-459/rdststdb/rdststdb.DataInsert_7ADB5D19965123A2/
```

Il existe deux conventions de dénomination des fichiers.

- Convention actuelle :

  ```
  batch_index/part-partition_index-random_uuid.format-based_extension
  ```

  L'indice de lot est un numéro de séquence qui représente un lot de données lues dans la table. Si votre table ne peut pas être partitionnée en petits segments à exporter en parallèle, il y aura plusieurs indices de lot. Il en va de même si votre table est partitionnée en plusieurs tables. Il y aura plusieurs indices de lot, un pour chacune des partitions effectuées à partir de la table principale.

  Si votre table peut être partitionnée en petits segments à lire en parallèle, il n'y aura que le dossier 1 d'indice de lot.

  Ce dossier inclut un ou plusieurs fichiers Parquet qui contiennent les données de votre table. Le préfixe du nom du fichier Parquet est `part-`*partition_index*. Si votre table est partitionnée, plusieurs fichiers commencent par l'indice de partition `00000`.

Il peut y avoir des écarts dans la séquence d'indices de partition. Cela se produit parce que chaque partition est générée à partir d'une requête basée sur une plage de données dans votre table. S'il n'y a pas de données dans la plage de cette partition, le numéro de séquence est ignoré.

Supposons, par exemple, que la colonne `id` soit la clé primaire de la table et que ses valeurs minimale et maximale soient `100` et `1000`. 9 Lorsque nous essayons d'exporter cette table avec 9 partitions, nous la lisons avec des requêtes parallèles telles que les suivantes :

```
SELECT * FROM table WHERE id <= 100 AND id < 200
 SELECT * FROM table WHERE id <= 200 AND id < 300
```

Cela devrait générer 9 fichiers, allant de `part-00000-`*`random_uuid`*`.gz.parquet` à `part-00008-`*`random_uuid`*`.gz.parquet`. Toutefois, s'il n'existe aucune ligne dont les identifiants sont compris entre `200` et `350`, l'une des partitions terminées est vide. Aucun fichier ne sera donc créé pour elle. Dans l'exemple précédent, `part-00001-`*`random_uuid`*`.gz.parquet` n'est pas créé.

- Ancienne convention :

```
part-partition_index-random_uuid.format-based_extension
```

Elle est identique à la convention actuelle, mais sans le préfixe *`batch_index`*, par exemple :

```
part-00000-c5a881bb-58ff-4ee6-1111-b41ecff340a3-c000.gz.parquet
 part-00001-d7a881cc-88cc-5ab7-2222-c41ecab340a4-c000.gz.parquet
 part-00002-f5a991ab-59aa-7fa6-3333-d41eccd340a7-c000.gz.parquet
```

La convention de dénomination de fichiers est sujette à modification. Par conséquent, lors de la lecture des tables cibles, nous vous conseillons de lire tout ce qui se trouve à l'intérieur du préfixe de base de la table.

Conversion des données lors de l'exportation vers un compartiment Amazon S3

Lorsque vous exportez un instantané de base de données vers un compartiment Amazon S3, Amazon Aurora convertit les données, les exporte et les stocke au format Parquet. Pour plus d'informations sur Parquet, consultez le site web Apache Parquet.

Parquet stocke toutes les données sous l'un des types primitifs suivants :

- BOOLEAN
- INT32
- INT64
- INT96
- FLOAT
- DOUBLE
- BYTE_ARRAY – Tableau d'octets de longueur variable, également connu sous le nom de binaire
- FIXED_LEN_BYTE_ARRAY – Tableau d'octets de longueur fixe utilisé lorsque les valeurs ont une taille constante

Les types de données Parquet sont peu nombreux afin de la complexité de la lecture et de l'écriture du format. Parquet fournit des types logiques pour étendre les types primitifs. Un type logique est implémenté sous forme d'annotation avec les données dans un champ de métadonnées `LogicalType`. L'annotation de type logique explique comment interpréter le type primitif.

Lorsque le type logique `STRING` annote un type `BYTE_ARRAY`, il indique que le tableau d'octets doit être interprété comme une chaîne de caractères codée en UTF-8. Une fois la tâche d'exportation terminée, Amazon Aurora vous avertit si une conversion de chaîne s'est produite. Les données sous-jacentes exportées sont toujours les mêmes que celles de la source. Cependant, en raison de la différence d'encodage en UTF-8, certains caractères peuvent apparaître différents de la source lorsqu'ils sont lus dans des outils tels que Athena.

Pour plus d'informations, consultez Parquet Logical Type Definitions dans la documentation Parquet.

Rubriques

- Mappage du type de données MySQL à Parquet
- Mappage de type de données PostgreSQL vers Parquet

Mappage du type de données MySQL à Parquet

Le tableau suivant montre le mappage des types de données MySQL aux types de données Parquet lorsque les données sont converties et exportées vers Amazon S3.

Type de données source	Type primitif du format Parquet	Annotation de type logique	Notes de conversion
Types de données numériques			
BIGINT	INT64		
BIGINT UNSIGNED	FIXED_LEN _BYTE_ARRAY(9)	DECIMAL(20,0)	Parquet ne prend en charge que les types signés, de sorte que le mappage nécessite un octet supplémentaire (8 plus 1) pour stocker le type BIGINT_UN SIGNED.
BIT	BYTE_ARRAY		
DECIMAL	INT32	DECIMAL(p,s)	Si la valeur source est inférieure à 2^{31}, elle est stockée sous la forme INT32.
	INT64	DECIMAL(p,s)	Si la valeur source est supérieure ou égale à 2^{31} mais inférieure à 2^{63}, elle est stockée sous la forme INT64.
	FIXED_LEN _BYTE_ARRAY(N)	DECIMAL(p,s)	Si la valeur source est supérieure ou égale à 2^{63}, elle est stockée sous la forme FIXED_LEN _BYTE_ARRAY(N).

Type de données source	Type primitif du format Parquet	Annotation de type logique	Notes de conversion
	BYTE_ARRAY	STRING	Parquet ne prend pas en charge une précision décimale supérieure à 38. La valeur décimale est convertie en une chaîne de type BYTE_ARRAY et encodée en UTF8.
DOUBLE	DOUBLE		
FLOAT	DOUBLE		
INT	INT32		
INT UNSIGNED	INT64		
MEDIUMINT	INT32		
MEDIUMINT UNSIGNED	INT64		
NUMERIC	INT32	DECIMAL(p,s)	Si la valeur source est inférieure à 2^{31}, elle est stockée sous la forme INT32.
	INT64	DECIMAL(p,s)	Si la valeur source est supérieure ou égale à 2^{31} mais inférieure à 2^{63}, elle est stockée sous la forme INT64.

Type de données source	Type primitif du format Parquet	Annotation de type logique	Notes de conversion
	FIXED_LEN _ARRAY(N)	DECIMAL(p,s)	Si la valeur source est supérieure ou égale à 2^{63}, elle est stockée sous la forme FIXED_LEN _BYTE_ARRAY(N).
	BYTE_ARRAY	STRING	Parquet ne prend pas en charge la précision numérique supérieure à 38. Cette valeur numérique est convertie en une chaîne de type BYTE_ARRAY et encodée en UTF8.
SMALLINT	INT32		
SMALLINT UNSIGNED	INT32		
TINYINT	INT32		
TINYINT UNSIGNED	INT32	INT(16, true)	
Types de données chaîne			
BINARY	BYTE_ARRAY		
BLOB	BYTE_ARRAY		
CHAR	BYTE_ARRAY		
ENUM	BYTE_ARRAY	STRING	

Type de données source	Type primitif du format Parquet	Annotation de type logique	Notes de conversion
LINESTRING	BYTE_ARRAY		
LONGBLOB	BYTE_ARRAY		
LONGTEXT	BYTE_ARRAY	STRING	
MEDIUMBLOB	BYTE_ARRAY		
MEDIUMTEXT	BYTE_ARRAY	STRING	
MULTILINESTRING	BYTE_ARRAY		
SET	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TINYBLOB	BYTE_ARRAY		
TINYTEXT	BYTE_ARRAY	STRING	
VARBINARY	BYTE_ARRAY		
VARCHAR	BYTE_ARRAY	STRING	
Types de données de date et d’heure			
DATE	BYTE_ARRAY	STRING	Une date est convertie en une chaîne de type BYTE_ARRAY et encodée en UTF8.
DATETIME	INT64	TIMESTAMP_MICROS	

Type de données source	Type primitif du format Parquet	Annotation de type logique	Notes de conversion
TIME	BYTE_ARRAY	STRING	Un type TIME est converti en une chaîne BYTE_ARRAY et encodé en UTF8.
TIMESTAMP	INT64	TIMESTAMP _MICROS	
YEAR	INT32		
Types de données géométriques			
GEOMETRY	BYTE_ARRAY		
GEOMETRYC OLLECTION	BYTE_ARRAY		
MULTIPOINT	BYTE_ARRAY		
MULTIPOLYGON	BYTE_ARRAY		
POINT	BYTE_ARRAY		
POLYGON	BYTE_ARRAY		
Type de données JSON			
JSON	BYTE_ARRAY	STRING	

Mappage de type de données PostgreSQL vers Parquet

Le tableau suivant montre le mappage des types de données PostgreSQL aux types de données Parquet lorsque les données sont converties et exportées vers Amazon S3.

Type de données PostgreSQL	Type primitif du format Parquet	Annotation de type logique	Notes de mappage
Types de données numériques			
BIGINT	INT64		
BIGSERIAL	INT64		
DECIMAL	BYTE_ARRAY	STRING	Un type DECIMAL est converti en une chaîne de type BYTE_ARRAY et encodé en UTF8. Cette conversio n vise à éviter les complications dues à la précision des données et aux valeurs de données qui ne sont pas un nombre (NaN).
DOUBLE PRECISION	DOUBLE		
INTEGER	INT32		
MONEY	BYTE_ARRAY	STRING	
REAL	FLOAT		
SERIAL	INT32		
SMALLINT	INT32	INT(16, true)	
SMALLSERIAL	INT32	INT(16, true)	
Types de chaînes et de données associés			

Type de données PostgreSQL	Type primitif du format Parquet	Annotation de type logique	Notes de mappage
ARRAY	BYTE_ARRAY	STRING	Un tableau est converti en chaîne et encodé en tant que BINARY (UTF8). Cette conversion vise à éviter les complications dues à la précision des données, aux valeurs de données qui ne sont pas un nombre (NaN) et aux valeurs de données horaires.
BIT	BYTE_ARRAY	STRING	
BIT VARYING	BYTE_ARRAY	STRING	
BYTEA	BINARY		
CHAR	BYTE_ARRAY	STRING	
CHAR(N)	BYTE_ARRAY	STRING	
ENUM	BYTE_ARRAY	STRING	
NAME	BYTE_ARRAY	STRING	
TEXT	BYTE_ARRAY	STRING	
TEXT SEARCH	BYTE_ARRAY	STRING	
VARCHAR(N)	BYTE_ARRAY	STRING	
xml	BYTE_ARRAY	STRING	

Type de données PostgreSQL	Type primitif du format Parquet	Annotation de type logique	Notes de mappage
Types de données de date et d'heure			
DATE	BYTE_ARRAY	STRING	
INTERVAL	BYTE_ARRAY	STRING	
TIME	BYTE_ARRAY	STRING	
TIME WITH TIME ZONE	BYTE_ARRAY	STRING	
TIMESTAMP	BYTE_ARRAY	STRING	
TIMESTAMP WITH TIME ZONE	BYTE_ARRAY	STRING	
Types de données géométriques			
BOX	BYTE_ARRAY	STRING	
CIRCLE	BYTE_ARRAY	STRING	
LINE	BYTE_ARRAY	STRING	
LINESEGMENT	BYTE_ARRAY	STRING	
PATH	BYTE_ARRAY	STRING	
POINT	BYTE_ARRAY	STRING	
POLYGON	BYTE_ARRAY	STRING	
Types de données JSON			
JSON	BYTE_ARRAY	STRING	
JSONB	BYTE_ARRAY	STRING	
Autres types de données			

Type de données PostgreSQL	Type primitif du format Parquet	Annotation de type logique	Notes de mappage
BOOLEAN	BOOLEAN		
CIDR	BYTE_ARRAY	STRING	Type de données de réseau
COMPOSITE	BYTE_ARRAY	STRING	
DOMAIN	BYTE_ARRAY	STRING	
INET	BYTE_ARRAY	STRING	Type de données de réseau
MACADDR	BYTE_ARRAY	STRING	
OBJECT IDENTIFIER	N/A		
PG_LSN	BYTE_ARRAY	STRING	
RANGE	BYTE_ARRAY	STRING	
UUID	BYTE_ARRAY	STRING	

Configuration de l’accès à un compartiment Amazon S3

Vous identifiez le compartiment Amazon S3, puis vous donnez à l’instantané la permission d’y accéder.

Rubriques

- Identification du compartiment Amazon S3 pour l’exportation
- Fournir l’accès à un compartiment Amazon S3 à l’aide d’un rôle IAM
- Utilisation d’un compartiment Amazon S3 entre comptes
- Utiliser un compte croisé AWS KMS key

Identification du compartiment Amazon S3 pour l’exportation

Identifiez le compartiment Amazon S3 vers lequel exporter l’instantané de base de données. Utilisez un compartiment S3 existant ou créez un nouveau compartiment S3.

> **Note**
>
> Le compartiment S3 vers lequel effectuer l'exportation doit se trouver dans la même AWS région que le snapshot.

Pour plus d’informations sur l’utilisation des Amazon S3 compartiments, consultez les points suivants dans le Guide de l’utilisateur Amazon Simple Storage Service :

- Comment afficher les propriétés d’un compartiment S3 ?
- Comment activer le chiffrement par défaut pour un compartiment Amazon S3 ?
- Comment créer un compartiment S3 ?

Fournir l’accès à un compartiment Amazon S3 à l’aide d’un rôle IAM

Avant d’exporter les données d’instantané de base de données vers Amazon S3, vous devez accorder aux tâches d’exportation d’instantané une autorisation d’accès en écriture au compartiment Amazon S3.

Pour accorder cette autorisation, créez une politique IAM qui donne accès au compartiment, puis créez un rôle IAM et attachez la politique au rôle. Vous pouvez ultérieurement affecter le rôle IAM à votre tâche d’exportation d’instantané.

> **Important**
>
> Si vous prévoyez d'utiliser le AWS Management Console pour exporter votre instantané, vous pouvez choisir de créer la politique IAM et le rôle automatiquement lorsque vous exportez le cliché. Pour obtenir des instructions, veuillez consulter Création de tâches d’exportation d’instantanés.

Pour accorder aux tâches d'instantané de base de données l'accès à Amazon S3

1. Créez une politique IAM. Cette politique fournit les autorisations d'accès au compartiment et aux objets qui permettent à votre tâche d'exportation d'instantané d'accéder à Amazon S3.

   Dans la politique, incluez les actions obligatoires suivantes pour permettre le transfert de fichiers depuis Amazon Aurora vers un compartiment S3 :

   - `s3:PutObject*`
   - `s3:GetObject*`
   - `s3:ListBucket`
   - `s3:DeleteObject*`
   - `s3:GetBucketLocation`

   Dans la politique, incluez les ressources suivantes pour identifier le compartiment S3 et les objets qu'il contient. La liste de ressources suivante indique le format Amazon Resource Name (ARN) pour l'accès à Amazon S3.

   - `arn:aws:s3:::amzn-s3-demo-bucket`
   - `arn:aws:s3:::amzn-s3-demo-bucket/*`

   Pour plus d'informations sur la création d'une politique IAM pour Amazon Aurora, consultez Création et utilisation d'une politique IAM pour l'accès à une base de données IAM. Consultez également Didacticiel : création et attachement de votre première politique gérée par le client dans le Guide de l'utilisateur IAM.

   La AWS CLI commande suivante crée une politique IAM nommée `ExportPolicy` avec ces options. Elle accorde un accès à un compartiment nommé *`amzn-s3-demo-bucket`*.

   > **Note**
   >
   > Après avoir créé la politique, notez son ARN. Vous en aurez besoin par la suite pour attacher la politique à un rôle IAM.

```
aws iam create-policy  --policy-name ExportPolicy --policy-document '{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ExportPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}'
```

2. Créez un rôle IAM, afin qu'Aurora puisse endosser ce rôle IAM en votre nom pour accéder à vos compartiments Amazon S3. Pour plus d'informations, consultez Création d'un rôle pour déléguer des autorisations à un utilisateur IAM dans le Guide de l'utilisateur IAM.

   L'exemple suivant montre comment utiliser la AWS CLI commande pour créer un rôle nommérds-s3-export-role.

```
aws iam create-role  --role-name rds-s3-export-role  --assume-role-policy-document
 '{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
           "Service": "export.rds.amazonaws.com"
         },
        "Action": "sts:AssumeRole"
      }
    ]
  }'
```

3. Attachez la politique IAM que vous avez créée au rôle IAM que vous venez de créer.

La AWS CLI commande suivante associe la politique créée précédemment au rôle nommé`rds-s3-export-role`. Remplacez *`your-policy-arn`* par l'ARN de stratégie que vous avez noté lors d'une étape précédente.

```
aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-export-role
```

Utilisation d'un compartiment Amazon S3 entre comptes

Vous pouvez utiliser des compartiments Amazon S3 sur plusieurs AWS comptes. Pour utiliser un compartiment entre comptes, ajoutez une politique de compartiment afin d'autoriser l'accès au rôle IAM que vous utilisez pour les exportations S3. Pour plus d'informations, consultez Exemple 2 : propriétaire d'un compartiment accordant à ses utilisateurs des autorisations entre comptes sur un compartiment.

- Attachez une politique de compartiment à votre compartiment, comme illustré dans l'exemple suivant.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/Admin"
            },
            "Action": [
                "s3:PutObject*",
                "s3:ListBucket",
                "s3:GetObject*",
                "s3:DeleteObject*",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
```

```
        }
    ]
}
```

Utiliser un compte croisé AWS KMS key

Vous pouvez utiliser un compte croisé AWS KMS key pour chiffrer les exportations Amazon S3. Tout d'abord, vous ajoutez une politique de clé au compte local, puis vous ajoutez des politiques IAM au compte externe. Pour plus d'informations, consultez Autorisation des utilisateurs d'autres comptes à utiliser une clé KMS.

Pour utiliser une clé KMS entre comptes

1. Ajoutez une politique de clé au compte local.

   L'exemple suivant accorde `ExampleRole` et `ExampleUser` dans les autorisations 444455556666 du compte externe dans le compte local 123456789012.

   ```
   {
       "Sid": "Allow an external account to use this KMS key",
       "Effect": "Allow",
       "Principal": {
           "AWS": [
               "arn:aws:iam::444455556666:role/ExampleRole",
               "arn:aws:iam::444455556666:user/ExampleUser"
           ]
       },
       "Action": [
           "kms:Encrypt",
           "kms:Decrypt",
           "kms:ReEncrypt*",
           "kms:GenerateDataKey*",
           "kms:CreateGrant",
           "kms:DescribeKey",
           "kms:RetireGrant"
       ],
       "Resource": "*"
   }
   ```

2. Ajoutez des politiques IAM au compte externe.

L'exemple de stratégie IAM suivant autorise le principal à utiliser la clé KMS dans le compte 123456789012 pour les opérations cryptographiques. Pour accorder cette autorisation aux `ExampleRole` et `ExampleUser` du compte 444455556666, attachez-leur la politique dans ce compte.

```
{
    "Sid": "Allow use of KMS key in account 123456789012",
    "Effect": "Allow",
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:CreateGrant",
        "kms:DescribeKey",
        "kms:RetireGrant"
    ],
    "Resource": "arn:aws:kms:us-
west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
}
```

Création de tâches d'exportation d'instantanés

Créez des tâches d'exportation d'instantané pour exporter les données de votre instantané vers un compartiment Amazon S3. Vous pouvez avoir jusqu'à cinq tâches d'exportation d'instantané de base de données simultanées en cours par Compte AWS.

> **Note**
>
> L'exportation d'instantanés RDS peut prendre un certain temps en fonction du type et de la taille de votre base de données. La tâche d'exportation commence par restaurer et mettre à l'échelle l'ensemble de la base de données avant d'extraire les données vers Amazon S3. La progression de la tâche au cours de cette phase s'affiche sous l'intitulé Starting. Lorsque la tâche passe à l'exportation de données vers S3, la progression affiche l'intitulé En cours. La durée nécessaire à l'exportation dépend des données stockées dans la base de données. Par exemple, l'exportation des tables comportant des colonnes numériques d'index ou de clé primaire bien distribuées est la plus rapide. L'opération prend plus de temps pour les tables qui ne contiennent pas de colonne adaptée au partitionnement et les tables avec un

seul index sur une colonne basée sur une chaîne. Ce délai d'exportation est plus long, car l'exportation utilise un processus à thread unique plus lent.

Vous pouvez exporter un instantané de base de données vers Amazon S3 à l'aide de l'AWS Management Console, de l'AWS CLI ou de l'API RDS.

Si vous utilisez une fonction Lambda pour exporter un instantané, ajoutez l'action `kms:DescribeKey` à la stratégie de fonction Lambda. Pour plus d'informations, consultez Autorisations AWS Lambda.

Console

L'option de console Exporter vers Amazon S3 s'affiche uniquement pour les instantanés pouvant être exportés vers Amazon S3. Un instantané peut ne pas être disponible pour l'exportation pour les raisons suivantes :

- Le moteur de base de données n'est pas pris en charge pour l'exportation S3.
- La version de l'instance de base de données n'est pas prise en charge pour l'exportation S3.
- L'exportation S3 n'est pas prise en charge dans la région AWS où l'instantané a été créé.

Pour exporter un instantané de base de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Snapshots.
3. Dans les onglets, choisissez le type d'instantané que vous souhaitez exporter.
4. Dans la liste des instantanés, choisissez celui que vous souhaitez exporter.
5. Pour actions, choisissez Export to Amazon S3 (Exporter vers Amazon S3).

   La fenêtre Export to Amazon S3 (Exporter vers Amazon S3) apparaît.
6. Dans Export Identifier (Identifiant d'exportation), entrez un nom pour identifier la tâche d'exportation. Cette valeur est également utilisée pour le nom du fichier créé dans le compartiment S3.
7. Choisissez les données à exporter :

   - Choisissez All (Tout) pour exporter toutes les données de l'instantané.

- Choisissez Partial (Partiel) pour exporter des parties spécifiques de l'instantané. Pour identifier les parties de l'instantané à exporter, entrez un(e) ou plusieurs bases de données, schémas ou tables pour Identifiants, séparés par des espaces.

  Utilisez le format suivant :

  ```
  database[.schema][.table] database2[.schema2][.table2] ... databasen[.scheman]
  [.tablen]
  ```

  Exemples :

  ```
  mydatabase mydatabase2.myschema1 mydatabase2.myschema2.mytable1
   mydatabase2.myschema2.mytable2
  ```

8. Pour S3 bucket (Compartiment S3), choisissez le compartiment vers lequel exporter.

   Pour affecter les données exportées à un chemin d'accès de dossier dans le compartiment S3, entrez le chemin d'accès facultatif pour S3 prefix (Préfixe S3).

9. Pour Rôle IAM, choisissez un rôle qui vous accorde un accès en écriture au compartiment S3 choisi, ou créez un nouveau rôle.

   - Si vous avez créé un rôle en suivant les étapes décrites dans Fournir l'accès à un compartiment Amazon S3 à l'aide d'un rôle IAM, choisissez ce rôle.
   - Si vous n'avez pas créé un rôle qui vous accorde un accès en écriture au compartiment S3 que vous avez choisi, choisissez Create a new role (Créer un nouveau rôle) pour créer le rôle automatiquement. Ensuite, saisissez un nom pour le rôle dans Nom du rôle IAM.

10. Pour AWS KMS key, entrez l'ARN de la clé à utiliser pour chiffrer les données exportées.
11. Choisissez Export to Amazon S3 (Exporter vers Amazon S3).

AWS CLI

Pour exporter un instantané de base de données vers Amazon S3 à l'aide de l'AWS CLI, utilisez la commande start-export-task avec les options requises suivantes :

- `--export-task-identifier`
- `--source-arn`
- `--s3-bucket-name`

- `--iam-role-arn`
- `--kms-key-id`

Dans les exemples suivants, la tâche d'exportation d'instantané est nommée *my_snapshot_export*. Elle exporte un instantané vers un compartiment S3 nommé *amzn-s3-demo-destination-bucket*.

Example

Pour Linux, macOS ou Unix :

```
aws rds start-export-task \
    --export-task-identifier my-snapshot-export \
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name \
    --s3-bucket-name amzn-s3-demo-destination-bucket \
    --iam-role-arn iam-role \
    --kms-key-id my-key
```

Pour Windows :

```
aws rds start-export-task ^
    --export-task-identifier my-snapshot-export ^
    --source-arn arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name ^
    --s3-bucket-name amzn-s3-demo-destination-bucket ^
    --iam-role-arn iam-role ^
    --kms-key-id my-key
```

Vous trouverez ci-après un exemple de sortie.

```
{
    "Status": "STARTING",
    "IamRoleArn": "iam-role",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-destination-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "my-key",
    "ExportTaskIdentifier": "my-snapshot-export",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:snapshot-name"
}
```

Pour fournir un chemin de dossier dans le compartiment S3 pour l'exportation d'instantané, incluez l'option `--s3-prefix` dans la commande start-export-task .

API RDS

Pour exporter un instantané de base de données vers Amazon S3 à l'aide de l'API Amazon RDS, utilisez l'opération StartExportTask avec les paramètres requis suivants :

- `ExportTaskIdentifier`
- `SourceArn`
- `S3BucketName`
- `IamRoleArn`
- `KmsKeyId`

Surveillance des exportations d'instantanés

Vous pouvez surveiller les exportations d'instantanés de bases de données à l'aide de l'AWS Management Console, de l'AWS CLI ou de l'API RDS.

Console

Pour surveiller les exportations d'instantanés de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Exports in Amazon S3 (Exportations dans Amazon S3).

   Les exportations d'instantanés de la base de données sont indiquées dans la colonne Source type (Type de source). L'état de l'exportation est affiché dans la colonne Status (État).
3. Pour afficher des informations détaillées sur une exportation d'instantané spécifique, choisissez la tâche d'exportation.

AWS CLI

Pour surveiller les exportations d'instantanés de bases de données à l'aide de l'AWS CLI, utilisez la commande describe-export-tasks .

L’exemple suivant montre comment afficher les informations actuelles sur toutes vos exportations d’instantanés.

Example

```
aws rds describe-export-tasks

{
    "ExportTasks": [
        {
            "Status": "CANCELED",
            "TaskEndTime": "2019-11-01T17:36:46.961Z",
            "S3Prefix": "something",
            "ExportTime": "2019-10-24T20:23:48.364Z",
            "S3Bucket": "amzn-s3-demo-bucket",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/
bPxRfiCYEXAMPLEKEY",
            "ExportTaskIdentifier": "anewtest",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-25T19:10:58.885Z",
            "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:parameter-
groups-test"
        },
{
            "Status": "COMPLETE",
            "TaskEndTime": "2019-10-31T21:37:28.312Z",
            "WarningMessage": "{\"skippedTables\":[],\"skippedObjectives\":[],\"general
\":[{\"reason\":\"FAILED_TO_EXTRACT_TABLES_LIST_FOR_DATABASE\"}]}",
            "S3Prefix": "",
            "ExportTime": "2019-10-31T06:44:53.452Z",
            "S3Bucket": "amzn-s3-demo-bucket1",
            "PercentProgress": 100,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "thursday-events-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 263,
            "TaskStartTime": "2019-10-31T20:58:06.998Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-31-06-44"
        },
        {
```

```
            "Status": "FAILED",
            "TaskEndTime": "2019-10-31T02:12:36.409Z",
            "FailureCause": "The S3 bucket my-exports isn't located in the current AWS
 Region. Please, review your S3 bucket name and retry the export.",
            "S3Prefix": "",
            "ExportTime": "2019-10-30T06:45:04.526Z",
            "S3Bucket": "amzn-s3-demo-bucket2",
            "PercentProgress": 0,
            "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/2Zp9Utk/
h3yCo8nvbEXAMPLEKEY",
            "ExportTaskIdentifier": "wednesday-afternoon-test",
            "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
            "TotalExtractedDataInGB": 0,
            "TaskStartTime": "2019-10-30T22:43:40.034Z",
            "SourceArn":
 "arn:aws:rds:AWS_Region:123456789012:snapshot:rds:example-1-2019-10-30-06-45"
        }
    ]
}
```

Pour afficher des informations sur une exportation d'instantané spécifique, incluez l'option `--export-task-identifier` avec la commande `describe-export-tasks`. Pour filtrer la sortie, incluez l'option `--Filters`. Pour plus d'options, consultez la commande describe-export-tasks .

API RDS

Pour afficher des informations sur les exportations d'instantanés de bases de données à l'aide de l'API Amazon RDS, utilisez l'opération DescribeExportTasks .

Pour suivre l'achèvement du workflow d'exportation ou pour initier un autre workflow, vous pouvez vous abonner à des rubriques Amazon Simple Notification Service. Pour plus d'informations sur Amazon SNS, consultez Utiliser la notification d'événements d'Amazon RDS.

Annulation d'une tâche d'exportation d'instantané

Vous pouvez annuler une tâche d'exportation d'instantané de base de données à l'aide de l'AWS Management Console, de l'AWS CLI ou de l'API RDS.

> **Note**
>
> L'annulation d'une tâche d'exportation d'instantané ne supprime aucune des données exportées vers Amazon S3. Pour plus d'informations sur la suppression des données à

l'aide de la console, consultez Comment supprimer des objets d'un compartiment S3 ?. Pour supprimer les données à l'aide de la CLI, utilisez la commande delete-object.

Console

Pour annuler une tâche d'importation d'instantané

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Exports in Amazon S3 (Exportations dans Amazon S3).

   Les exportations d'instantanés de la base de données sont indiquées dans la colonne Source type (Type de source). L'état de l'exportation est affiché dans la colonne Status (État).
3. Choisissez la tâche d'exportation d'instantané que vous souhaitez annuler.
4. Choisissez Cancel (Annuler).
5. Choisissez Cancel export task (Annuler la tâche d'exportation) sur la page de confirmation.

AWS CLI

Pour annuler une tâche d'exportation d'instantané à l'aide de l'AWS CLI, utilisez la commande cancel-export-task. La commande requiert l'option `--export-task-identifier`.

Example

```
aws rds cancel-export-task --export-task-identifier my_export
{
    "Status": "CANCELING",
    "S3Prefix": "",
    "ExportTime": "2019-08-12T01:23:53.109Z",
    "S3Bucket": "amzn-s3-demo-bucket",
    "PercentProgress": 0,
    "KmsKeyId": "arn:aws:kms:AWS_Region:123456789012:key/K7MDENG/bPxRfiCYEXAMPLEKEY",
    "ExportTaskIdentifier": "my_export",
    "IamRoleArn": "arn:aws:iam::123456789012:role/export-to-s3",
    "TotalExtractedDataInGB": 0,
    "TaskStartTime": "2019-11-13T19:46:00.173Z",
    "SourceArn": "arn:aws:rds:AWS_Region:123456789012:snapshot:export-example-1"
```

```
}
```

API RDS

Pour annuler une tâche d'exportation d'instantané à l'aide de l'API Amazon RDS, utilisez l'opération CancelExportTask avec le paramètre `ExportTaskIdentifier`.

Performances d'exportation dans Aurora MySQL

Les instantanés de cluster de bases de données MySQL version 2 et version 3 utilisent un mécanisme d'exportation avancé pour améliorer les performances et réduire le temps d'exportation. Le mécanisme comprend des optimisations telles que les threads d'exportation multiples et la requête parallèle Aurora MySQL pour tirer parti de l'architecture de stockage partagé Aurora. Les optimisations sont appliquées de manière adaptative, en fonction de la taille et de la structure de l'ensemble des données.

Vous n'avez pas besoin d'activer la requête parallèle pour utiliser le processus d'exportation plus rapide, mais ce processus présente les mêmes limites que la requête parallèle. En outre, certaines valeurs de données ne sont pas prises en charge, comme les dates pour lesquelles le jour du mois est `0` ou l'année est `0000`. Pour plus d'informations, consultez Requêtes parallèles pour Amazon Aurora MySQL.

Lorsque des optimisations de performances sont appliquées, vous pouvez également voir des fichiers Parquet beaucoup plus grands (environ 200 Go) pour les exportations Aurora MySQL version 2 et 3.

Si le processus d'exportation plus rapide ne peut être utilisé, par exemple en raison de l'incompatibilité des types de données ou des valeurs, Aurora passe automatiquement à un mode d'exportation à thread unique sans requête parallèle. Selon le processus utilisé et la quantité de données à exporter, les performances d'exportation peuvent varier.

Résolution des problèmes d'exportation d'instantanés

Consultez les sections suivantes afin de résoudre les messages d'échec et les erreurs d'autorisation PostgreSQL dans le cadre des exportations de clusters de bases de données vers Amazon S3.

Messages d'échec relatifs aux tâches d'exportation Amazon S3

Le tableau suivant décrit les messages renvoyés en cas d'échec des tâches d'exportation Amazon S3.

Message d’échec	Description
Une erreur interne inconnue s’est produite.	La tâche a échoué en raison d’une erreur inconnue, d’une exception ou d’un échec.
Une erreur interne inconnue s’est produite lors de l’écriture des métadonnées de la tâche d’exporta tion dans le compartiment S3 [nom du compartiment].	La tâche a échoué en raison d’une erreur inconnue, d’une exception ou d’un échec.
L’exportation RDS n’a pas réussi à écrire les métadonnées de la tâche d’exportation, car elle ne peut pas assumer le rôle IAM [ARN du rôle].	La tâche d’exportation assume votre rôle IAM pour vérifier si elle est autorisée à écrire des métadonnées dans votre compartiment S3. Si la tâche ne peut pas assumer votre rôle IAM, elle échoue.
L’exportation RDS n’a pas réussi à écrire les métadonnées de la tâche d’exportation dans le compartiment S3 [nom du compartiment] à l’aide du rôle IAM [ARN du rôle] avec la clé KMS [ID de la clé]. Code d’erreur : [code d’erreur]	Une ou plusieurs autorisations sont manquantes et dès lors, la tâche d’exportation ne peut pas accéder au compartiment S3. Ce message d’échec est généré lors de la réception de l’un des codes d’erreur suivants : • `AWSSecurityTokenServiceException` avec le code d’erreur `AccessDenied` • `AmazonS3Exception` avec le code d’erreur `NoSuchBucket` , `AccessDenied` , `KMS.KMSInvalidStateException` , `403 Forbidden` ou `KMS.DisabledException` Ces codes d’erreur indiquent que les paramètres sont mal configurés pour le rôle IAM, le compartiment S3 ou la clé KMS.
Le rôle IAM [ARN du rôle] n’est pas autorisé à appeler [action S3] sur le compartiment S3 [nom du compartim ent]. Examinez vos autorisations et retentez l’exportation.	La politique IAM est mal configurée. L’autorisation pour l’action S3 spécifique sur le compartiment S3 est manquante, ce qui entraîne l’échec de la tâche d’exportation.

Message d’échec	Description
La vérification de la clé KMS a échoué. Vérifiez les informations d’identification de votre clé KMS et réessayez.	La vérification des informations d’identification de clé KMS a échoué.
La vérification des informations d’identification S3 a échoué. Vérifiez les autorisations de votre compartiment S3 et de la politique IAM.	La vérification des informations d’identification S3 a échoué.
Le compartiment S3 [nom du compartiment] n’est pas valide. Il n’est peut-être pas situé dans la Région AWS actuelle ou il n’existe pas. Vérifiez le nom de votre compartiment S3 et retentez l’exportation.	Le compartiment S3 n’est pas valide.
Le compartiment S3 [nom du compartiment] ne se trouve pas dans la Région AWS actuelle. Vérifiez le nom de votre compartiment S3 et retentez l’exportation.	Le compartiment S3 ne se trouve pas dans la bonne Région AWS.

Dépannage des erreurs d’autorisations PostgreSQL

Lors de l’exportation de bases de données PostgreSQL vers Amazon S3, vous pouvez voir une erreur `PERMISSIONS_DO_NOT_EXIST` indiquant que certaines tables ont été ignorées. Cette erreur se produit généralement lorsque le superutilisateur, que vous avez spécifié lors de la création de l’instance de base de données, n’a pas les autorisations nécessaires pour accéder à ces tables.

Pour corriger cette erreur, exécutez la commande suivante :

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA schema_name TO superuser_name
```

Pour plus d’informations sur les privilèges des superutilisateurs, consultez Privilèges du compte utilisateur principal.

Restauration d’un cluster de bases de données à une date définie

Vous pouvez restaurer un cluster de bases de données à un moment donné, et créer ainsi un cluster de bases de données.

Lorsque vous restaurez un cluster de bases de données à un moment donné, vous pouvez choisir le groupe de sécurité de cloud privé virtuel (VPC) par défaut. Vous pouvez également appliquer un groupe de sécurité VPC personnalisé à votre cluster de bases de données.

Les clusters de bases de données restaurés sont automatiquement associés au cluster et aux groupes de paramètres de base de données par défaut. Cependant, vous pouvez appliquer des groupes de paramètres personnalisés en les définissant lors d’une restauration.

Amazon Aurora charge en continu les enregistrements de journaux pour les clusters de bases de données vers Amazon S3. Pour connaître l'heure de restauration la plus récente pour un cluster de base de données, utilisez la AWS CLI describe-db-clusterscommande et examinez la valeur renvoyée dans le `LatestRestorableTime` champ correspondant au cluster de base de données.

Vous pouvez procéder à une restauration à n’importe quel moment dans le passé au cours de la période de rétention des sauvegardes. Pour connaître l'heure de restauration la plus proche pour un cluster de base de données, utilisez la AWS CLI describe-db-clusterscommande et examinez la valeur renvoyée dans le `EarliestRestorableTime` champ correspondant au cluster de base de données.

La période de conservation des sauvegardes du cluster de bases de données restauré est identique à celle du cluster de bases de données source.

> **Note**
>
> Les informations de cette rubrique s’appliquent à Amazon Aurora. Pour plus d’informations sur la restauration d’une instance de base de données Amazon RDS, consultez Restauration d’une instance de base de données à un instant spécifié.
> Pour plus d’informations sur la sauvegarde et la restauration d’un cluster de bases de données Aurora, consultez Présentation de la sauvegarde et de la restauration d’un cluster de bases de données Aurora.
> Pour Aurora MySQL, vous pouvez restaurer un cluster de bases de données alloué dans un cluster de bases de données Aurora Serverless. Pour de plus amples informations, veuillez consulter Restauration d’un cluster de bases de données Aurora Serverless v1.

Vous pouvez également l'utiliser AWS Backup pour gérer les sauvegardes des clusters de bases de données Amazon Aurora. Si votre cluster de base de données est associé à un plan de sauvegarde dans AWS Backup, ce plan de sauvegarde est utilisé pour la point-in-time restauration. Pour plus d'informations, consultez Restauration d’un cluster de bases de données à un instant spécifié en utilisant AWS Backup.

Pour plus d’informations sur la restauration d’un cluster de bases de données Aurora ou d’un cluster global avec une version de support étendu RDS, consultez Restauration d’un cluster de bases de données Aurora ou d’un cluster global avec le support étendu Amazon RDS.

Restaurez un cluster de base de données à une heure spécifiée à partir d'une sauvegarde automatique, d'une sauvegarde automatique conservée ou en utilisant AWS Backup.

Rubriques

- Restauration d’un cluster de bases de données à un instant dans le passé
- Restauration d’un cluster de bases de données à un instant déterminé à partir d’une sauvegarde automatique conservée
- Restauration d’un cluster de bases de données à un instant spécifié en utilisant AWS Backup

Restauration d’un cluster de bases de données à un instant dans le passé

Vous pouvez restaurer un cluster de base de données à un moment donné à l'aide de l' AWS Management Console API AWS CLI, de ou de l'API RDS.

Console

Pour restaurer un cluster de bases de données à un moment donné

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Automated backups (Sauvegardes automatisées).

   Les sauvegardes automatisées sont affichées dans l’onglet Current Region (Région actuelle).

3. Choisissez le cluster de bases de données à restaurer.
4. Sous Actions, sélectionnez Restaurer à un moment donné.

   La fenêtre Restaurer à un instant dans le passé s'affiche.
5. Choisissez Dernière heure de restauration possible pour restaurer à la dernière heure possible, ou choisissez Personnalisé pour choisir une heure.

   Si vous choisissez Custom (Personnalisé), saisissez la date et l'heure auxquelles vous souhaitez restaurer le cluster.

   > **Note**
   >
   > Les heures sont exprimées dans votre fuseau horaire local, qui est indiqué par son décalage par rapport à l'heure UTC. Par exemple, UTC-5 correspond à l' Time/Central heure avancée normale de l'Est.
6. Pour Identifiant du cluster de bases de données, saisissez le nom du cluster de bases de données restauré cible. Le nom doit être unique.
7. Choisissez d'autres options selon vos besoins, comme la classe d'instance de la base de données et la configuration du stockage du cluster de bases de données.

   Pour plus d'informations sur chaque paramètre, consultez Paramètres pour les clusters de bases de données Aurora.
8. Choisissez Restaurer à un instant dans le passé.

AWS CLI

Pour restaurer un cluster de base de données à une heure spécifiée, utilisez la AWS CLI commande restore-db-cluster-to- point-in-time pour créer un nouveau cluster de base de données.

Vous pouvez spécifier d'autres paramètres. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour les clusters de bases de données Aurora.

Si des balises sont fournies dans la demande, les balises fournies sont appliquées au cluster de base de données restauré. Si aucune balise n'est fournie dans la demande et si le cluster de base de données source est actif dans la région et possède des balises, Aurora ajoute les dernières balises du cluster de base de données source au cluster de base de données restauré.

Example

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-to-point-in-time \
       --source-db-cluster-identifier mysourcedbcluster \
       --db-cluster-identifier mytargetdbcluster \
       --restore-to-time 2017-10-14T23:45:00.000Z
```

Pour Windows :

```
aws rds restore-db-cluster-to-point-in-time ^
       --source-db-cluster-identifier mysourcedbcluster ^
       --db-cluster-identifier mytargetdbcluster ^
       --restore-to-time 2017-10-14T23:45:00.000Z
```

Important

Si vous utilisez la console pour restaurer un cluster de bases de données à un instant spécifié, Amazon RDS crée automatiquement l'instance principale (scripteur) pour votre cluster de bases de données. Si vous utilisez le AWS CLI pour restaurer un cluster de base de données à une heure spécifiée, vous devez créer explicitement l'instance principale pour votre cluster de base de données. L'instance principale est la première instance créée dans un cluster de bases de données.

Pour créer l'instance principale de votre cluster de base de données, appelez la create-db-instance AWS CLI commande. Incluez le nom du cluster de bases de données comme valeur de l'option `--db-cluster-identifier`.

API RDS

Pour restaurer un cluster de bases de données à une date spécifiée, appelez l'opération d'API Amazon RDS `RestoreDBClusterToPointInTime` avec les paramètres suivants :

- `SourceDBClusterIdentifier`

- `DBClusterIdentifier`
- `RestoreToTime`

> **Important**
>
> Si vous utilisez la console pour restaurer un cluster de bases de données à un instant spécifié, Amazon RDS crée automatiquement l'instance principale (scripteur) pour votre cluster de bases de données. Si vous utilisez l'API RDS pour restaurer un cluster de bases de données à un instant spécifié, assurez-vous de créer explicitement l'instance principale pour votre cluster de bases de données. L'instance principale est la première instance créée dans un cluster de bases de données.
>
> Pour créer l'instance principale de votre cluster de base de données, appelez l'opération Create DBInstance de l'API RDS. Incluez le nom du cluster de bases de données comme valeur de paramètre `DBClusterIdentifier`.

Restauration d'un cluster de bases de données à un instant déterminé à partir d'une sauvegarde automatique conservée

Vous pouvez restaurer un cluster de bases de données à partir d'une sauvegarde automatique conservée après avoir supprimé le cluster de bases de données source, si la sauvegarde se situe dans la période de conservation du cluster source. Le processus est similaire à la restauration d'un cluster de bases de données à partir d'une sauvegarde automatique.

> **Note**
>
> Vous ne pouvez pas restaurer un cluster de bases de données Aurora Serverless v1 à l'aide de cette procédure, car les sauvegardes automatiques des clusters Aurora Serverless v1 ne sont pas conservées.

Console

Pour restaurer un cluster de bases de données à un moment donné

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

2. Dans le panneau de navigation, choisissez Automated backups (Sauvegardes automatisées).
3. Choisissez l'onglet Rétention.

4. Choisissez le cluster de bases de données à restaurer.
5. Sous Actions, sélectionnez Restaurer à un moment donné.

   La fenêtre Restaurer à un instant dans le passé s'affiche.
6. Choisissez Dernière heure de restauration possible pour restaurer à la dernière heure possible, ou choisissez Personnalisé pour choisir une heure.

   Si vous choisissez Custom (Personnalisé), saisissez la date et l'heure auxquelles vous souhaitez restaurer le cluster.

   > **Note**
   >
   > Les heures sont exprimées dans votre fuseau horaire local, qui est indiqué par son décalage par rapport à l'heure UTC. Par exemple, UTC-5 est l'heure normale de l'Est/heure avancée du Centre.

7. Pour Identifiant du cluster de bases de données, saisissez le nom du cluster de bases de données restauré cible. Le nom doit être unique.
8. Choisissez d'autres options selon vos besoins, comme la classe d'instance de base de données.

   Pour plus d'informations sur chaque paramètre, consultez Paramètres pour les clusters de bases de données Aurora.
9. Choisissez Restaurer à un instant dans le passé.

AWS CLI

Pour restaurer un cluster de bases de données à un instant déterminé, utilisez la commande AWS CLI restore-db-cluster-to-point-in-time pour créer un nouveau cluster de bases de données.

Vous pouvez spécifier d'autres paramètres. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour les clusters de bases de données Aurora.

L'étiquetage des ressources est pris en charge pour cette opération. Lorsque vous utilisez l'option `--tags`, les identifications du cluster de bases de données source sont ignorées et celles qui sont fournies sont utilisées. Sinon, les dernières identifications du cluster source sont utilisées.

Example

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-resource-id cluster-123ABCEXAMPLE \
    --db-cluster-identifier mytargetdbcluster \
    --restore-to-time 2017-10-14T23:45:00.000Z
```

Pour Windows :

```
aws rds restore-db-cluster-to-point-in-time ^
    --source-db-cluster-resource-id cluster-123ABCEXAMPLE ^
    --db-cluster-identifier mytargetdbcluster ^
    --restore-to-time 2017-10-14T23:45:00.000Z
```

> **Important**
>
> Si vous utilisez la console pour restaurer un cluster de bases de données à un instant spécifié, Amazon RDS crée automatiquement l'instance principale (scripteur) pour votre cluster de bases de données. Si vous utilisez l'AWS CLI pour restaurer un cluster de bases de données à un instant spécifié, vous devez créer explicitement l'instance principale pour votre cluster de bases de données. L'instance principale est la première instance créée dans un cluster de bases de données.
>
> Appelez la commande create-db-instance de l'AWS CLI pour créer l'instance principale de votre cluster de bases de données. Incluez le nom du cluster de bases de données comme valeur de l'option `--db-cluster-identifier`.

API RDS

Pour restaurer un cluster de bases de données à une date spécifiée, appelez l'opération d'API Amazon RDS `RestoreDBClusterToPointInTime` avec les paramètres suivants :

- `SourceDbClusterResourceId`
- `DBClusterIdentifier`
- `RestoreToTime`

> **Important**
>
> Si vous utilisez la console pour restaurer un cluster de bases de données à un instant spécifié, Amazon RDS crée automatiquement l'instance principale (scripteur) pour votre cluster de bases de données. Si vous utilisez l'API RDS pour restaurer un cluster de bases de données à un instant spécifié, assurez-vous de créer explicitement l'instance principale pour votre cluster de bases de données. L'instance principale est la première instance créée dans un cluster de bases de données.
> Appelez l'opération d'API RDS CreateDBInstance pour créer l'instance principale pour votre cluster de bases de données. Incluez le nom du cluster de bases de données comme valeur de paramètre `DBClusterIdentifier`.

Restauration d'un cluster de bases de données à un instant spécifié en utilisant AWS Backup

Vous pouvez utiliser AWS Backup pour gérer vos sauvegardes automatiques, puis pour les restaurer à une date spécifiée. Pour ce faire, vous créez un plan de sauvegarde dans AWS Backup et assignez votre cluster de bases de données en tant que ressource. Vous activez ensuite les sauvegardes continues pour la récupération ponctuelle dans la règle de sauvegarde. Pour plus d'informations sur les plans et les règles de sauvegarde, consultez le Guide du développeur AWS Backup.

Activation des sauvegardes continues dans AWS Backup

Vous activez les sauvegardes continues dans les règles de sauvegarde.

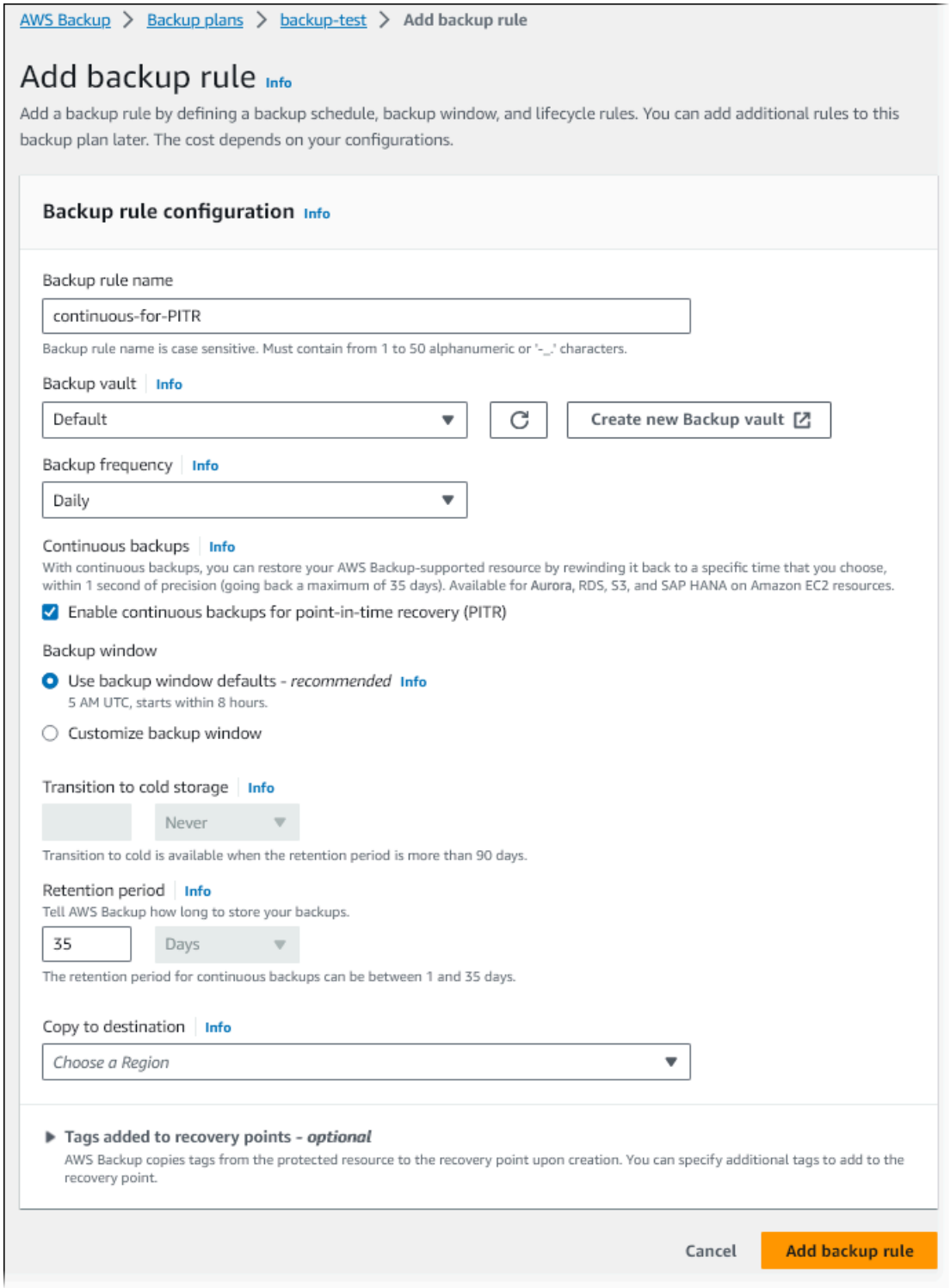
Pour activer les sauvegardes continues pour la récupération ponctuelle

1. Connectez-vous à la AWS Management Console et ouvrez la console AWS Backup à l'adresse https://console.aws.amazon.com/backup.
2. Dans le panneau de navigation, choisissez Backup plans (Plans de sauvegarde).
3. Sous Nom du plan de sauvegarde, sélectionnez le plan de sauvegarde que vous utilisez pour sauvegarder votre cluster de bases de données.

4. Sous la section Règles de sauvegarde, choisissez Ajouter une règle de sauvegarde.

   La page Ajouter une règle de sauvegarde apparaît.

5. Cochez la case Activer les sauvegardes continues pour la reprise ponctuelle (PITR).



6. Choisissez d’autres paramètres selon vos besoins, puis choisissez Ajouter la règle de sauvegarde.

Restauration à partir d’une sauvegarde continue dans AWS Backup

Vous effectuez une restauration à un instant spécifié à partir d’un coffre de sauvegarde.

Console

Vous pouvez utiliser la AWS Management Console pour restaurer un cluster de bases de données à un instant spécifié.

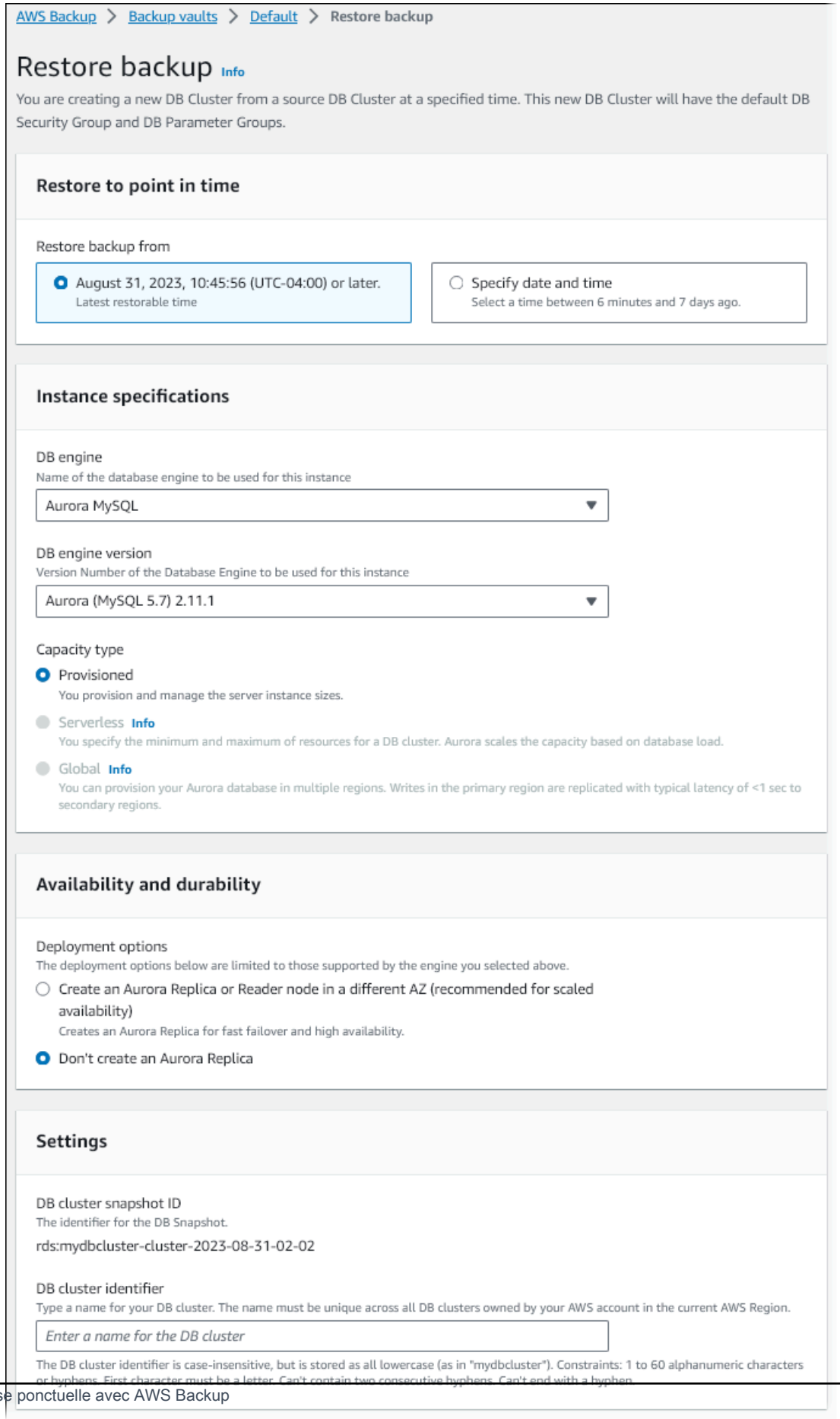
Pour restaurer à partir d’une sauvegarde continue dans AWS Backup

1. Connectez-vous à la AWS Management Console et ouvrez la console AWS Backup à l’adresse https://console.aws.amazon.com/backup.
2. Dans le panneau de navigation, choisissez Backup vaults (Coffres-forts de sauvegarde).
3. Choisissez le coffre de sauvegarde qui contient votre sauvegarde continue, par exemple Par défaut.

   La page de détails du coffre de sauvegarde apparaît.
4. Sous Points de restauration, sélectionnez le point de récupération pour la sauvegarde automatique.

   Il a un type de sauvegarde En continu et un nom avec `continuous:cluster-`*`AWS-Backup-job-number`*.
5. Pour Actions, choisissez Restaurer.

   La page Restaurer la sauvegarde apparaît.


AWS Backup > Backup vaults > Default > Restore backup
Restore backup Info
You are creating a new DB Cluster from a source DB Cluster at a specified time. This new DB Cluster will have the default DB Security Group and DB Parameter Groups.
Restore to point in time
Restore backup from
August 31, 2023, 10:45:56 (UTC-04:00) or later.
Latest restorable time
Specify date and time
Select a time between 6 minutes and 7 days ago.
Instance specifications
DB engine
Name of the database engine to be used for this instance
Aurora MySQL
DB engine version
Version Number of the Database Engine to be used for this instance
Aurora (MySQL 5.7) 2.11.1
Capacity type
Provisioned
You provision and manage the server instance sizes.
Serverless Info
You specify the minimum and maximum of resources for a DB cluster. Aurora scales the capacity based on database load.
Global Info
You can provision your Aurora database in multiple regions. Writes in the primary region are replicated with typical latency of <1 sec to secondary regions.
Availability and durability
Deployment options
The deployment options below are limited to those supported by the engine you selected above.
Create an Aurora Replica or Reader node in a different AZ (recommended for scaled availability)
Creates an Aurora Replica for fast failover and high availability.
Don't create an Aurora Replica
Settings
DB cluster snapshot ID
The identifier for the DB Snapshot.
rds:mydbcluster-cluster-2023-08-31-02-02
DB cluster identifier
Type a name for your DB cluster. The name must be unique across all DB clusters owned by your AWS account in the current AWS Region.
Enter a name for the DB cluster
The DB cluster identifier is case-insensitive, but is stored as all lowercase (as in "mydbcluster"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

6. Pour Restaurer à un moment donné, sélectionnez Spécifier la date et l'heure pour restaurer à un moment précis.
7. Choisissez les autres paramètres nécessaires pour restaurer le cluster de bases de données, puis choisissez Restaurer la sauvegarde.

   La page Tâches apparaît, montrant le volet Tâches de restauration. Un message en haut de la page fournit des informations sur la tâche de restauration.

Une fois le cluster de bases de données restauré, vous devez y ajouter l'instance de base de données principale (enregistreur). Appelez la commande create-db-instance de l'AWS CLI pour créer l'instance principale de votre cluster de bases de données. Incluez le nom du cluster de bases de données comme valeur de paramètre `--db-cluster-identifier`.

Interface de ligne de commande (CLI)

Vous utilisez la commande AWS CLI start-restore-job pour restaurer le cluster de bases de données à un instant spécifié. Les paramètres suivants sont obligatoires :

- `--recovery-point-arn` : Amazon Resource Name (ARN) du point de récupération à partir duquel effectuer la restauration.
- `--resource-type` : utilisez `Aurora`.
- `--iam-role-arn` : ARN du rôle IAM que vous utilisez pour les opérations AWS Backup.
- `--metadata` : métadonnées que vous utilisez pour restaurer le cluster de bases de données. Les paramètres suivants sont obligatoires :
  - `DBClusterIdentifier`
  - `Engine`
  - `RestoreToTime` ou `UseLatestRestorableTime`

L'exemple suivant montre comment restaurer un cluster de bases de données à un instant spécifié.

```
aws backup start-restore-job \
--recovery-point-arn arn:aws:backup:eu-central-1:123456789012:recovery-
point:continuous:cluster-itsreallyjustanexample1234567890-487278c2 \
--resource-type Aurora \
--iam-role-arn arn:aws:iam::123456789012:role/service-role/AWSBackupDefaultServiceRole
 \
```

```
--metadata '{"DBClusterIdentifier":"backup-pitr-test","Engine":"aurora-
mysql","RestoreToTime":"2023-09-01T17:00:00.000Z"}'
```

L'exemple suivant montre comment restaurer un cluster de bases de données à l'heure de restauration la plus récente.

```
aws backup start-restore-job \
--recovery-point-arn arn:aws:backup:eu-central-1:123456789012:recovery-
point:continuous:cluster-itsreallyjustanexample1234567890-487278c2 \
--resource-type Aurora \
--iam-role-arn arn:aws:iam::123456789012:role/service-role/AWSBackupDefaultServiceRole
 \
--metadata '{"DBClusterIdentifier":"backup-pitr-latest","Engine":"aurora-
mysql","UseLatestRestorableTime":"true"}'
```

Une fois le cluster de bases de données restauré, vous devez y ajouter l'instance de base de données principale (enregistreur). Appelez la commande create-db-instance de l'AWS CLI pour créer l'instance principale de votre cluster de bases de données. Incluez le nom du cluster de bases de données comme valeur de paramètre `--db-cluster-identifier`.

Suppression d’un instantané de cluster de bases de données

Vous pouvez supprimer des instantanés de cluster de bases de données gérés par Amazon RDS lorsque vous n’en avez plus besoin.

> **Note**
>
> Pour supprimer des sauvegardes gérées par AWS Backup, utilisez la console AWS Backup. Pour des informations sur AWS Backup, consultez le AWS Backupmanuel du développeur .

Suppression d’un instantané de cluster de bases de données

Vous pouvez supprimer un instantané de cluster de bases de données par l’intermédiaire de la console, de l’AWS CLI ou de l’API RDS.

Pour supprimer un instantané partagé ou public, vous devez vous connecter au compte AWS propriétaire de l’instantané.

Console

Pour supprimer un instantané de cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Snapshots.
3. Choisissez l’instantané de cluster de bases de données à supprimer.
4. Pour Actions, choisissez Delete snapshot (Supprimer la pile).
5. Dans la page de confirmation, sélectionnez Supprimer.

AWS CLI

Vous pouvez supprimer un instantané de cluster de bases de données en utilisant la commande delete-db-cluster-snapshot de l’AWS CLI.

Les options suivantes sont utilisées pour supprimer un instantané de cluster de bases de données.

- `--db-cluster-snapshot-identifier` – Identifiant de l’instantané de cluster de bases de données.

Example

Le code suivant supprime l'instantané de cluster de bases de données `mydbclustersnapshot`.

Pour Linux, macOS ou Unix :

```
aws rds delete-db-cluster-snapshot \
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

Pour Windows :

```
aws rds delete-db-cluster-snapshot ^
    --db-cluster-snapshot-identifier mydbclustersnapshot
```

API RDS

Vous pouvez supprimer un instantané de cluster de bases de données en utilisant l'opération d'API Amazon RDS DeleteDBClusterSnapshot.

Les paramètres suivants sont utilisés pour supprimer un instantané de cluster de bases de données.

- `DBClusterSnapshotIdentifier` – Identifiant de l'instantané de cluster de bases de données.

Tutoriel : restaurez un cluster de bases de données Amazon Aurora à partir d'un instantané de cluster de bases de données

Un scénario fréquent lors de l'utilisation d'Amazon Aurora consiste à avoir une instance de base de données que vous utilisez occasionnellement, mais dont vous n'avez pas besoin en permanence. Par exemple, vous pouvez utiliser un cluster de bases de données pour contenir les données d'un rapport que vous n'exécutez que tous les trimestres. Dans ce scénario, une manière d'économiser est de prendre un instantané du cluster de bases de données après la génération du rapport. Vous supprimez ensuite le cluster de bases de données et le restaurez lorsque vous devez charger de nouvelles données et exécuter le rapport au cours du trimestre suivant.

Lorsque vous restaurez un cluster de bases de données, vous fournissez le nom de l'instantané du cluster de bases de données à restaurer. Vous fournissez ensuite un nom pour le nouveau cluster de bases de données qui est créé à partir de l'opération de restauration. Pour plus d'informations sur la restauration de clusters de bases de données à partir d'instantanés, consultez Restauration à partir d'un instantané de cluster de bases de données.

Dans ce tutoriel, nous mettons également à niveau le cluster de bases de données restauré de la version 2 de Aurora MySQL (compatible avec MySQL 5.7) à la version 3 de Aurora MySQL (compatible avec MySQL 8.0).

Restauration d'un cluster de bases de données à un moment spécifié à partir d'un instantané de cluster de bases de données à l'aide de la console Amazon RDS ou de l'AWS CLI.

Pour plus d'informations sur la gestion des clés AWS KMS pour Amazon RDS, consultez Gestion AWS KMS key.

Rubriques

- Didacticiel : Restauration d'un cluster de bases de données à partir d'un instantané de cluster de bases de données à l'aide de la console Amazon RDS
- Didacticiel : Restauration d'un cluster de bases de données à partir d'un instantané de cluster de bases de données, à l'aide de l'AWS CLI

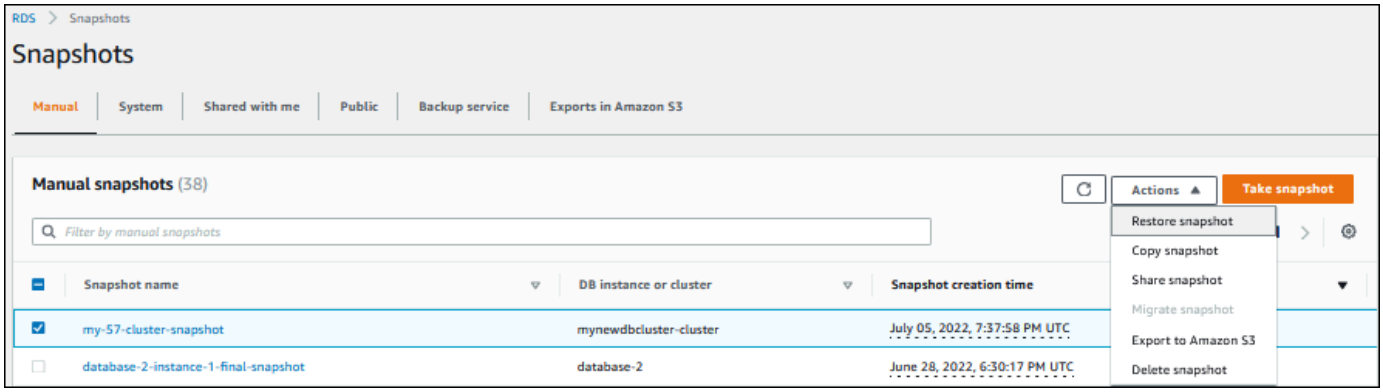
Didacticiel : Restauration d'un cluster de bases de données à partir d'un instantané de cluster de bases de données à l'aide de la console Amazon RDS

Dans ce didacticiel, vous restaurez un cluster de bases de données à partir d'un instantané de cluster de bases de données à l'aide de la console Amazon RDS. Lorsque vous restaurez un cluster de bases de données à partir d'un instantané à l'aide de la AWS Management Console, l'instance de base de données principale (en écriture) est également créée.

> **Note**
>
> Pendant la création de l'instance de base de données principale, elle apparaît comme une instance en lecture, mais après la création, elle devient une instance en écriture.

Pour restaurer un cluster DB à partir d'un instantané de cluster DB

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Snapshots.
3. Choisissez l'instantané de cluster de bases de données à partir duquel vous voulez restaurer.
4. Pour Actions, choisissez Restaurer l'instantané.

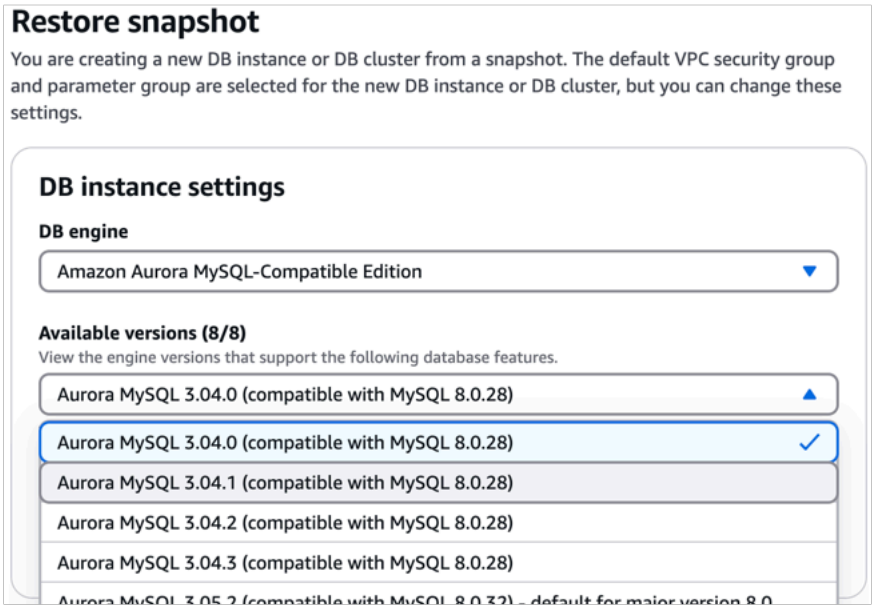


La page Restaurer l'instantané s'affiche.

5. Sous DB instance settings (Paramètres de l'instance de la base de données), procédez comme suit :

   a. Utilisez le paramètre par défaut pour DB engine (Moteur de la base de données).

b. Pour Versions disponibles, choisissez une version compatible MySQL–8.0, par exemple Aurora MySQL 3.04.0 (compatible avec MySQL 8.0.28).



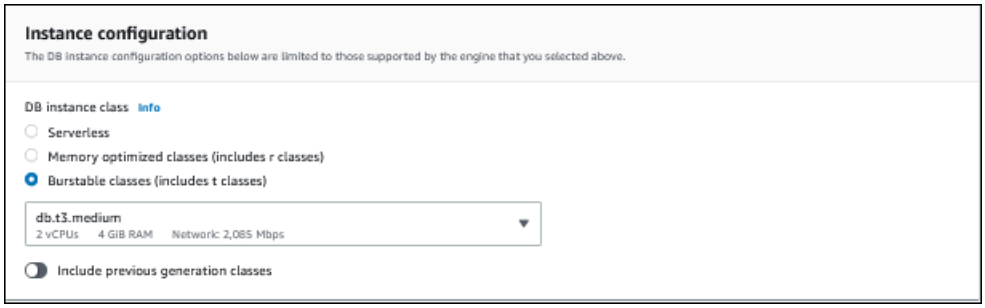
6. Sous Settings (Paramètres), pour DB instance identifier (Identifiant d'instance de base de données), saisissez le nom unique que vous voulez utiliser pour l'instance de base de données restaurée, par exemple **`my-80`**.

> **Note**
>
> Pour créer l'identifiant du cluster de bases de données, Amazon RDS ajoute `-cluster` à l'identifiant d'instance de base de données que vous spécifiez.

7. Sous Connectivity (Connectivité), utilisez les paramètres par défaut pour les éléments suivants :
   - Cloud privé virtuel (VPC)
   - Groupe de sous-réseaux de base de données
   - Accès public
   - VPC security group (firewall) [Groupe de sécurité VPC (pare-feu)]
8. Choisissez la classe d'instance de base de données.

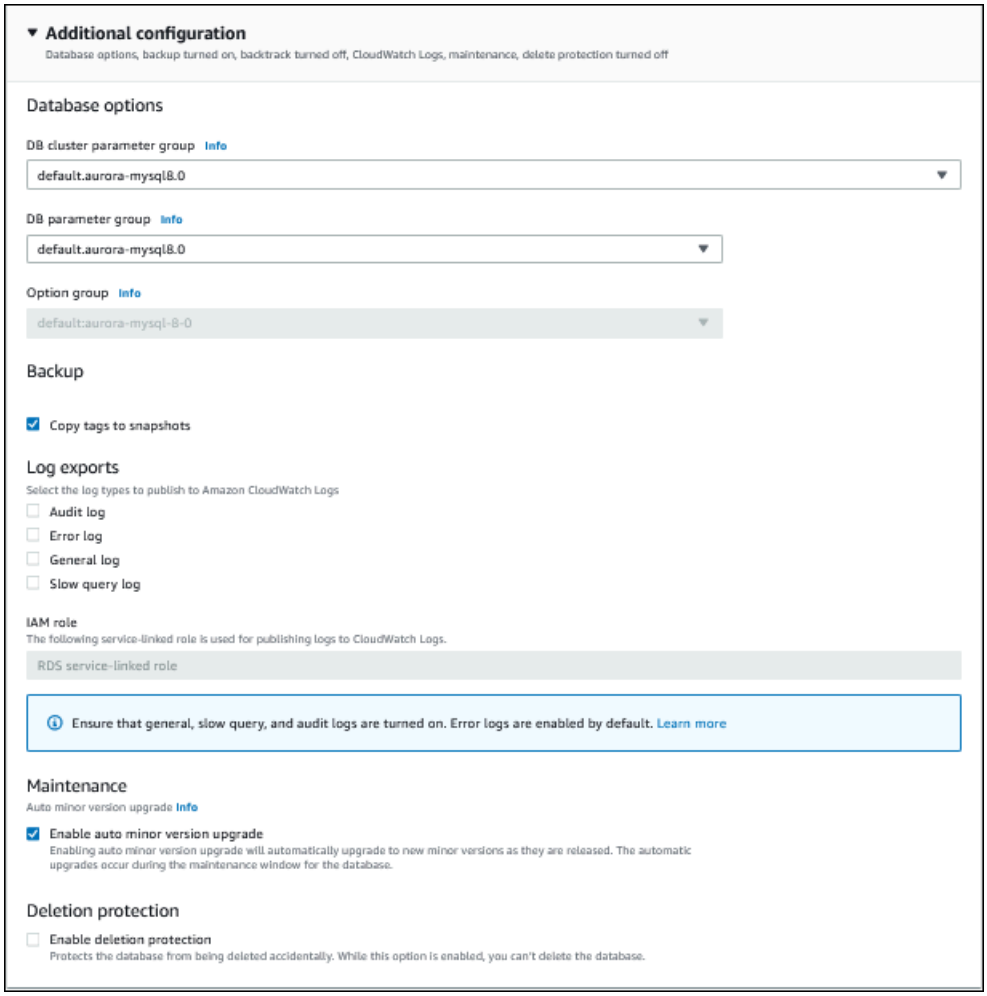
   Pour ce didacticiel, choisissez Burstable classes (includes t classes) [Classes à capacité extensible (inclut les classes t)], puis db.t3.medium.

> **Note**
>
> Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à la production. Pour plus de détails sur les classes d'instance T, consultez Types de classes d'instance de base de données.



9. Pour Database authentication (Authentification de la base de données), utilisez le paramètre par défaut.
10. Pour Encryption (Chiffrement), utilisez les paramètres par défaut.

    Si le cluster de bases de données source de l'instantané était chiffré, le cluster de bases de données restauré est également chiffré. Vous ne pouvez pas la rendre non chiffrée.
11. Développez Additional configuration (Configuration supplémentaire) en bas de la page.



12. Faites les choix suivants :

    a. Pour ce tutoriel, utilisez la valeur par défaut pour DB cluster parameter group (Groupe de paramètres de base de données).

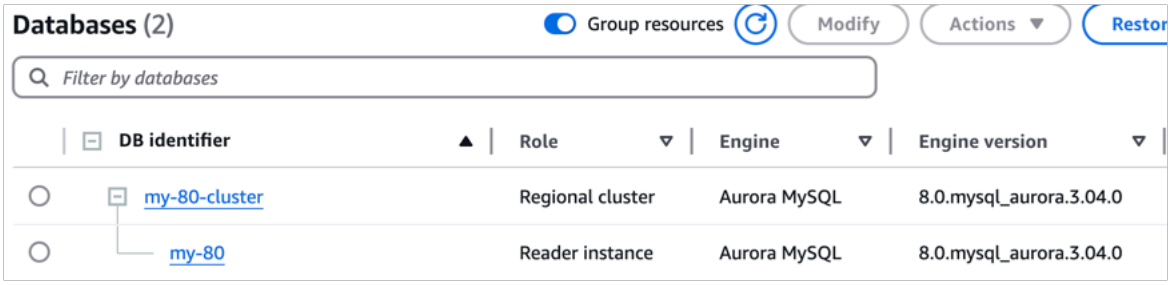
    b. Pour ce tutoriel, utilisez la valeur par défaut pour DB parameter group (Groupe de paramètres de base de données).

    c. Pour Log exports (Exportations de journaux), cochez toutes les cases.

    d. Pour Deletion protection (Protection contre la suppression), cochez la case Enable deletion protection (Activer la protection contre la suppression).

13. Choisissez Restore DB Instance (Restaurer une instance de base de données).

La page Bases de données affiche le cluster de bases de données restaurée, avec le statut `Creating`.



Pendant la création de l'instance de base de données principale, elle apparaît comme une instance en lecture, mais après la création, elle devient une instance en écriture.

Didacticiel : Restauration d'un cluster de bases de données à partir d'un instantané de cluster de bases de données, à l'aide de l'AWS CLI

Dans ce didacticiel, vous restaurez un cluster de bases de données à partir d'un instantané de clusters de bases de données, à l'aide de l'AWS CLI. Pour restaurer un cluster de bases de données à partir d'un instantané avec la AWS CLI, suivez les deux étapes suivantes :

1. Restauration du cluster de bases de données à l'aide de la commande restore-db-cluster-from-snapshot
2. Création de l'instance de base de données principale (écriture) à l'aide de la commande create-db-instance

Restauration du cluster de bases de données

Vous utilisez la commande `restore-db-cluster-from-snapshot`. Les options suivantes sont requises :

- `--db-cluster-identifier` : le nom du cluster de bases de données restauré.
- `--snapshot-identifier` : le nom de l'instantané de la base depuis lequel effectuer la restauration.
- `--engine` : le moteur de base de données du cluster de bases de données restauré. Il doit être compatible avec le moteur de base de données du cluster de bases de données source.

  Les choix sont les suivants :

  - `aurora-mysql` : Aurora compatible avec MySQL 5.7 et 8.0.
  - `aurora-postgresql` : compatible avec Aurora PostgreSQL.

Dans cet exemple, nous utilisons `aurora-mysql`.

- `--engine-version` : la version de la base de données restaurée. Dans cet exemple, nous utilisons une version compatible avec MySQL-8.0.

L'exemple suivant restaure un cluster de bases de données compatible avec Aurora MySQL 8.0 nommé `my-new-80-cluster` à partir d'un instantané de cluster de bases de données nommé `my-57-cluster-snapshot`.

Pour restaurer le cluster de bases de données

- Utilisez l'une des commandes suivantes.

  Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier my-new-80-cluster \
    --snapshot-identifier my-57-cluster-snapshot \
    --engine aurora-mysql \
    --engine-version 8.0.mysql_aurora.3.02.0
```

  Pour Windows :

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier my-new-80-cluster ^
    --snapshot-identifier my-57-cluster-snapshot ^
    --engine aurora-mysql ^
    --engine-version 8.0.mysql_aurora.3.02.0
```

La sortie se présente comme suit :

```
{
    "DBCluster": {
        "AllocatedStorage": 1,
        "AvailabilityZones": [
            "eu-central-1b",
            "eu-central-1c",
            "eu-central-1a"
        ],
        "BackupRetentionPeriod": 14,
```

```
        "DatabaseName": "",
        "DBClusterIdentifier": "my-new-80-cluster",
        "DBClusterParameterGroup": "default.aurora-mysql8.0",
        "DBSubnetGroup": "default",
        "Status": "creating",
        "Endpoint": "my-new-80-cluster.cluster-############.eu-
central-1.rds.amazonaws.com",
        "ReaderEndpoint": "my-new-80-cluster.cluster-ro-############.eu-
central-1.rds.amazonaws.com",
        "MultiAZ": false,
        "Engine": "aurora-mysql",
        "EngineVersion": "8.0.mysql_aurora.3.02.0",
        "Port": 3306,
        "MasterUsername": "admin",
        "PreferredBackupWindow": "01:55-02:25",
        "PreferredMaintenanceWindow": "thu:21:14-thu:21:44",
        "ReadReplicaIdentifiers": [],
        "DBClusterMembers": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-########",
                "Status": "active"
            }
        ],
        "HostedZoneId": "Z1RLNU0EXAMPLE",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:eu-central-1:123456789012:key/#######-5ccc-49cc-8aaa-
############",
        "DbClusterResourceId": "cluster-ZZ12345678ITSJUSTANEXAMPLE",
        "DBClusterArn": "arn:aws:rds:eu-central-1:123456789012:cluster:my-new-80-
cluster",
        "AssociatedRoles": [],
        "IAMDatabaseAuthenticationEnabled": false,
        "ClusterCreateTime": "2022-07-05T20:45:42.171000+00:00",
        "EngineMode": "provisioned",
        "DeletionProtection": false,
        "HttpEndpointEnabled": false,
        "CopyTagsToSnapshot": false,
        "CrossAccountClone": false,
        "DomainMemberships": [],
        "TagList": []
    }
}
```

Création de l’instance de base de données principale (écriture)

Pour créer l’instance de base de données principale (écriture), vous utilisez la commande `create-db-instance`. Les options suivantes sont requises :

- `--db-cluster-identifier` : le nom du cluster de bases de données restauré.
- `--db-instance-identifier` : le nom de l’instance principale de la base de données.
- `--db-instance-class` : la classe d’instance de l’instance principale de la base de données. Dans cet exemple, nous utilisons `db.t3.medium`.

  > **Note**
  >
  > Nous recommandons d’utiliser les classes d’instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d’autres serveurs non dédiés à la production. Pour plus de détails sur les classes d’instance T, consultez Types de classes d’instance de base de données.

- `--engine` : le moteur de base de données de l’instance de base de données principale. Il doit s’agir du même moteur de base de données que celui utilisé par le cluster de bases de données restauré.

  Les choix sont les suivants :

  - `aurora-mysql` : Aurora compatible avec MySQL 5.7 et 8.0.
  - `aurora-postgresql` : compatible avec Aurora PostgreSQL.

  Dans cet exemple, nous utilisons `aurora-mysql`.

L’exemple suivant crée une instance de base de données principale (écriture) nommée `my-new-80-cluster-instance` dans le cluster de bases de données compatible avec Aurora MySQL 8.0 restauré nommé `my-new-80-cluster`.

Pour créer l’instance de base de données principale

- Utilisez l’une des commandes suivantes.

  Pour Linux, macOS ou Unix :

```
aws rds create-db-instance \
```

```
    --db-cluster-identifier my-new-80-cluster \
    --db-instance-identifier my-new-80-cluster-instance \
    --db-instance-class db.t3.medium \
    --engine aurora-mysql
```

Pour Windows :

```
aws rds create-db-instance ^
    --db-cluster-identifier my-new-80-cluster ^
    --db-instance-identifier my-new-80-cluster-instance ^
    --db-instance-class db.t3.medium ^
    --engine aurora-mysql
```

La sortie se présente comme suit :

```
{
    "DBInstance": {
        "DBInstanceIdentifier": "my-new-80-cluster-instance",
        "DBInstanceClass": "db.t3.medium",
        "Engine": "aurora-mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "admin",
        "AllocatedStorage": 1,
        "PreferredBackupWindow": "01:55-02:25",
        "BackupRetentionPeriod": 14,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-########",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.aurora-mysql8.0",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-2305ca49",
```

```
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1a"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1b"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-########",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-central-1c"
                    },
                    "SubnetOutpost": {},
                    "SubnetStatus": "Active"
                }
            ]
        },
        "PreferredMaintenanceWindow": "sat:02:41-sat:03:11",
        "PendingModifiedValues": {},
        "MultiAZ": false,
        "EngineVersion": "8.0.mysql_aurora.3.02.0",
        "AutoMinorVersionUpgrade": true,
        "ReadReplicaDBInstanceIdentifiers": [],
        "LicenseModel": "general-public-license",
        "OptionGroupMemberships": [
            {
                "OptionGroupName": "default:aurora-mysql-8-0",
                "Status": "in-sync"
            }
        ],
        "PubliclyAccessible": false,
        "StorageType": "aurora",
        "DbInstancePort": 0,
```

```
        "DBClusterIdentifier": "my-new-80-cluster",
        "StorageEncrypted": true,
        "KmsKeyId": "arn:aws:kms:eu-central-1:534026745191:key/#######-5ccc-49cc-8aaa-
############",
        "DbiResourceId": "db-5C6UT5PU0YETANOTHEREXAMPLE",
        "CACertificateIdentifier": "rds-ca-2019",
        "DomainMemberships": [],
        "CopyTagsToSnapshot": false,
        "MonitoringInterval": 0,
        "PromotionTier": 1,
        "DBInstanceArn": "arn:aws:rds:eu-central-1:123456789012:db:my-new-80-cluster-
instance",
        "IAMDatabaseAuthenticationEnabled": false,
        "PerformanceInsightsEnabled": false,
        "DeletionProtection": false,
        "AssociatedRoles": [],
        "TagList": []
    }
}
```

Surveillance des métriques d’un cluster de bases de données Amazon Aurora

Amazon Aurora utilise un cluster de serveurs de base de données répliqués. La surveillance d’un cluster Aurora requiert habituellement de vérifier l’intégrité de différentes instances de base de données. Les instances peuvent avoir des rôles spécialisés, gérant principalement des opérations d’écriture, seulement des opérations de lecture, ou une combinaison des deux. Vous surveillez également l’intégrité globale du cluster en mesurant le décalage de réplication. Il s’agit de la durée pendant laquelle les modifications apportées par une instance de base de données doivent être disponibles pour les autres instances.

Rubriques

Plan de surveillance

Avant de commencer la surveillance de , créez un plan de surveillance. Ce plan doit répondre aux questions suivantes :

- Quels sont les objectifs de la surveillance ?
- Quelles sont les ressources à surveiller ?
- A quelle fréquence les ressources doivent-elles être surveillées ?
- Quels outils de surveillance utiliser ?
- Qui exécute les tâches de surveillance ?
- Qui doit être informé en cas de problème ?

Référence des performances

Pour atteindre vos objectifs de surveillance, vous devez établir une référence. Pour ce faire, mesurez les performances dans différentes conditions de charge et à différents moments dans votre environnement Amazon Aurora. Vous pouvez surveiller les métriques suivantes :

- Débit réseau
- Connexions client
- I/O pour les opérations de lecture, d'écriture ou de métadonnées
- Soldes de crédit en rafales pour vos instances de base de données

Nous vous recommandons de stocker les données de performances historiques pour Amazon Aurora. À l'aide des données stockées, vous pouvez comparer les performances actuelles aux tendances passées. Vous pouvez également faire la distinction entre les modèles normaux de performances et les anomalies, puis concevoir des techniques pour résoudre les problèmes.

Instructions sur les performances

En général, les valeurs acceptables pour les métriques de performances dépendent de l'activité de votre application par rapport à votre référence. Enquêtez sur les écarts cohérents ou tendanciels de vos données de référence. Les métriques suivantes sont souvent à l'origine des problèmes de performances :

- Forte utilisation de l'UC et de la RAM – Des valeurs importantes de l'utilisation de l'UC et de la RAM peuvent être appropriées, si elles sont conformes aux objectifs pour votre application (comme le débit ou la simultanéité).
- Utilisation de l'espace disque – Enquêtez sur l'utilisation de l'espace disque si l'espace utilisé est constamment égal ou supérieur à 85 pour cent de l'espace disque total. Voyez s'il est possible

de supprimer des données de l’instance ou d’archiver des données sur un système différent pour libérer de l’espace.

- Trafic réseau – Pour le trafic réseau, discutez avec votre administrateur pour connaître le débit attendu pour votre domaine réseau et votre connexion Internet. Enquêtez sur le trafic réseau si le débit est constamment inférieur à vos attentes.
- Connexions de la base de données – Envisagez de limiter les connexions de la base de données si vous constatez un nombre important de connexions utilisateur en même temps qu’une baisse des performances de l’instance et des temps de réponse. Le bon nombre de connexions utilisateur pour votre instance de base de données dépend de votre classe d’instance et de la complexité des opérations exécutées. Pour déterminer le nombre de connexions de la base de données, associez votre instance de base de données à un groupe de paramètres dans lequel le paramètre `User Connections` est configuré sur une autre valeur que 0 (illimité). Vous pouvez utiliser un groupe de paramètres existant ou en créer un nouveau. Pour plus d’informations, consultez Groupes de paramètres pour Amazon Aurora.
- Métriques IOPS – Les valeurs attendues pour les métriques d’IOPS dépendent de la spécification du disque et de la configuration du serveur, donc utilisez vos données de référence pour connaître les caractéristiques typiques. Enquêtez si les valeurs sont constamment différentes de vos données de référence. Pour de meilleures performances IOPS, veillez à ce que votre ensemble de travail typique puisse être chargé en mémoire pour minimiser les opérations de lecture et écriture.

Lorsque les performances se situent en dehors de votre de base établie, vous devrez peut-être apporter des modifications pour optimiser la disponibilité de votre base de données pour votre charge de travail. Par exemple, vous devrez peut-être modifier la classe d’instance de votre instance de base de données. Ou encore, modifier le nombre d’instances de base de données et de réplicas en lecture disponibles pour les clients.

Surveillance des outils d’Amazon Aurora

La surveillance constitue une part importante de la gestion de la fiabilité, de la disponibilité et des performances d’Amazon Aurora et de vos autres solutions AWS. AWS fournit des outils de surveillance suivants pour surveiller Amazon Aurora, signaler les problèmes et prendre des mesures automatiques, le cas échéant.

Rubriques

Outils de surveillance automatique

Nous vous recommandons d’automatiser le plus possible les tâches de supervision.

Rubriques

État et recommandations de du cluster Amazon Aurora

Vous pouvez utiliser les outils automatiques suivants pour surveiller Amazon Aurora et signaler un problème éventuel :

- État du cluster Amazon Aurora : afficher les détails de l’état actuel de votre cluster à l’aide de la console Amazon RDS, de l’AWS CLI ou de l’API RDS.
- Recommandations de Amazon Aurora — Consultez les recommandations automatisées pour les ressources de base de données, telles que les instances de base de données, les clusters de bases de données, et les groupes de paramètres de cluster de bases de données. Pour plus d’informations, consultez Recommandations d’Amazon Aurora.

Métriques Amazon CloudWatch pour Amazon Aurora

Amazon Aurora s'intègre à Amazon CloudWatch pour offrir des fonctionnalités de surveillance complémentaires.

- Amazon CloudWatch – Ce service surveille vos ressources AWS et les applications que vous exécutez sur AWS en temps réel. Vous pouvez utiliser les fonctions Amazon CloudWatch suivantes avec Amazon Aurora :
  - Métriques Amazon CloudWatch – Amazon Aurora envoient automatiquement chaque minute les métriques à CloudWatch pour chaque base de données active. Vous n'aurez pas de frais supplémentaires à régler pour les métriques Amazon RDS dans CloudWatch. Pour plus d'informations, consultez CloudWatch Métriques Amazon pour Amazon Aurora
  - Alarmes Amazon CloudWatch – Vous pouvez regarder une seule métrique Amazon Aurora sur une période donnée. Vous pouvez ensuite effectuer une ou plusieurs actions en fonction de la valeur de la métrique selon le seuil que vous définissez.

Analyse des performances d'Amazon RDS et surveillance du système d'exploitation

Vous pouvez utiliser les outils automatisés suivants pour surveiller les performances d'Amazon Aurora :

- Amazon RDS Performance Insights : pour évaluer rapidement la charge sur votre base de données et déterminer où et quand prendre des mesures. Pour plus d'informations, consultez Surveillance de la charge de la base de données avec Performance Insights sur .
- Surveillance améliorée Amazon RDS : consultez les métriques en temps réel pour le système d'exploitation. Pour plus d'informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.

Services intégrés

Les services AWS suivants sont intégrés à Amazon Aurora :

- Amazon EventBridge est un service de bus d'événements sans serveur qui permet de connecter facilement vos applications à des données provenant de diverses sources. Pour plus d'informations, consultez Surveillance des événements Amazon Aurora.

- Amazon CloudWatch Logs vous permet de surveiller, de stocker et d'accéder à vos fichiers journaux provenant des instances Amazon Aurora, de CloudTrail et d'autres sources. Pour plus d'informations, consultez Surveillance des fichiers journaux Amazon Aurora.
- AWS CloudTrail capture les appels d'API et les événements associés créés par votre Compte AWS ou au nom de celui-ci et livre les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Pour plus d'informations, consultez Surveillance des appels d'API Amazon Aurora dansAWS CloudTrail.
- Database Activity Streams (Flux d'activité de base de données) est une fonction Amazon Aurora qui fournit un flux en temps quasi réel de l'activité dans votre cluster de base de données . Pour plus d'informations, consultez Surveillance d'Amazon Aurora à l'aide des flux d'activité de base de données.
- Amazon DevOps Guru pour RDS est une fonctionnalité d'Amazon DevOps Guru qui applique le machine learning aux métriques de Performance Insights pour les bases de données Amazon Aurora. Pour plus d'informations, consultez Analyse des anomalies de performance d'Aurora avec Amazon DevOps Guru pour Amazon RDS.

Outils de surveillance manuelle

Vous devez surveiller manuellement les éléments que les CloudWatch alarmes ne couvrent pas. Les tableaux de bord Amazon RDS, CloudWatch, AWS Trusted Advisor et autres tableaux de bord de console AWS donnent un aperçu de l'état de votre environnement AWS. Nous recommandons de consulter également les fichiers journaux sur votre instance de base de données.

- À partir de la console Amazon RDS, vous pouvez surveiller les éléments suivants pour vos ressources :
  - Nombre de connexions à une instance de base de données
  - Quantité d'opérations de lecture et d'écriture à une instance de base de données
  - Volume de stockage en cours d'utilisation par une instance de base de données
  - Quantité de mémoire et d'UC utilisée pour une instance de base de données
  - Quantité de trafic réseau en direction et à partir d'une instance de base de données
- A partir du tableau de bord Trusted Advisor, vous pouvez vérifier les amélioration dans les domaines de l'optimisation des coûts, de la sécurité, de la tolérance aux pannes et des performances :
  - Instances de base de données Amazon RDS inactives

  - Risque lié à l’accès aux groupes de sécurité Amazon RDS
  - Sauvegardes Amazon RDS
  - Multi-AZ Amazon RDS
  - Aurora Accessibilité d’instance de base de données

  Pour plus d’informations sur ces vérifications, consultez Bonnes pratiques Trusted Advisor (Checks).
- La page d’accueil CloudWatch présente :
  - Alarmes et statuts en cours
  - Graphiques des alarmes et des ressources
  - Statut d’intégrité du service

  De plus, vous pouvez utiliser CloudWatch pour effectuer les tâches suivantes :
  - Créer des tableaux de bord personnalisés pour surveiller les services qui vous intéressent.
  - Données de métriques de graphiques pour résoudre les problèmes et découvrir les tendances.
  - Rechercher et parcourir toutes vos métriques de ressources AWS.
  - Créer et modifier des alarmes pour être informé des problèmes.

Affichage du statut du cluster

En utilisant la console Amazon RDS, vous pouvez accéder rapidement au statut du cluster de bases de données.

Rubriques

Affichage d’un cluster de bases de données Amazon Aurora

Vous disposez de plusieurs options pour afficher des informations sur vos clusters de bases de données Amazon Aurora et les instances de base de données dans vos clusters de bases de données.

- Vous pouvez afficher des clusters de bases de données et des instances de base de données dans la console Amazon RDS en choisissant Bases de données dans le panneau de navigation.
- Vous pouvez obtenir des informations sur le cluster de bases de données et les instances de base de données à l'aide du AWS Command Line Interface (AWS CLI).
- Vous pouvez obtenir des informations sur le cluster de bases de données et l’instance de base de données à l’aide de l’API Amazon RDS.

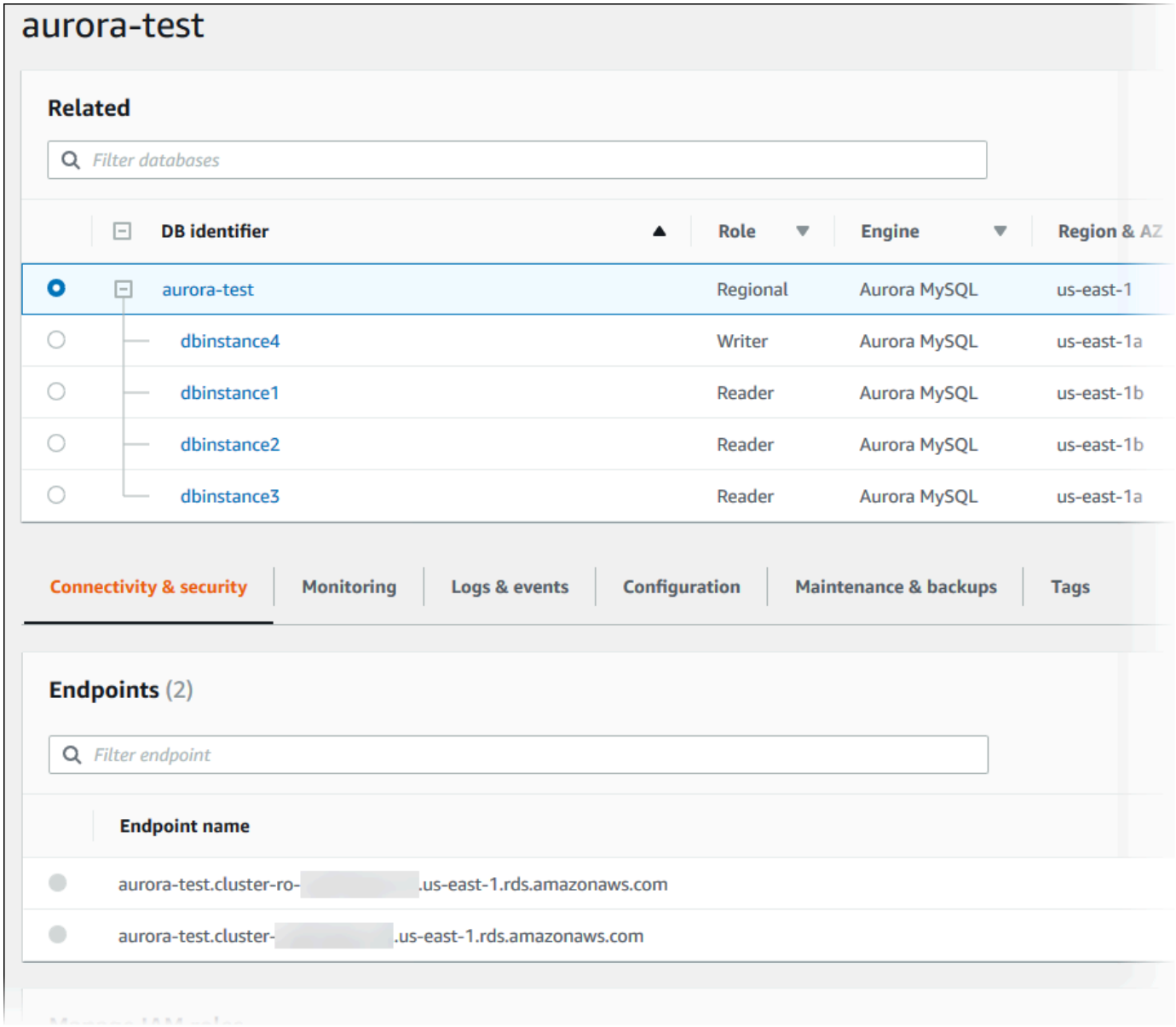
Console

Dans la console Amazon RDS, vous pouvez afficher les détails d’un cluster de bases de données en choisissant Bases de données dans le panneau de navigation de la console. Vous pouvez également afficher les détails des instances de base de données qui sont membres d’un cluster de bases de données Amazon Aurora.

Pour afficher ou modifier les clusters de bases de données dans la console Amazon RDS

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Bases de données.
3. Choisissez dans la liste le nom du cluster de bases de données Aurora que vous souhaitez visualiser.

   Par exemple, l’image suivante affiche la page de détails pour le cluster DB nommé `aurora-test`. Le cluster de bases de données a quatre instance affichées dans la liste des identifiants de bases de données. L’instance de base de données d’enregistreur, `dbinstance4`, est l’instance principale du cluster de bases de données.



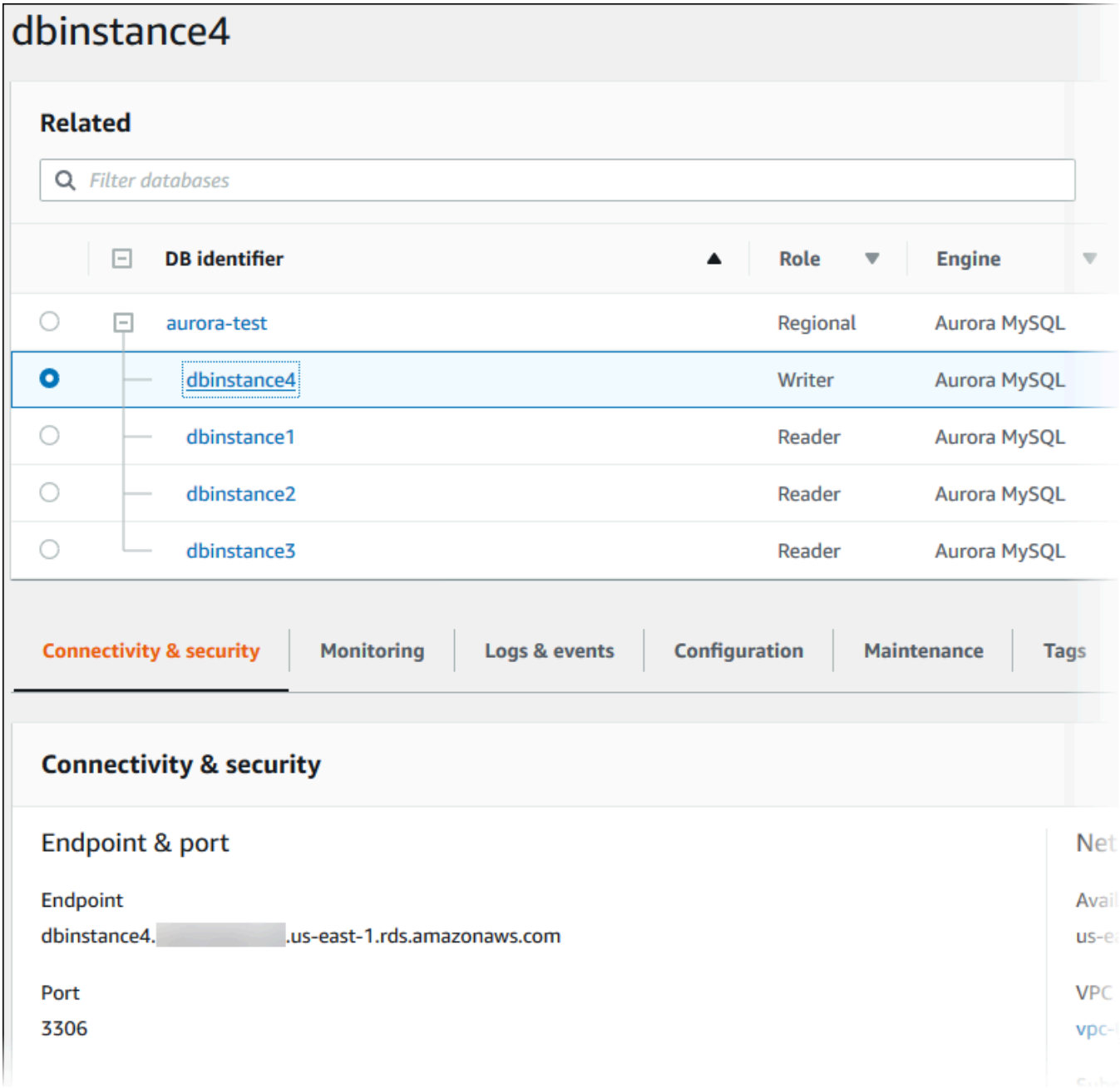
4. Pour modifier un cluster de bases de données, choisissez ce dernier dans la liste, puis choisissez Modify (Modifier).

Pour afficher ou modifier les instances d'un cluster de bases de données dans la console Amazon RDS

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Bases de données.
3. Effectuez l'une des actions suivantes :

- Pour afficher une instance de base de données, choisissez-en une dans la liste qui est membre du cluster de bases de données Aurora.

  Par exemple, si vous choisissez l'identifiant de l'instance de base de données `dbinstance4`, la console affiche la page de détails de l'instance de base de données `dbinstance4`, comme le montre l'image suivante.



- Pour modifier une instance de base de données, choisissez l'instance de base de données dans la liste et sélectionnez Modify (Modifier). Pour plus d'informations sur la modification d'un cluster, consultez Modification d'un cluster de bases de données Amazon Aurora.

AWS CLI

Pour afficher les informations du cluster de base de données à l'aide de AWS CLI, utilisez la describe-db-clusterscommande. Par exemple, la AWS CLI commande suivante répertorie les informations du cluster de base de données pour tous les clusters de base de données de la `us-east-1` région de modification pour le AWS compte configuré.

```
aws rds describe-db-clusters --region us-east-1
```

La commande renvoie le résultat suivant si vous êtes AWS CLI configuré pour une sortie JSON.

```
{
    "DBClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "Endpoint": "sample-cluster1.cluster-123456789012.us-
east-1.rds.amazonaws.com"
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "sample-cluster1",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2023-03-30T03:35:42.563Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-replica"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "sample-primary"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "03:34-04:04",
            "VpcSecurityGroups": [
```

```
                {
                    "Status": "active",
                    "VpcSecurityGroupId": "sg-ddb65fec"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.7.mysql_aurora.2.11.0",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2023-03-31T20:06:08.903Z",
            "PreferredMaintenanceWindow": "wed:08:15-wed:08:45"
        },
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "Endpoint": "aurora-sample.cluster-123456789012.us-
east-1.rds.amazonaws.com",
            "AllocatedStorage": 1,
            "DBClusterIdentifier": "aurora-sample-cluster",
            "MasterUsername": "mymasteruser",
            "EarliestRestorableTime": "2023-03-30T10:21:34.826Z",
            "DBClusterMembers": [
                {
                    "IsClusterWriter": false,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-replica-sample"
                },
                {
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "DBInstanceIdentifier": "aurora-sample"
                }
            ],
            "Port": 3306,
            "PreferredBackupWindow": "10:20-10:50",
            "VpcSecurityGroups": [
                {
```

```
                    "Status": "active",
                    "VpcSecurityGroupId": "sg-55da224b"
                }
            ],
            "DBSubnetGroup": "default",
            "StorageEncrypted": false,
            "DatabaseName": "sample",
            "EngineVersion": "5.7.mysql_aurora.2.11.0",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "BackupRetentionPeriod": 1,
            "AvailabilityZones": [
                "us-east-1b",
                "us-east-1c",
                "us-east-1d"
            ],
            "LatestRestorableTime": "2023-03-31T20:00:11.491Z",
            "PreferredMaintenanceWindow": "sun:03:53-sun:04:23"
        }
    ]
}
```

API RDS

Pour afficher les informations du cluster de bases de données à l'aide de l'API Amazon RDS, utilisez l'DBClustersopération Describe.

Affichage du statut du cluster de bases de données

L'état d'un cluster de base de données indique son état de fonctionnement actuel. Vous pouvez consulter l'état d'un cluster de base de données et des instances de cluster à l'aide de la console Amazon RDS, de l' AWS CLI API ou de l'API.

> **Note**
>
> Aurora utilise également un autre statut appelé état de la maintenance, qui est affiché dans la colonne Maintenance de la console Amazon RDS. Cette valeur indique l'état de tout correctif de maintenance qui doit être appliqué à un cluster de bases de données. L'état de la maintenance est indépendant du statut de cluster de bases de données. Pour plus d'informations sur le statut de la maintenance, consultez Application des mises à jour à un cluster de bases de données.

Recherchez les valeurs d'état possibles pour les clusters de bases de données dans le tableau suivant.

Statut du cluster de bases de données	Facturé	Description
Disponible	Facturé	Le cluster de base de données est disponible pour des modifications. Lorsqu'un cluster Aurora sans serveur est disponible et mis en pause, vous êtes facturé uniquement pour le stockage.
Backing-up	Facturé	Le cluster de bases de données est en cours de sauvegarde.
Backtracking	Facturé	Le cluster de bases de données est en cours de retour en arrière. Ce statut s'applique uniquement à Aurora MySQL.
Cloning-failed	Non facturé	Échec du clonage d'un cluster de bases de données

Statut du cluster de bases de données	Facturé	Description
Création	Non facturé	Le cluster de bases de données est en cours de création. Le cluster de bases de données n’est pas accessible pendant sa création.
Suppression en cours	Non facturé	Le cluster de bases de données est en cours de suppression.
Failing-over	Facturé	Un basculement à partir de l’instance principale vers un réplica Aurora est en cours.
Inaccessible-encryption-credentials	Non facturé	Le AWS KMS key fichier utilisé pour chiffrer ou déchiffrer le cluster de base de données n'est pas accessible ni récupéré.
Inaccessible-encryption-credentials-recoverable	Facturé pour stockage	La clé KMS utilisée pour chiffrer ou déchiffrer le cluster de bases de données n’est pas accessible. Cependant, si la clé KMS est active, le redémarrage du cluster de bases de données peut la récupérer. Pour plus d’informations, consultez Chiffrement d’un cluster de bases de données Amazon Aurora.
Maintenance	Facturé	Amazon RDS applique une mise à jour de maintenance au cluster de bases de données. Cet état est utilisé pour la maintenance de niveau de cluster de bases de données que RDS planifie suffisamment à l’avance.
Migrating	Facturé	Un instantané de cluster de bases de données est en cours de restauration à partir d’un cluster de bases de données.
Migration-failed	Non facturé	Échec d’une migration.

Statut du cluster de bases de données	Facturé	Description
Modification	Facturé	Le cluster de bases de données est en cours de modification en raison d'une demande du client de modification du cluster de bases de données.
Promoting	Facturé	Un réplica en lecture est en cours de promotion dans un cluster de bases de données autonome.
Preparing-data-migration	Facturé	Amazon RDS prépare la migration des données vers Aurora.
Renommage	Facturé	Le cluster de bases de données est train d'être renommée en raison d'une demande du client pour la renommer.
Resetting-master-credentials	Facturé	Les informations d'identification principales du cluster de bases de données sont en cours de réinitialisation en raison d'une demande du client pour les réinitialiser.
Démarrage en cours	Facturé pour stockage	Le cluster de bases de données démarre.
Arrêté(e)	Facturé pour stockage	Le cluster de bases de données est arrêté.
Arrêt en cours	Facturé pour stockage	Le cluster de bases de données s'arrête.

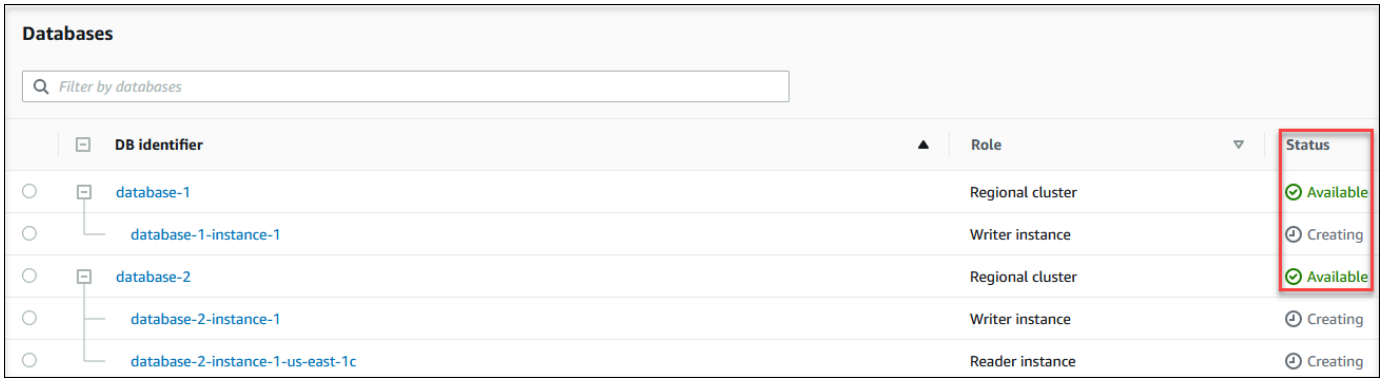
Statut du cluster de bases de données	Facturé	Description
Storage-optimization	Facturé	Votre instance de base de données est modifiée afin de changer la taille ou le type de stockage. L'instance de base de données est entièrement opérationnelle. Toutefois, tant que l'état de votre instance de base de données est storage-optimization, vous ne pouvez pas demander de modification au niveau de son stockage. Le processus d'optimisation du stockage est généralement court mais peut parfois prendre jusqu'à 24 heures ou même davantage.
Update-iam-db-auth	Facturé	L'autorisation IAM pour le cluster de bases de données est en cours de mise à jour.
Upgrading	Facturé	La version du moteur du cluster de bases de données ou du système d'exploitation est en cours de mise à niveau.
échec de la mise à niveau	Non facturé	Le cluster de base de données n'a pas pu être mis à niveau vers une version prise en charge. Aurora crée un instantané final avec le préfixe`erds-final` .

Console

Pour afficher le statut d'un cluster de bases de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Bases de données.

   La Databases page (page Bases de données) apparaît avec la liste des clusters de bases de données. Pour chaque cluster de bases de données, la valeur de statut est affichée.



INTERFACE DE LIGNE DE COMMANDE (CLI)

Pour afficher uniquement l'état des clusters de base de données, utilisez la requête suivante dans AWS CLI.

```
aws rds describe-db-clusters --query 'DBClusters[*].[DBClusterIdentifier,Status]' --
output table
```

Affichage du statut de l’instance de base de données dans un cluster Aurora

L'état d'une instance de base de données dans un cluster Aurora indique son état de fonctionnement actuel. Vous pouvez utiliser les procédures suivantes pour afficher l'état de l'instance de base de données d'un cluster dans la console Amazon RDS, dans la AWS CLI commande ou dans le fonctionnement de l'API.

> **Note**
>
> Amazon RDS utilise également un autre statut appelé état de la maintenance, qui est affiché dans la colonne Maintenance de la console Amazon RDS. Cette valeur indique l’état de tout correctif de maintenance qui doit être appliqué à une instance de base de données. L’état de la maintenance est indépendant du statut de l’instance de base de données. Pour plus d’informations sur le statut de la maintenance, consultez Application des mises à jour à un cluster de bases de données.

Recherchez les valeurs d’état possibles pour les instances de base de données dans le tableau suivant. Il indique également si vous serez facturé pour l’instance de base de données et son stockage, uniquement pour le stockage, ou si vous ne serez pas facturé. Pour tous les statuts d’instance de base de données, vous êtres toujours facturé pour l’utilisation de la sauvegarde.

Statut d’instance de bases de données	Facturé	Description
available	Facturé	L'instance de base de données est disponible pour des modifications.
backing-up	Facturé	L’instance de base de données est en cours de sauvegarde.
backtracking	Facturé	Un retour en arrière est en cours pour l’instance de base de données. Ce statut s’applique uniquement à Aurora MySQL.

Statut d’instance de bases de données	Facturé	Description
configuring-enhanced-monitoring	Facturé	La surveillance améliorée est en cours d’activation ou de désactivation pour cette instance de base de données.
configuring-iam-database-auth	Facturé	Gestion des identités et des accès AWS L'authentification de base de données (IAM) est activée ou désactivée pour cette instance de base de données.
configuring-log-exports	Facturé	La publication de fichiers CloudWatch journaux sur Amazon Logs est activée ou désactivée pour cette instance de base de données.
converting-to-vpc	Facturé	L’instance de base de données est en cours de conversion depuis une instance qui ne se trouve pas dans un Amazon Virtual Private Cloud (Amazon VPC) vers une instance qui est dans un Amazon VPC.
creating	Non facturé (non PITR) Facturé (PITR uniquement)	L’instance de base de données est en cours de création. L’instance de base de données n’est pas accessible pendant sa création. Si vous restaurez une base de données pendant la point-in-time restauration (PITR), vous êtes facturé alors que la base de données est en cours de création. C’est le seul scénario dans lequel l’état de création encourt des frais.
delete-precheck	Non facturé	Amazon RDS vérifie que les répliques de lecture peuvent être supprimées en toute sécurité.
deleting	Non facturé	L’instance de base de données est en cours de suppression.

Statut d’instance de bases de données	Facturé	Description
failed	Non facturé	L’instance de base de données a échoué et Amazon RDS ne peut pas la récupérer. Effectuez une point-in-time restauration à l'heure de restauration la plus récente de l'instance de base de données pour récupérer les données.
inaccessible-encryption-credentials	Non facturé	Le AWS KMS key fichier utilisé pour chiffrer ou déchiffrer l'instance de base de données n'est pas accessible ni récupéré.
inaccessible-encryption-credentials-recoverable	Facturé pour stockage	La clé KMS utilisée pour chiffrer ou déchiffrer l’instance de base de données n’est pas accessible. Toutefois, si la clé KMS est active, le redémarrage de l’instance de base de données peut la récupérer. Pour plus d’informations, consultez Chiffrement d’un cluster de bases de données Amazon Aurora.
incompatible-create	Non facturé	Amazon RDS tente de créer une instance de base de données, mais n’y parvient pas, car les ressources sont incompatibles avec votre instance de base de données. Cette situation peut se produire si, par exemple, le profil de votre instance de base de données ne dispose pas des autorisations appropriées.
incompatible-network	Non facturé	Amazon RDS tente d’effectuer une action de récupération sur une instance de base de données, mais n’y parvient pas, car l’état du VPC empêche l’exécution de l’action. Cette situation peut se produire si, par exemple, toutes les adresses IP disponibles d’un sous-réseau sont en cours d’utilisation et qu’Amazon RDS ne peut pas obtenir d’adresse IP pour l’instance de base de données.

Statut d’instance de bases de données	Facturé	Description
incompatible-option-group	Facturé	Amazon RDS a tenté d’appliquer une modification du groupe d’options, mais n’y parvient pas, et Amazon RDS ne peut pas rétablir l’état antérieur du groupe options. Pour plus d’informations, consultez la liste Événements récents de l’instance de base de données. Cette situation peut se produire si, par exemple, le groupe d’options contient une option telle que TDE et que l’instance de base de données ne comporte pas d’informations chiffrées.
incompatible-parameters	Facturé	Amazon RDS ne peut pas démarrer l’instance de base de données, car les paramètres spécifiés dans le groupe de paramètres de base de données de l’instance ne sont pas compatibles avec l’instance. Annulez les modifications des paramètres ou rendez-les compatibles avec l’instance de base de données pour rétablir l’accès à votre instance. Pour en savoir plus sur les paramètres incompatibles, consultez la liste Événements récents correspondant à l’instance de base de données.
incompatible-restore	Non facturé	Amazon RDS ne peut pas effectuer de point-in-time restauration. Parmi les causes courantes de ce statut figurent l’utilisation des tables temporaires ou des tables MyISAM avec MySQL.
insufficient-capacity	Non facturé	Amazon RDS ne peut pas créer votre instance, car la capacité disponible est insuffisante. Pour créer votre instance de base de données dans la même AZ avec le même type d’instance, supprimez votre instance de base de données, attendez quelques heures et essayez de la recréer. En variante, vous pouvez créer une nouvelle instance en utilisant une classe d’instances différente ou AZ.

Statut d’instance de bases de données	Facturé	Description
maintenance	Facturé	Amazon RDS applique une mise à jour de maintenance à l’instance base de données. Cet état est utilisé pour la maintenance de niveau d’instance que RDS planifie suffisamment à l’avance.
modifying	Facturé	L’instance de base de données est en cours de modification en raison d’une demande du client de modification de l’instance.
moving-to-vpc	Facturé	L’instance de base de données est en cours de transfert vers un nouveau Amazon Virtual Private Cloud (Amazon VPC).
rebooting	Facturé	L’instance de base de données est en cours de redémarrage en raison d’une demande du client ou d’un processus Amazon RDS nécessitant le redémarrage de l’instance.
resetting-master-credentials	Facturé	Les informations d’identification principales de l’instance de base de données sont en cours de réinitialisation en raison d’une demande du client pour les réinitialiser.
renaming	Facturé	L’instance de base de données est en cours de changement de nom en raison d’une demande du client pour la renommer.
restore-error	Facturé	L'instance de base de données a rencontré une erreur lors de la tentative de restauration depuis point-in-time ou vers un instantané.
starting	Facturé pour stockage	L’instance de base de données démarre.
stopped	Facturé pour stockage	L’instance de base de données est arrêtée.

Statut d’instance de bases de données	Facturé	Description
stopping	Facturé pour stockage	L’instance de base de données est en cours d’arrêt.
storage-config-upgrade	Facturé	La configuration du système de fichiers de stockage de l’instance de base de données est en cours de mise à niveau. Ce statut s’applique uniquement aux bases de données vertes dans le cadre d’un déploiement bleu/vert, ou aux réplicas en lecture d’instances de base de données.
storage-full	Facturé	L’instance de base de données a atteint sa capacité de stockage allouée. Son statut est critique et nous vous recommandons de corriger ce problème immédiatement. Pour cela, mettez votre stockage à l’échelle en modifiant l’instance de base de données. Pour éviter cette situation, configurez les CloudWatch alarmes Amazon pour qu'elles vous avertissent lorsque l'espace de stockage est insuffisant.
storage-initialization	Facturé	L’instance de base de données charge des blocs de données depuis Amazon S3 afin d’optimiser les performances du volume après avoir été restaurée à partir d’un instantané. Elle reste disponible pour les opérations, mais les performances risquent de ne pas être optimales tant que l’initialisation n’est pas terminée.

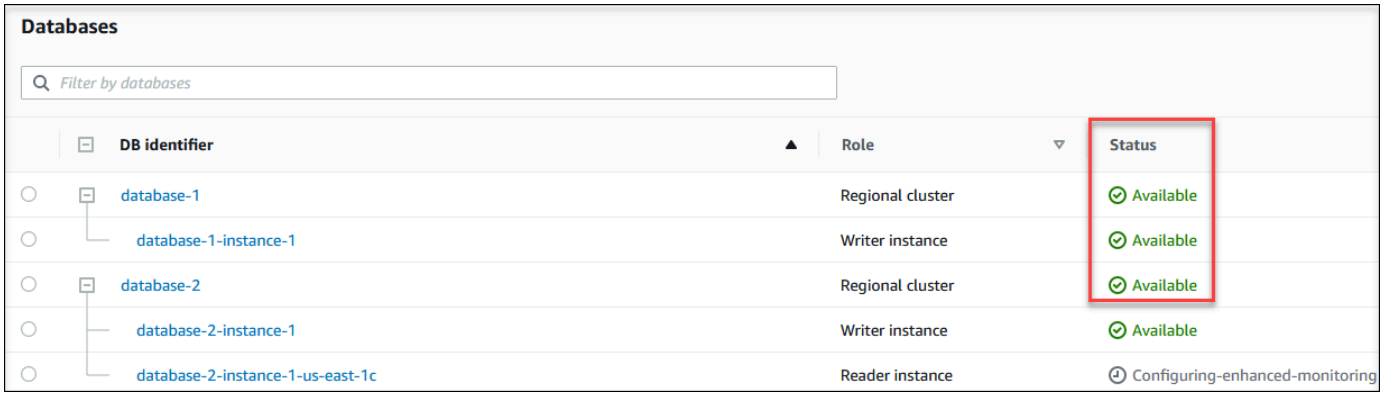
Statut d’instanc e de bases de données	Facturé	Description
storage-optimization	Facturé	Amazon RDS optimise le stockage de votre instance de base de données. Le processus d’optimisation du stockage est généralement court mais peut parfois prendre jusqu’à 24 heures ou même davantage. Pendant l’optimisation du stockage, l’instance de base de données reste disponible. L’optimisation du stockage est un processus d’arrière-plan qui n’affecte pas la disponibilité de l’instance.
upgrading	Facturé	La version du moteur de base de données ou du système d’exploitation est en cours de mise à niveau.
échec de la mise à niveau	Non facturé	L'instance de base de données n'a pas pu être mise à niveau vers une version prise en charge. Aurora crée un instantané final avec le préfixe`rds-final` .

Console

Pour afficher le statut d’une l’instance de base de données

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Bases de données.

   La Databases page (page Bases de données) apparaît avec la liste des instances de base de données. Pour chaque instance de la base de données dans un cluster, la valeur du statut est affichée.



INTERFACE DE LIGNE DE COMMANDE (CLI)

Pour afficher l'instance de base de données et ses informations d'état à l'aide de AWS CLI, utilisez la describe-db-instancescommande. Par exemple, la AWS CLI commande suivante répertorie toutes les informations relatives aux instances de base de données.

```
aws rds describe-db-instances
```

Pour afficher une instance de base de données spécifique et son état, appelez la describe-db-instancescommande avec l'option suivante :

- `DBInstanceIdentifier` – Nom de l'instance de base de données.

```
aws rds describe-db-instances --db-instance-identifier mydbinstance
```

Pour afficher uniquement le statut de toutes les instances de base de données, utilisez la requête suivante dans AWS CLI.

```
aws rds describe-db-instances --query 'DBInstances[*].
[DBInstanceIdentifier,DBInstanceStatus]' --output table
```

« Hello, World! »

Pour consulter l'état de l'instance de base de données à l'aide de l'API Amazon RDS, appelez l'DBInstancesopération Describe.

Recommandations d'Amazon Aurora

Amazon Aurora fournit des recommandations automatisées pour les ressources de base de données, telles que les instances de base de données, les clusters de bases de données, et les groupes de paramètres de base de données. Ces recommandations fournissent des conseils quand aux bonnes pratiques en analysant la configuration du cluster de bases de données, la configuration de l'instance de base de données, son utilisation et les données de performances.

L'Analyse des performances d'Amazon RDS surveille des métriques spécifiques et crée automatiquement des seuils en analysant les niveaux susceptibles de poser problème pour une ressource spécifique. Lorsque les nouvelles valeurs métriques dépassent un seuil prédéfini sur une période donnée, Performance Insights génère une recommandation proactive. Cette recommandation permet d'éviter tout impact futur sur les performances de la base de données. Par exemple, la recommandation « Transaction inactive » est générée pour les instances Aurora PostgreSQL lorsque des sessions connectées à la base de données ne réalisent aucune activité, mais peuvent bloquer certaines ressources de la base de données. Pour bénéficier des recommandations proactives, vous devez activer Performance Insights avec une période de conservation payante. Pour en savoir plus sur l'activation de Performance Insights, consultez Activation ou désactivation de l'Analyse des performances pour Aurora. Pour en savoir plus sur la tarification et la conservation des données pour Performance Insights, consultez Tarification et conservation des données pour Performance Insights.

DevOps Guru for RDS surveille certaines métriques afin de détecter lorsque leur comportement devient particulièrement inhabituel ou anormal. Ces anomalies sont signalées sous forme d'insights réactifs accompagnés de recommandations. Par exemple, DevOps Guru for RDS peut vous recommander d'augmenter la capacité de l'UC ou d'examiner les événements d'attente qui contribuent à la charge de base de données. DevOps Guru for RDS fournit également des recommandations proactives basées sur des seuils. Pour bénéficier de ces recommandations, vous devez activer DevOps Guru for RDS. Pour en savoir plus sur l'activation de DevOps Guru for RDS, consultez Activer DevOps Guru et spécifier la couverture des ressources.

Les recommandations peuvent se trouver dans l'un des états suivants : active, ignorée, en attente ou résolue. Les recommandations résolues sont disponibles pendant 365 jours.

Vous pouvez consulter ou ignorer les recommandations. Vous pouvez appliquer une recommandation active basée sur la configuration immédiatement, la planifier pour la prochaine fenêtre de maintenance ou l'ignorer. Dans le cas des recommandations proactives fondées sur des

seuils et des recommandations réactives basées sur le machine learning, il convient d'examiner la cause proposée du problème avant de mettre en œuvre les actions recommandées pour le résoudre.

Les recommandations sont prises en charge dans les Régions AWS suivantes :

- USA Est (Ohio)
- USA Est (Virginie du Nord)
- USA Ouest (Californie du Nord)
- USA Ouest (Oregon)
- Asie-Pacifique (Mumbai)
- Asie-Pacifique (Séoul)
- Asie-Pacifique (Singapour)
- Asie-Pacifique (Sydney)
- Asie-Pacifique (Tokyo)
- Canada (Centre)
- Europe (Francfort)
- Europe (Irlande)
- Europe (Londres)
- Europe (Paris)
- Europe (Stockholm)
- Amérique du Sud (São Paulo)

Découvrez comment consulter, appliquer, ignorer et modifier les recommandations d'Amazon Aurora dans les sections suivantes.

Rubriques

Affichage des recommandations Amazon Aurora

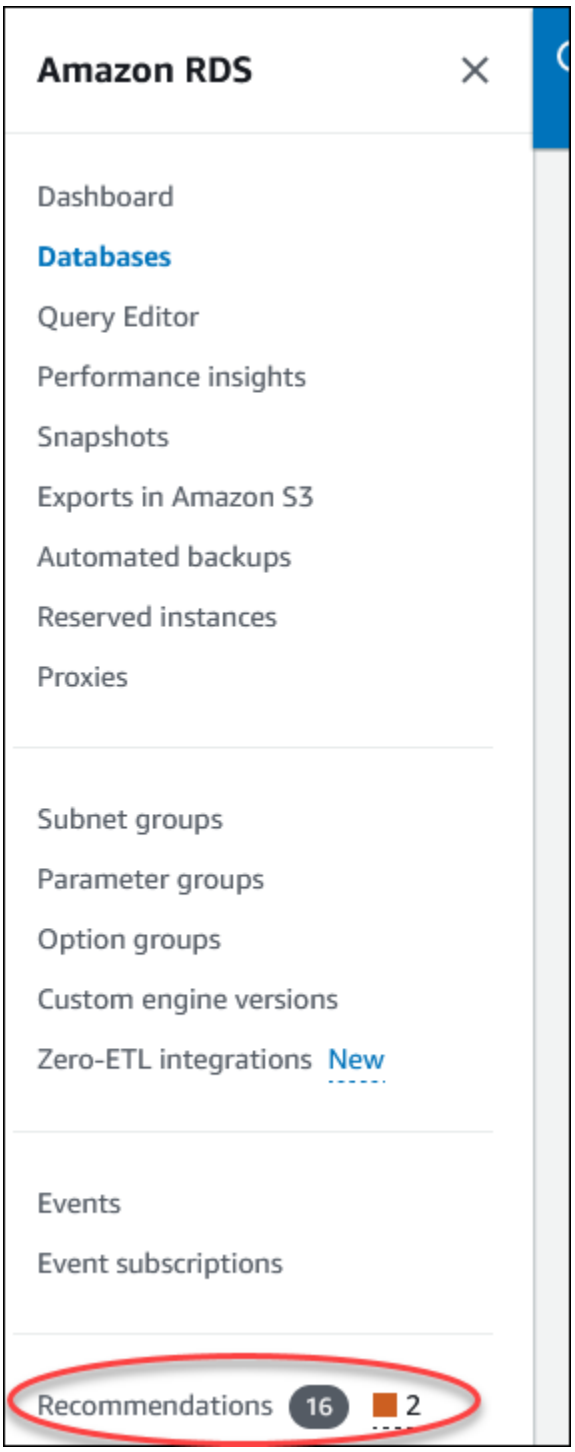
À l’aide de la console Amazon RDS, vous pouvez consulter les recommandations Amazon Aurora relatives aux ressources de votre base de données. Pour un cluster de bases de données, les recommandations s’affichent pour le cluster de bases de données et ses instances.

Console

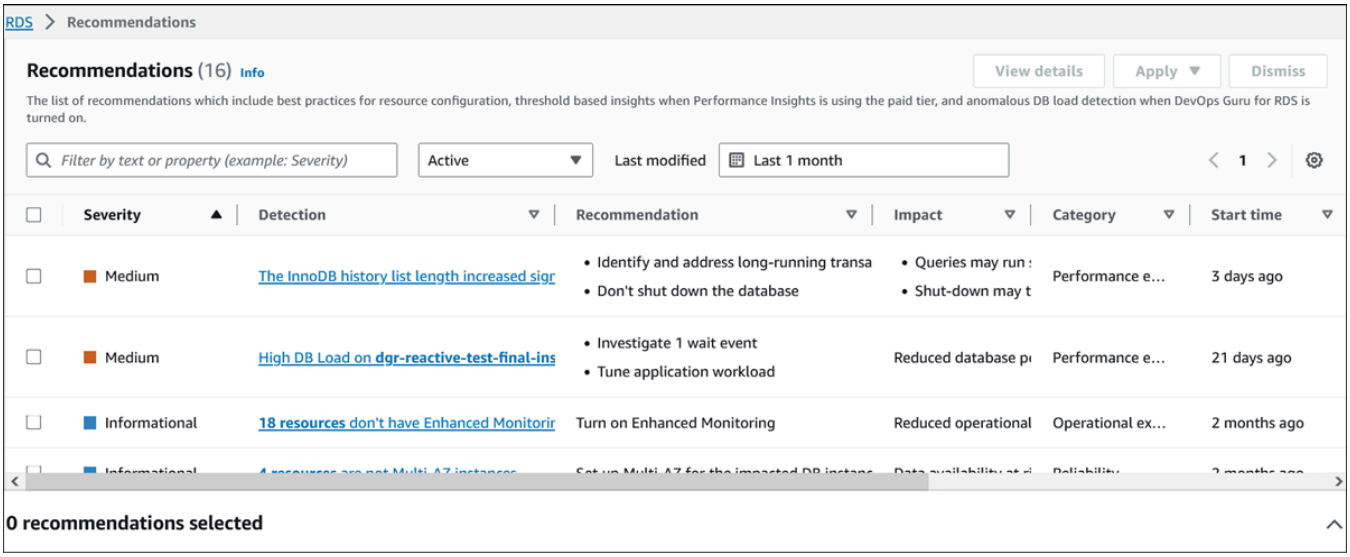
Pour afficher les recommandations Amazon Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, effectuez l’une des opérations suivantes :

   - Choisissez Recommandations. Le nombre de recommandations actives pour vos ressources et le nombre de recommandations avec le niveau de gravité le plus élevé générées le mois dernier sont disponibles à côté de Recommandations. Pour connaître le nombre de recommandations actives pour chaque niveau de gravité, choisissez celui qui indique le niveau de gravité le plus élevé.

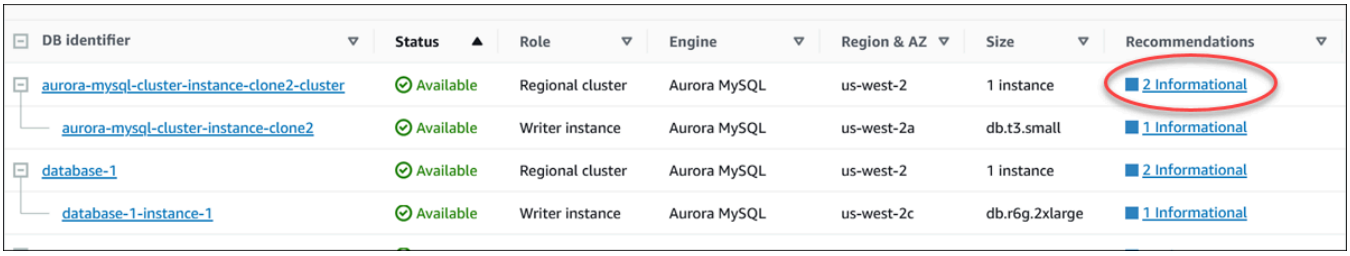


Par défaut, la page Recommandations affiche la liste des nouvelles recommandations du mois dernier. Amazon Aurora fournit des recommandations pour toutes les ressources de votre compte et trie les recommandations en fonction de leur gravité.

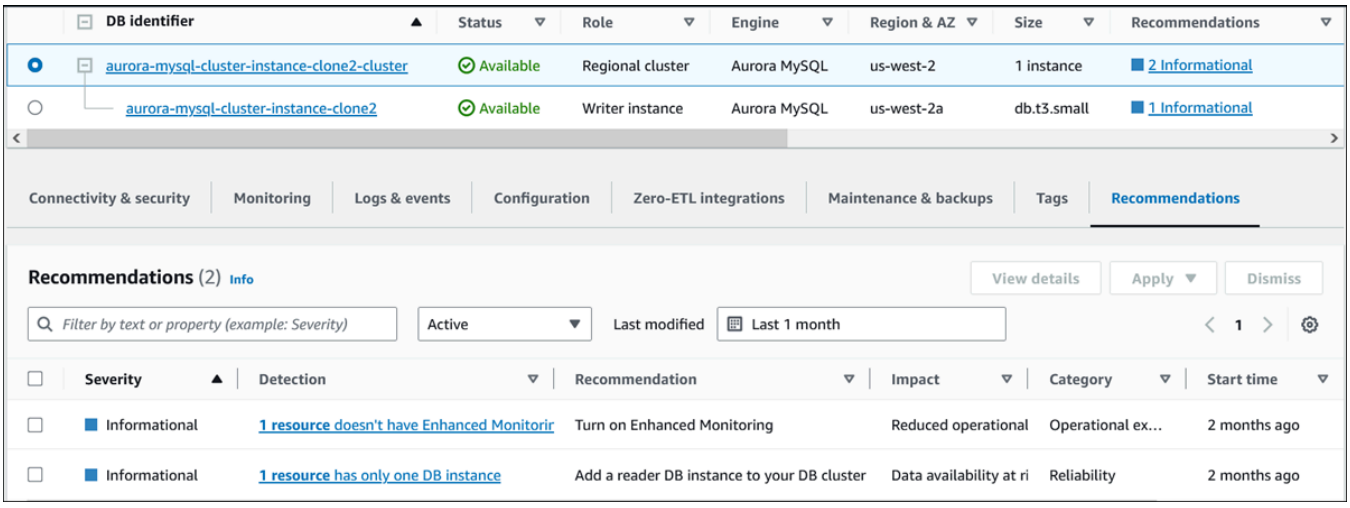


Vous pouvez choisir une recommandation pour consulter une section au bas de la page qui contient les ressources concernées et les détails de la manière dont la recommandation sera appliquée.

- Sur la page Bases de données, sélectionnez Recommandations pour une ressource.



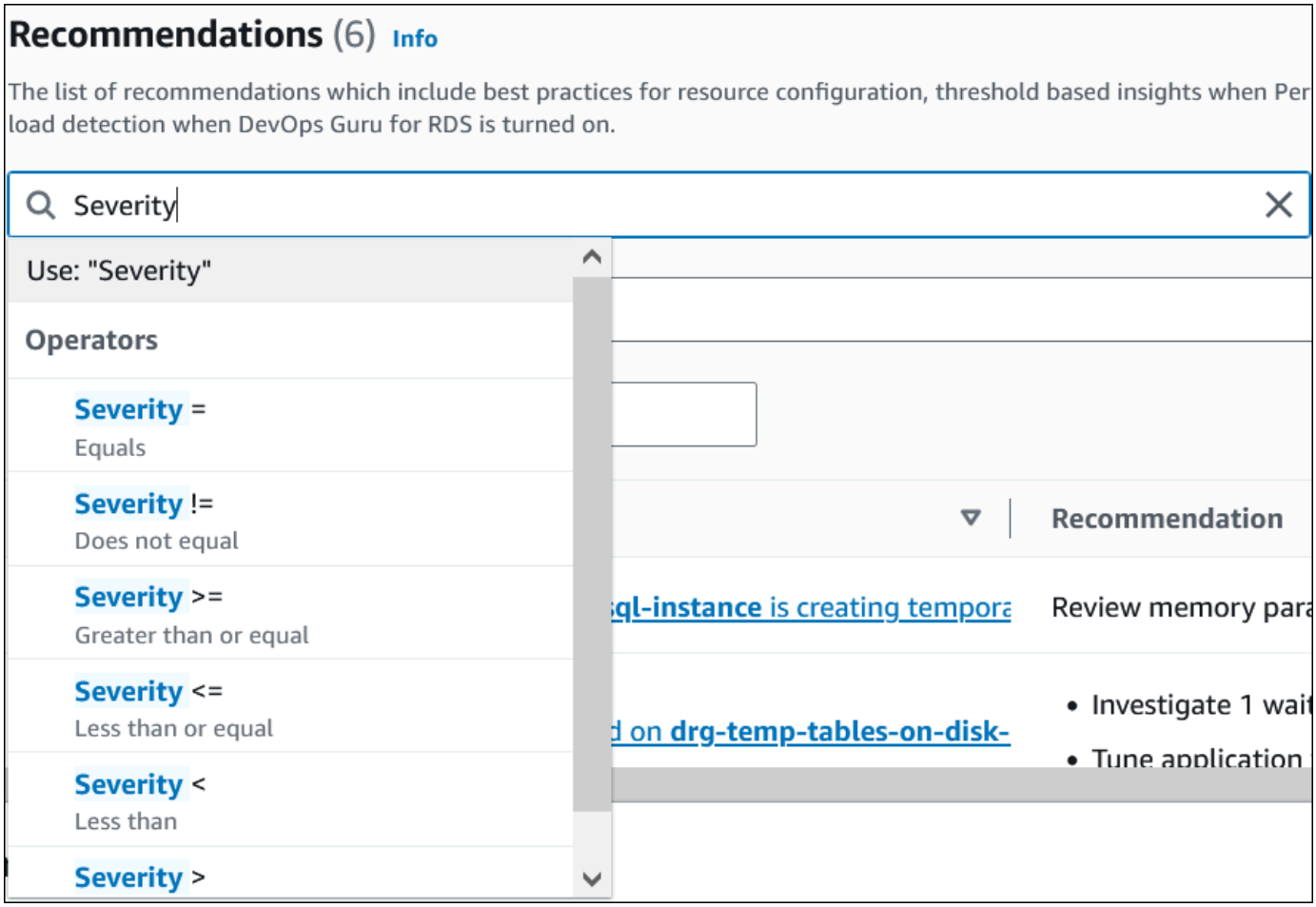
L'onglet Recommandations affiche les recommandations et leurs détails pour la ressource sélectionnée.



Les informations suivantes sont disponibles pour les recommandations :

- Gravité : niveau d’implication du problème. Les niveaux de gravité sont Élevée, Moyenne, Faible et Informatif.
- Détection : nombre de ressources affectées et brève description du problème. Cliquez sur ce lien pour afficher la recommandation et les détails de l’analyse.
- Recommandation : brève description de l’action recommandée à appliquer.
- Impact : brève description de l’impact possible lorsque la recommandation n’est pas appliquée.
- Catégorie : type de recommandation. Les catégories sont Efficacité en matière de performance, Sécurité, Fiabilité, Optimisation des coûts, Excellence opérationnelle et Durabilité.
- Statut : statut actuel de la recommandation. Les statuts possibles sont Toutes, Active, Rejetée, Résolue et En attente.
- Heure de début : heure à laquelle le problème a commencé. Par exemple, Il y a 18 heures.
- Dernière modification : heure à laquelle la recommandation a été mise à jour pour la dernière fois par le système en raison d’une modification de Gravité, ou heure à laquelle vous avez répondu à la recommandation. Par exemple, Il y a 10 heures.
- Heure de fin : heure à laquelle le problème a pris fin. L’heure ne s’affiche pas en cas de problème persistant.
- Identifiant de ressource : nom d’une ou de plusieurs ressources.

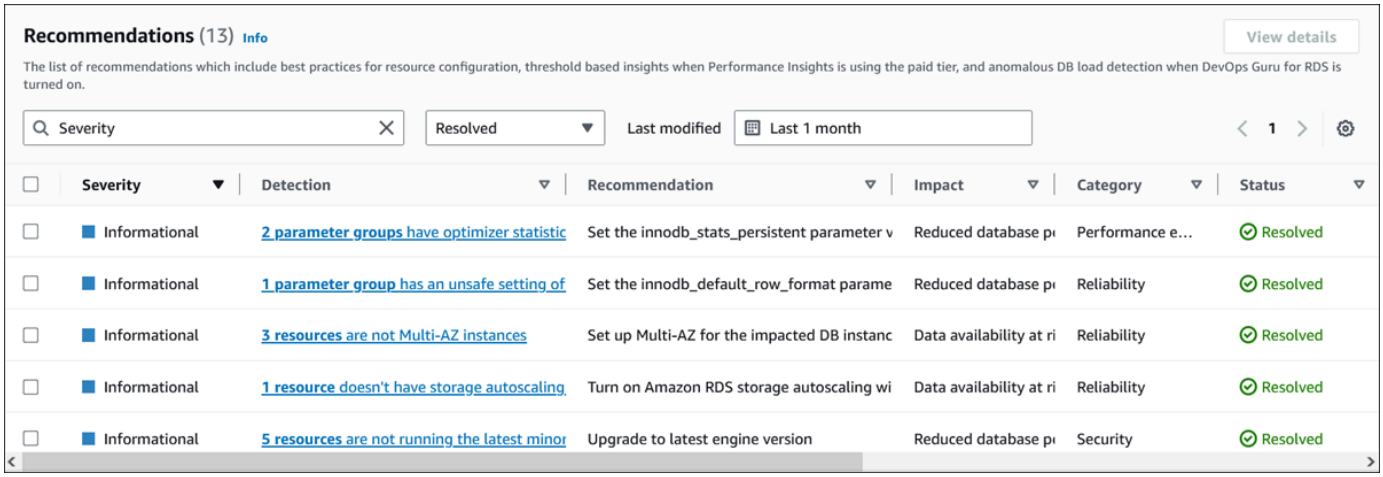
3. (Facultatif) Choisissez les opérateurs de Gravité ou de Catégorie dans le champ pour filtrer la liste des recommandations.



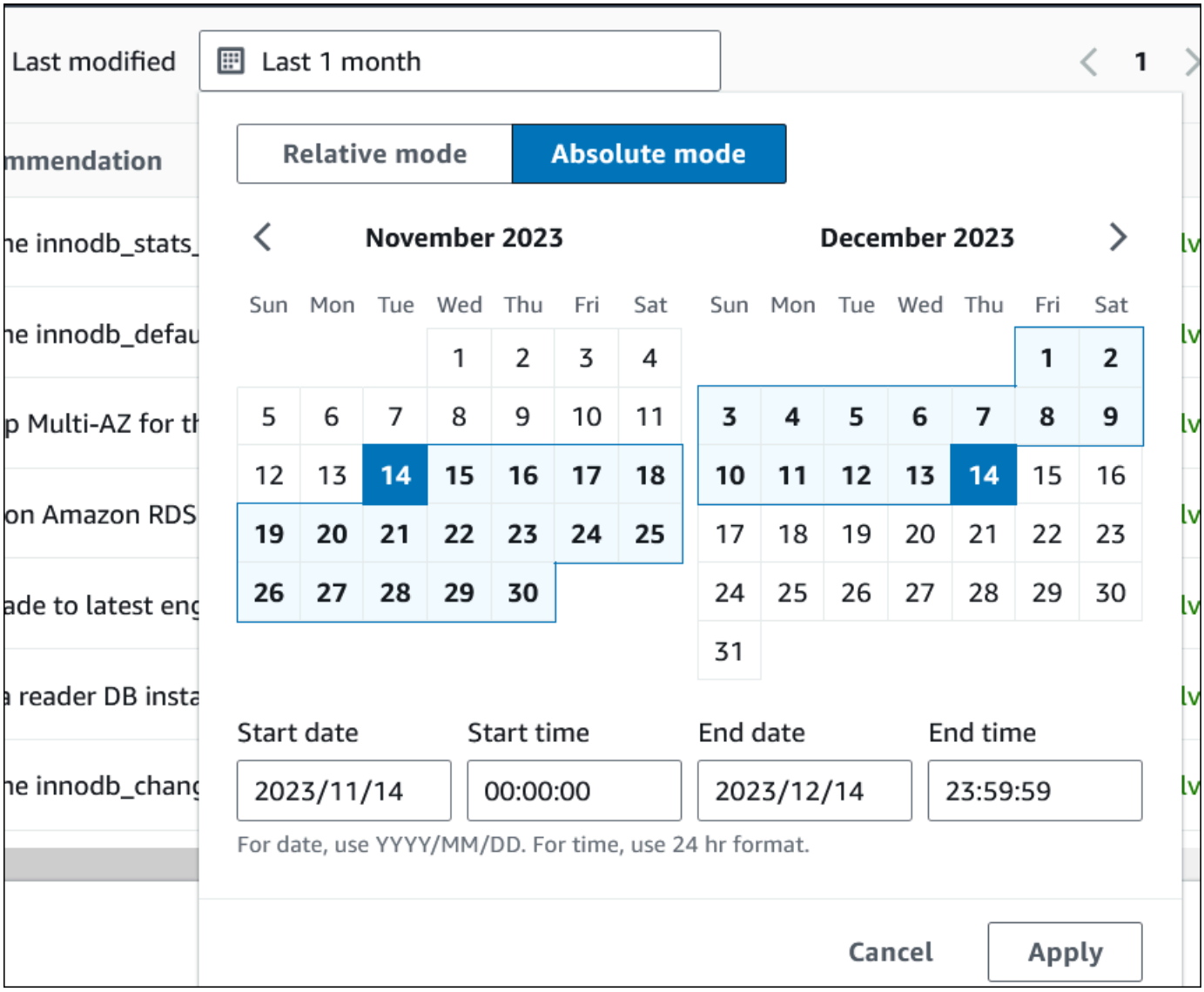
Les recommandations relatives à l'opération sélectionnée s'affichent.

4. (Facultatif) Choisissez l'un des statuts de recommandation suivants :

   - Active (valeur par défaut) : affiche les recommandations en cours que vous pouvez appliquer, planifier pour la prochaine fenêtre de maintenance ou rejeter.
   - Toutes : affiche toutes les recommandations avec leur statut actuel.
   - Rejetée : affiche les recommandations rejetées.
   - Résolue : affiche les recommandations qui ont été résolues.
   - En attente : affiche les recommandations dont les actions recommandées sont en cours ou planifiées pour la prochaine fenêtre de maintenance.



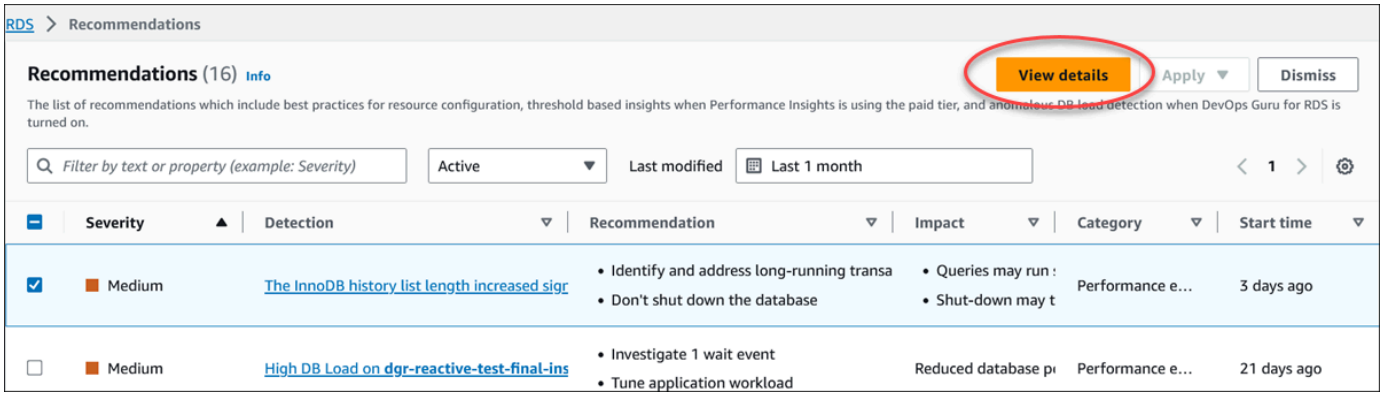
5. (Facultatif) Choisissez Mode relatif ou Mode absolu dans Dernière modification pour modifier la période. La page Recommandations affiche les recommandations générées au cours de la période. La période par défaut est le dernier mois. Dans le Mode absolu, vous pouvez choisir la période ou saisir l'heure dans les champs Date de début et Date de fin.



Les recommandations relatives à la période définie s'affichent.

Notez que vous pouvez consulter toutes les recommandations relatives aux ressources de votre compte en définissant la plage sur Toutes.

6. (Facultatif) Choisissez Préférences sur la droite pour personnaliser les détails à afficher. Vous pouvez choisir un format de page, renvoyer le texte à la ligne et autoriser ou masquer les colonnes.
7. (Facultatif) Choisissez une recommandation, puis Afficher les détails.



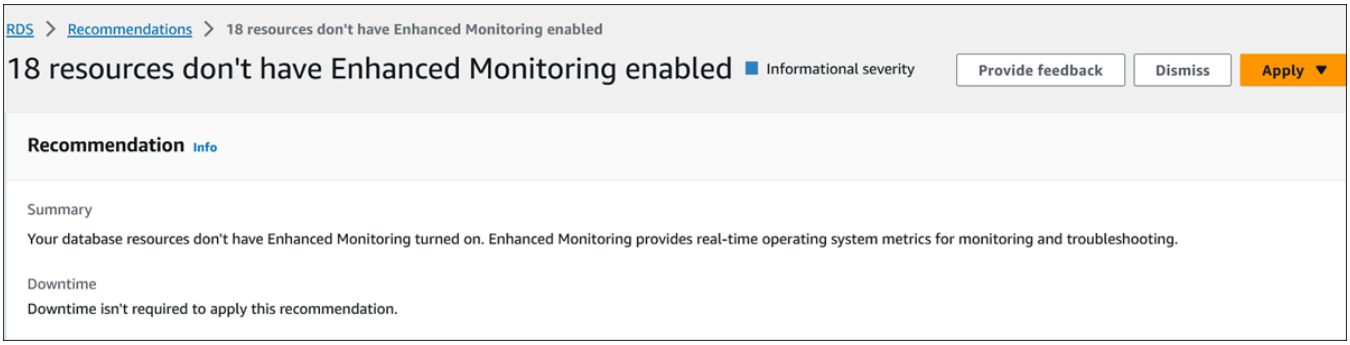
La page des détails de la recommandation s'affiche. Le titre indique le nombre total de ressources ainsi que le problème détecté et sa gravité.

Pour plus d'informations sur les composants figurant sur la page de détails d'une recommandation réactive basée sur les anomalies, consultez Affichage des anomalies réactives dans le Guide de l'utilisateur Amazon DevOps Guru.

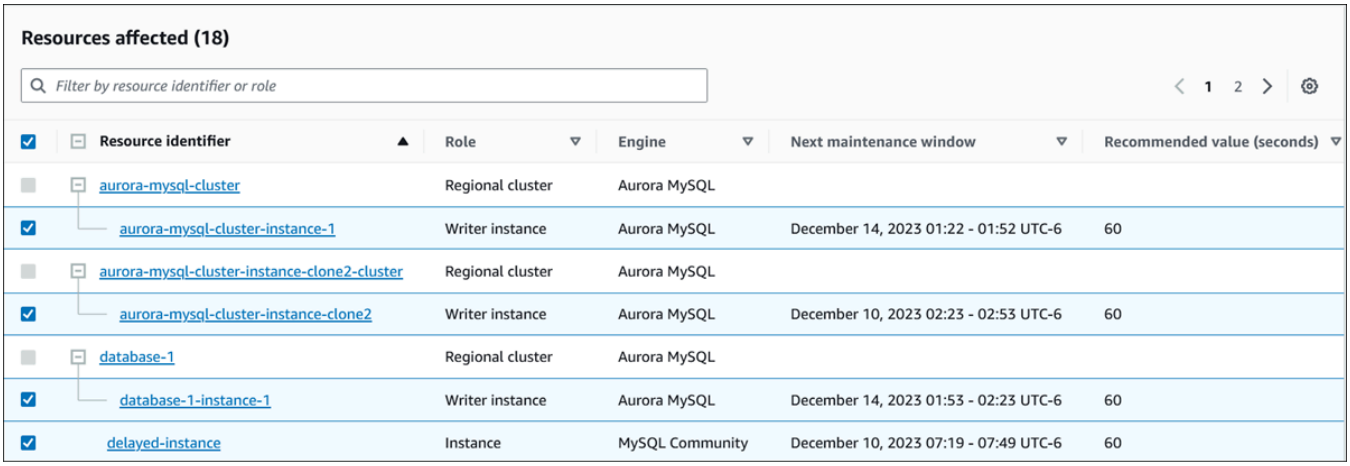
Pour plus d'informations sur les composants figurant sur la page de détails d'une recommandation proactive basée sur un seuil, consultez Affichage des recommandations proactives de Performance Insights.

Les autres recommandations automatiques affichent les composants suivants sur la page de détails de la recommandation :

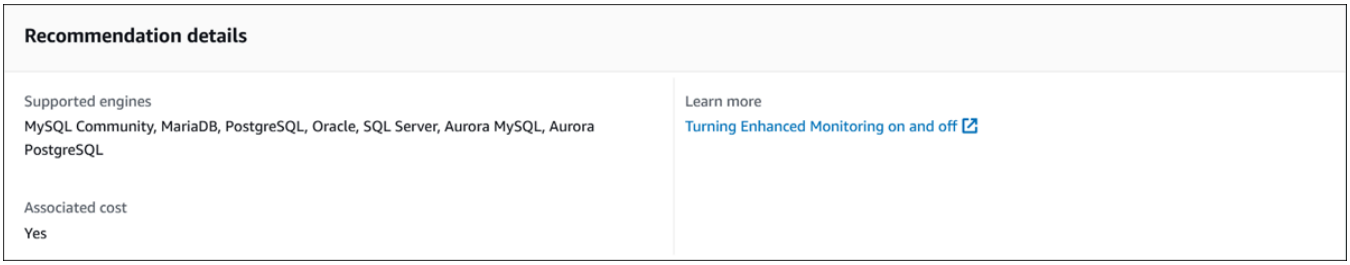
- Recommandation : un résumé de la recommandation et indiquant si une durée d'indisponibilité est nécessaire pour appliquer la recommandation.



- Ressources affectées : détails des ressources affectées.



- Détails de la recommandation : informations sur le moteur pris en charge, tout coût associé requis pour appliquer la recommandation et lien vers la documentation pour en savoir plus.



Interface de ligne de commande (CLI)

Pour consulter les recommandations Amazon RDS relatives aux instances de base de données ou aux clusters de bases de données, utilisez la commande suivante dans l'AWS CLI.

```
aws rds describe-db-recommendations
```

API RDS

Pour afficher les recommandations Amazon RDS à l'aide de l'API Amazon RDS, utilisez l'opération DescribeDBRecommendations.

Application des recommandations Amazon Aurora

Pour appliquer les recommandations Amazon Aurora à l'aide de la console Amazon RDS, sélectionnez une recommandation basée sur la configuration ou une ressource affectée sur la page de détails. Choisissez ensuite d'appliquer la recommandation immédiatement ou de la planifier pour la fenêtre de maintenance suivante. Il peut être nécessaire de redémarrer la ressource pour que cette

modification prenne effet. Pour quelques recommandations relatives aux groupes de paramètres de base de données, vous devrez peut-être redémarrer les ressources.

Les recommandations proactives ou réactives basées sur des seuils ne pourront pas être appliquées et peuvent nécessiter un examen supplémentaire.

Console

Pour appliquer une recommandation basée sur la configuration

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, effectuez l'une des opérations suivantes :

   - Choisissez Recommandations.

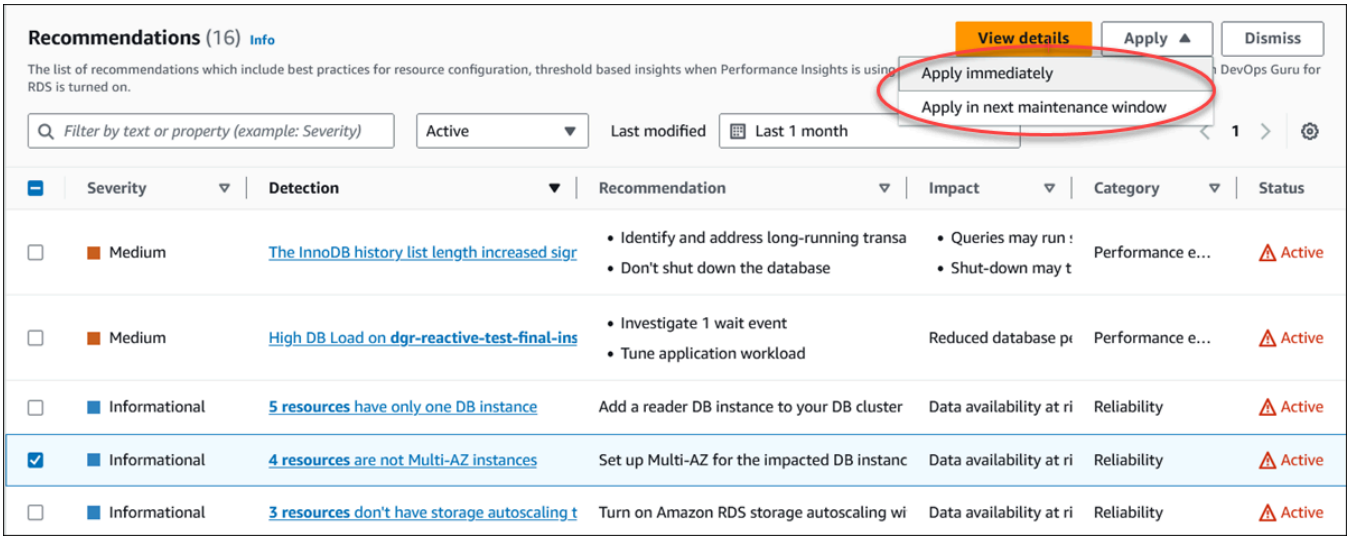
     La page Recommandations contenant la liste de toutes les recommandations s'affiche.
   - Dans la page des bases de données, choisissez Bases de données, puis Recommandations pour une ressource.

     Les détails s'affichent dans l'onglet Recommandations pour la recommandation sélectionnée.
   - Choisissez Détection pour une recommandation active dans la page Recommandations ou sur l'onglet Recommandations de la page Bases de données.

     La page des détails de la recommandation s'affiche.
3. Choisissez une recommandation, ou une ou plusieurs ressources concernées dans la page de détails des recommandations, puis effectuez l'une des opérations suivantes :

   - Choisissez Appliquer, puis Appliquer immédiatement pour appliquer la recommandation immédiatement.
   - Choisissez Appliquer, puis Appliquer lors de la prochaine fenêtre de maintenance pour planifier l'application au cours de la prochaine fenêtre de maintenance.

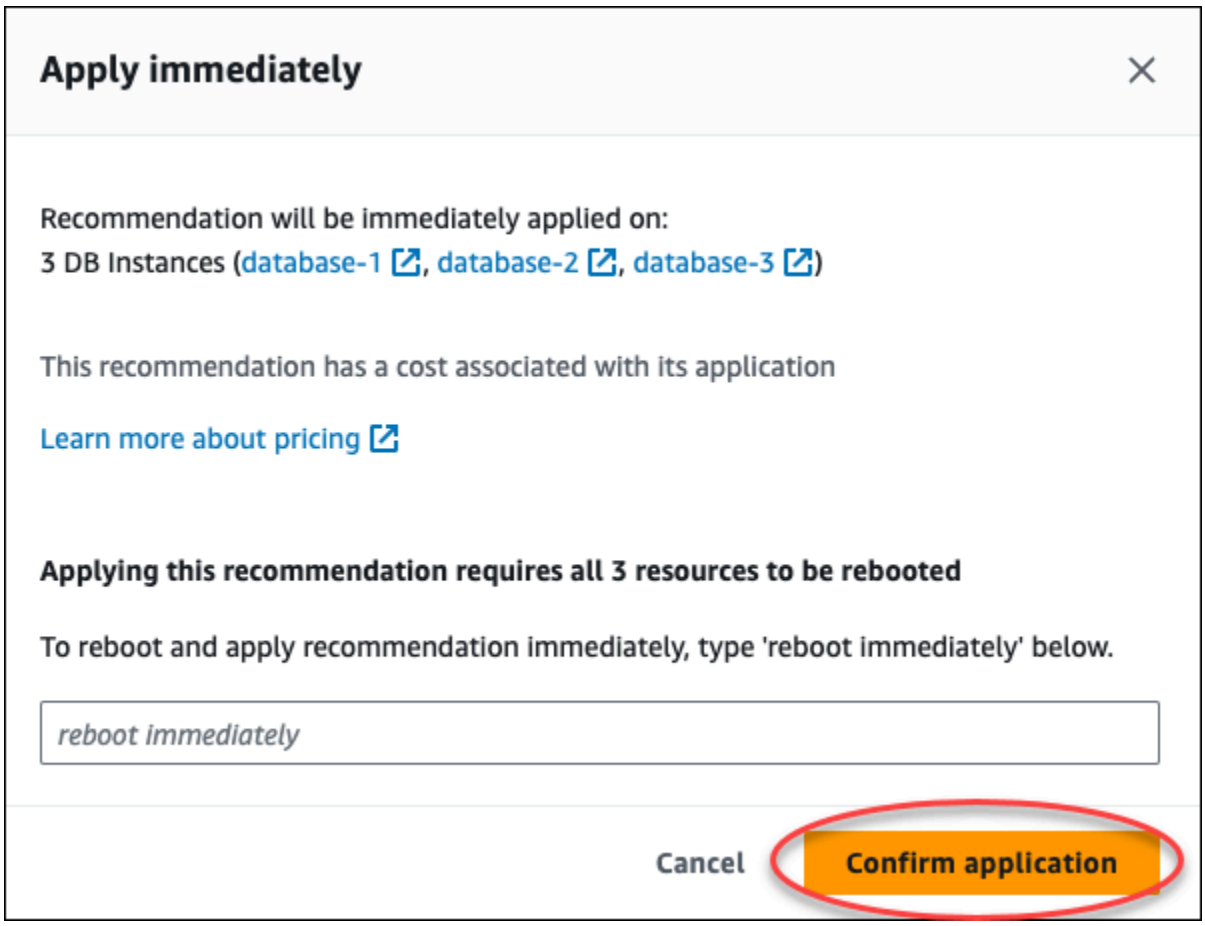
     Le statut de la recommandation sélectionnée est mis à jour à En attente jusqu'à la fenêtre de maintenance suivante.



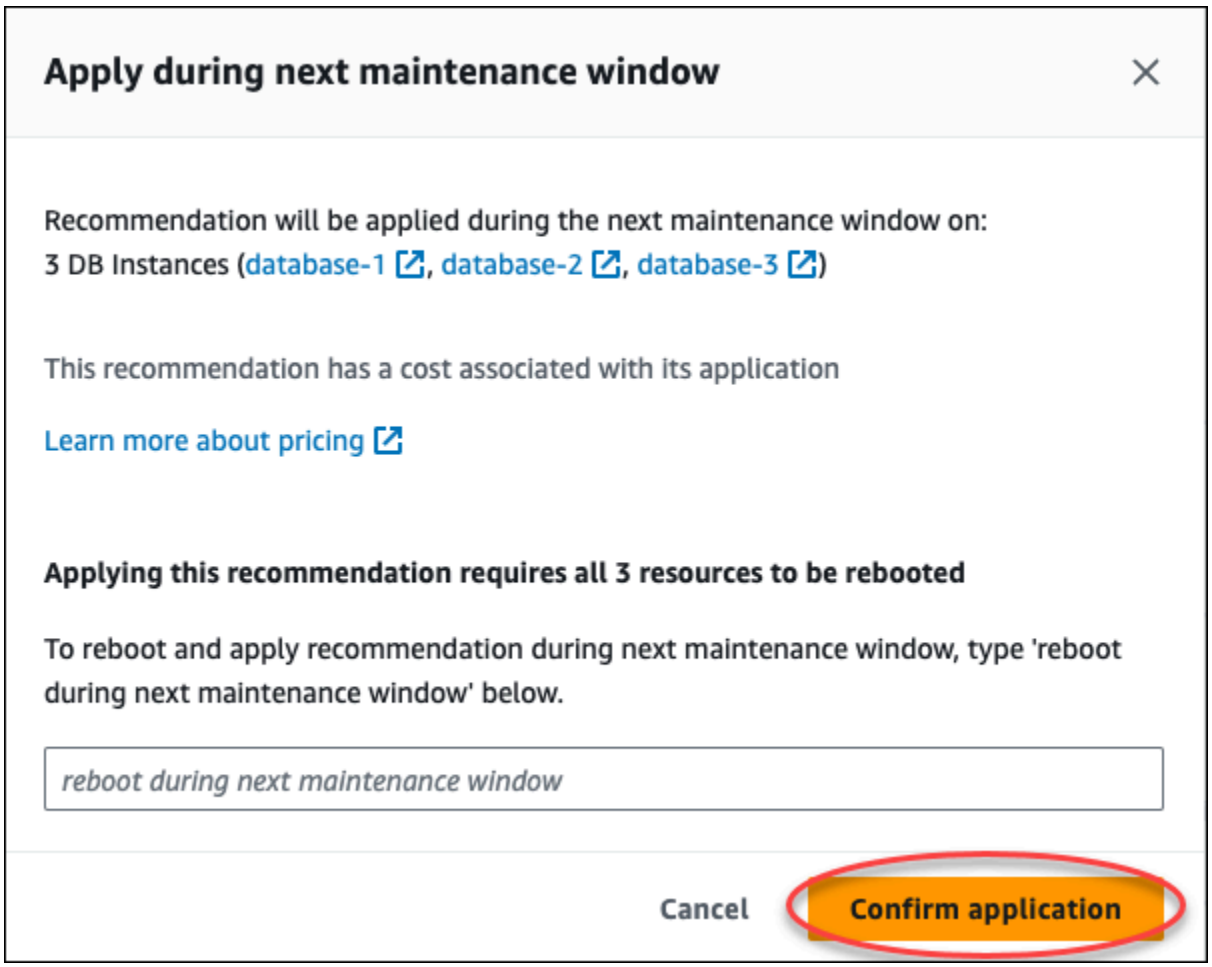
Une fenêtre de confirmation s'affiche.

4. Choisissez Confirmer l'application pour appliquer la recommandation. Cette fenêtre confirme si les ressources ont besoin d'un redémarrage automatique ou manuel pour que les modifications prennent effet.

   L'exemple suivant montre la fenêtre de confirmation pour appliquer la recommandation immédiatement.



L'exemple suivant montre la fenêtre de confirmation permettant de planifier l'application de la recommandation dans la fenêtre de maintenance suivante.



Une bannière affiche un message en cas de réussite ou d'échec de la recommandation appliquée.

L'exemple suivant montre la bannière avec le message de réussite.

L'exemple suivant montre la bannière avec le message d'échec.

API RDS

Pour appliquer une recommandation Aurora basée sur la configuration à l'aide de l'API Amazon RDS

1. Utilisez l'opération DescribeDBRecommendations. Les `RecommendedActions` dans la sortie peuvent contenir une ou plusieurs actions recommandées.
2. Utilisez l'objet RecommendedAction pour chaque action recommandée à l'étape 1. La sortie contient `Operation` et `Parameters`.

   L'exemple suivant illustre la sortie avec une action recommandée.

```
   "RecommendedActions": [
       {
           "ActionId": "0b19ed15-840f-463c-a200-b10af1b552e3",
           "Title": "Turn on auto backup", // localized
           "Description": "Turn on auto backup for my-mysql-instance-1", //
localized
           "Operation": "ModifyDbInstance",
           "Parameters": [
               {
                   "Key": "DbInstanceIdentifier",
                   "Value": "my-mysql-instance-1"
               },
               {
                   "Key": "BackupRetentionPeriod",
                   "Value": "7"
               }
           ],
           "ApplyModes": ["immediately", "next-maintenance-window"],
           "Status": "applied"
       },
       ... // several others
   ],
```

3. Utilisez l'`operation` pour chaque action recommandée à partir de la sortie de l'étape 2 et entrez les valeurs de `Parameters`.
4. Une fois l'opération de l'étape 2 réussie, utilisez l'opération ModifyDBRecommendation pour modifier le statut de la recommandation.

Rejet des recommandations Amazon Aurora

Vous pouvez rejeter une ou plusieurs recommandations Amazon Aurora à l'aide de la console Amazon RDS, de l'AWS CLI ou de l'API Amazon RDS.

Console

Pour rejeter une ou plusieurs recommandations

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, effectuez l'une des opérations suivantes :

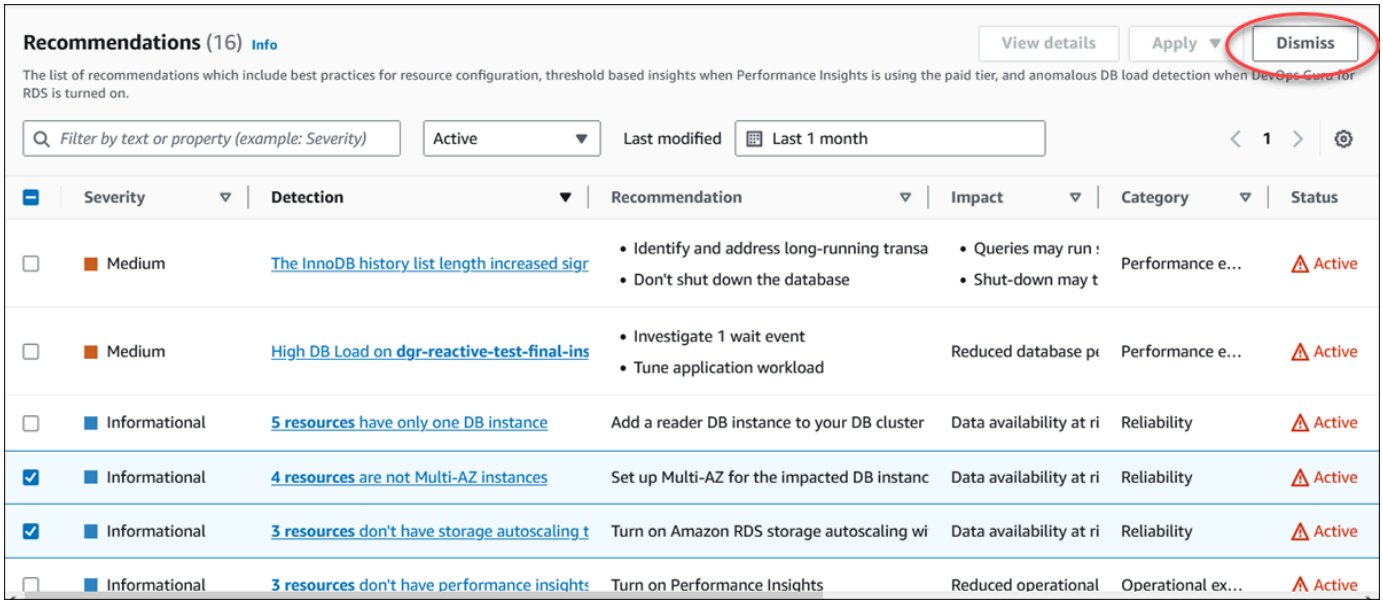
   - Choisissez Recommandations.

     La page Recommandations contenant la liste de toutes les recommandations s'affiche.
   - Dans la page des bases de données, choisissez Bases de données, puis Recommandations pour une ressource.

     Les détails s'affichent dans l'onglet Recommandations pour la recommandation sélectionnée.
   - Choisissez Détection pour une recommandation active dans la page Recommandations ou sur l'onglet Recommandations de la page Bases de données.

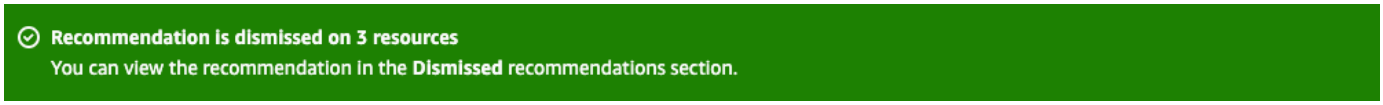
     La page de détails des recommandations affiche la liste des ressources concernées.
3. Choisissez une ou plusieurs recommandations, ou une ou plusieurs ressources concernées dans la page de détails des recommandations, puis choisissez Rejeter.

   L'exemple suivant montre la page Recommandations avec plusieurs recommandations actives sélectionnées pour être ignorées.



Une bannière affiche un message lorsque la ou les recommandations sélectionnées sont rejetées.

L'exemple suivant montre la bannière avec le message de réussite.



L'exemple suivant montre la bannière avec le message d'échec.

Interface de ligne de commande (CLI)

Pour rejeter une recommandation Aurora à l'aide de l'AWS CLI

1. Exécutez la commande `aws rds describe-db-recommendations --filters "Name=status,Values=active"`.

   La sortie fournit une liste de recommandations ayant le statut `active`.

2. Recherchez `recommendationId` pour la recommandation que vous souhaitez ignorer à partir de l'étape 1.

3. Exécutez la commande `>aws rds modify-db-recommendation --status dismissed --recommendationId <ID>` avec l'`recommendationId` à partir de l'étape 2 pour ignorer la recommandation.

API RDS

Pour rejeter une recommandation Aurora à l'aide de l'API Amazon RDS, utilisez l'opération ModifyDBRecommendation.

Transformation des recommandations Amazon Aurora ignorées en recommandations actives

Vous pouvez faire passer une ou plusieurs recommandations Amazon Aurora ignorées à actives en utilisant la console Amazon RDS, l'AWS CLI, ou l'API Amazon RDS.

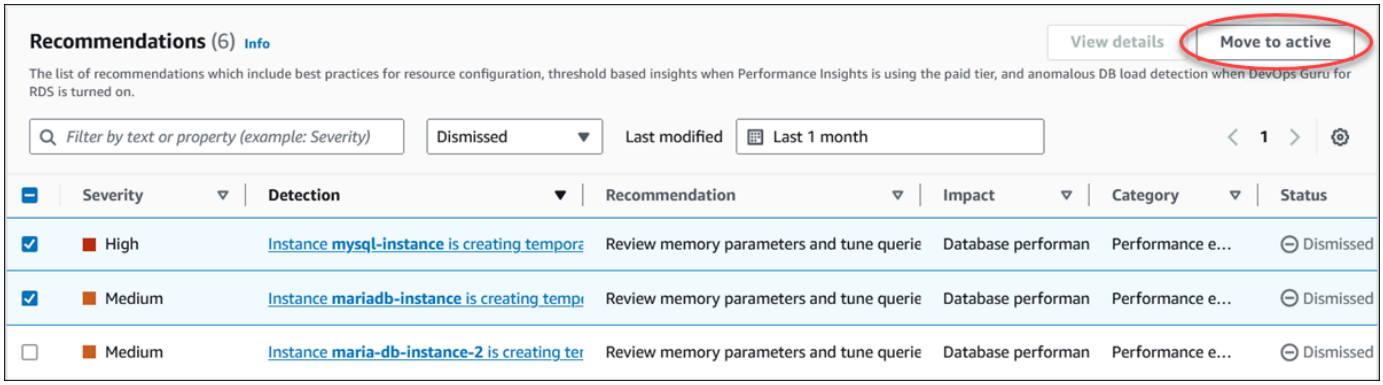
Console

Pour faire passer une ou plusieurs recommandations ignorées à actives

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, effectuez l'une des opérations suivantes :
   - Choisissez Recommandations.

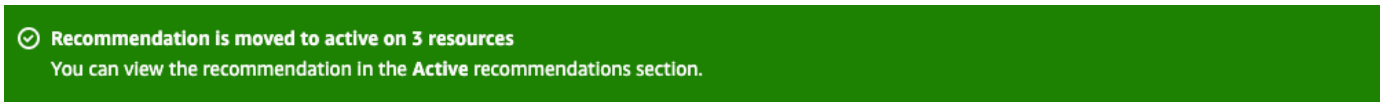
     La page Recommandations affiche une liste de recommandations triées par gravité pour toutes les ressources de votre compte.
   - Dans la page des bases de données, choisissez Bases de données, puis Recommandations pour une ressource.

     L'onglet Recommandations affiche les recommandations et leurs détails pour la ressource sélectionnée.
3. Choisissez une ou plusieurs recommandations ignorées dans la liste, puis choisissez Passer à actif.



Une bannière affiche un message de réussite ou d'échec lorsque les recommandations sélectionnées passent du statut Ignoré à Actif.

L'exemple suivant montre la bannière avec le message de réussite.



L'exemple suivant montre la bannière avec le message d'échec.

Interface de ligne de commande (CLI)

Pour transformer une recommandation Aurora ignorée en recommandation active à l'aide de l'AWS CLI

1. Exécutez la commande `aws rds describe-db-recommendations --filters "Name=status,Values=dismissed"`.

   La sortie fournit une liste de recommandations ayant le statut `dismissed`.
2. Recherchez `recommendationId` pour la recommandation dont vous souhaitez modifier le statut à partir de l'étape 1.
3. Exécutez la commande `>aws rds modify-db-recommendation --status active --recommendationId <ID>` avec `recommendationId` à partir de l'étape 2 pour remplacer le statut de la recommandation par Actif.

API RDS

Pour remplacer une recommandation Aurora ignorée par une recommandation active à l'aide de l'API Amazon RDS, utilisez l'opération ModifyDBRecommendation.

Référence des recommandations d'Amazon Aurora

Amazon Aurora génère des recommandations pour une ressource lors de la création ou de la modification de celle-ci. Vous trouverez des exemples de recommandations d'Amazon Aurora dans le tableau suivant.

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Les sauvegardes automatiques des ressources sont désactivées	Les sauvegardes automatiques ne sont pas activées pour vos instances de base de données. Les sauvegardes automatisées sont recommandées car elles permettent la point-in-time restauration de vos instances de base de données.	Activez les sauvegardes automatiques avec une période de conservation allant jusqu'à 14 jours.	Oui	Présentation de la sauvegarde et de la restauration d'un cluster de bases de données Aurora Demystifying Amazon RDS backup storage costs sur le blog AWS Database
Une mise à niveau de version mineure du moteur est requise	Les ressources de votre base de données n'exécutent pas la dernière version mineure de moteur de base de données. La dernière version mineure contient les correctif	Mettez à niveau vers la dernière version du moteur.	Oui	Entretien d'un cluster de bases de données Amazon Aurora

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
	s de sécurité les plus récents et d'autres améliorations.			
La surveillance améliorée est désactivée	La surveillance améliorée n'est pas activée sur les ressources de votre base de données. La surveillance améliorée fournit des métriques de système d'exploitation en temps réel pour la surveillance et le dépannage.	Activez la fonctionnalité Surveillance améliorée.	Non	Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le chiffrement du stockage est désactivé	Amazon RDS prend en charge le chiffrement au repos pour tous les moteurs de base de données en utilisant les clés que vous gérez dans AWS Key Management Service (AWS KMS). Sur une instance de base de données active avec le chiffrement Amazon RDS, les données stockées au repos dans le stockage sont chiffrées, comme les sauvegardes automatiques, les réplicas en lecture et les instantanés. Si le chiffrement n'est pas activé lors de la création d'un cluster de bases de données Aurora, vous devez restaurer un instantané déchiffré sur un cluster de bases de données chiffré.	Activez le chiffrement des données au repos pour votre cluster de bases de données.	Oui	Sécurité dans Amazon Aurora

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Clusters de bases de données avec toutes les instances dans la même zone de disponibilité	Les clusters de bases de données se trouvent actuellement dans une seule zone de disponibilité. Utilisez plusieurs zones de disponibilité pour améliorer la disponibilité.	Ajoutez les instances de base de données à plusieurs zones de disponibilité de votre cluster de bases de données.	Non	Haute disponibilité pour Amazon Aurora
Instances de base de données dans les clusters avec des tailles d'instance hétérogènes	Nous vous recommandons d'utiliser les mêmes classe et taille d'instance de base de données pour toutes les instances dans votre cluster de bases de données.	Utilisez les mêmes classe et taille d'instance pour toutes les instances de base de données de votre cluster de bases de données.	Oui	Réplication avec Amazon Aurora
Instances de base de données dans les clusters avec des classes d'instance hétérogènes	Nous vous recommandons d'utiliser les mêmes classe et taille d'instance de base de données pour toutes les instances dans votre cluster de bases de données.	Utilisez les mêmes classe et taille d'instance pour toutes les instances de base de données de votre cluster de bases de données.	Oui	Réplication avec Amazon Aurora

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Instances de base de données dans les clusters avec des groupes de paramètres hétérogènes	Nous vous recommandons d'utiliser le même groupe de paramètres de base de données pour toutes les instances de base de données du cluster de bases de données.	Associez l'instance de base de données avec le groupe de paramètres de base de données associé à l'instance de rédacteur du cluster de bases de données.	Non	Groupes de paramètres pour Amazon Aurora
Les clusters de bases de données Amazon RDS ont une instance de base de données	Ajoutez au moins une instance de base de données supplémentaire à votre cluster de bases de données pour améliorer la disponibilité et les performances.	Ajoutez une instance de base de données de lecteur à votre cluster de bases de données.	Non	Haute disponibilité pour Amazon Aurora
Performance Insights est désactivé	Performance Insights surveille la charge de votre instance de base de données pour vous permettre d'analyser et de résoudre les problèmes de performances de base de données. Nous vous recommandons d'activer Performance Insights.	Activez Performance Insights.	Non	Surveillance de la charge de la base de données avec Performance Insights sur

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
La mise à jour des versions majeures des ressources RDS est requise	Les bases de données dotées de la version majeure actuelle du moteur de base de données ne seront pas prises en charge. Nous vous recommandons de mettre à niveau vers la dernière version majeure, qui inclut de nouvelles fonctionnalités et améliorations.	Mettez à niveau vers la dernière version majeure du moteur de base de données.	Oui	Mises à jour d'Amazon Aurora Création d'un blue/green déploiement dans)
Taille maximale du volume du cluster de bases de données	Les nouvelles versions du moteur prennent en charge des volumes de stockage plus importants pour votre cluster de base de données.	Nous vous recommandons de mettre à niveau la version du moteur de votre cluster de base de données vers la dernière version afin de bénéficier d'une capacité de stockage accrue.	Oui	Limites de taille Amazon Aurora

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Clusters de bases de données avec toutes les instances de lecteur dans la même zone de disponibilité	Les zones de disponibilité (AZs) sont des emplacements distincts les uns des autres afin de permettre une isolation en cas de panne dans chaque AWS région. Nous vous recommandons de répartir l'instance principale et les instances de lecteur de votre cluster de base de données sur plusieurs instances afin AZs d'améliorer la disponibilité de votre cluster de bases de données. Vous pouvez créer un cluster multi-AZ à l'aide de la console AWS de gestion, de la AWS CLI ou de l'API Amazon RDS lors de la création du cluster. Vous pouvez également transformer un cluster Aurora en cluster multi-	Votre cluster de bases de données a toutes ses instances de lecture dans la même zone de disponibilité. Nous vous recommandons de distribuer les instances de lecteur entre plusieurs zones de disponibilité. La distribution augmente la disponibilité et améliore le temps de réponse en réduisant la latence du réseau entre les clients et la base de données.	Non	Haute disponibilité pour Amazon Aurora

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
	AZ en ajoutant une nouvelle instance de lecteur et en spécifiant une autre zone de disponibilité.			
Les paramètres de mémoire de base de données divergent de ceux par défaut	Les paramètres de mémoire des instances de base de données sont significativement différents des valeurs par défaut. Ces paramètres peuvent avoir un impact sur les performances et provoquer des erreurs. Nous vous recommandons de rétablir les paramètres de mémoire personnalisés pour l'instance de base de données à leurs valeurs par défaut dans le groupe de paramètres de base de données.	Rétablissez les paramètres de mémoire à leurs valeurs par défaut.	Non	Groupes de paramètres pour Amazon Aurora

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre de cache des requêtes est activé	Lorsque les modifications nécessitent la purge de votre cache de requêtes, votre instance de base de données semble bloquée. La plupart des charges de travail ne bénéficient pas d'un cache de requête. Le cache de requête a été supprimé de MySQL 8.0 et versions ultérieures. Nous vous recommandons de définir le paramètre query_cache_type sur 0.	Définissez le paramètre `query_cache_type` sur `0` dans votre groupe de paramètres de base de données.	Oui	Groupes de paramètres pour Amazon Aurora

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre `log_output` est défini sur table	Lorsque `log_output` est défini sur `TABLE`, plus d'espace de stockage est utilisé que lorsque `log_output` est défini sur `FILE`. Nous vous recommandons de définir le paramètre sur `FILE` pour éviter d'atteindre la limite de taille de stockage. Défini sur `FILE` par défaut dans MySQL 8.4 et les versions ultérieures.	Définissez le paramètre `log_output` sur `FILE` dans votre groupe de paramètres de base de données.	Non	Fichiers journaux de base de données Aurora MySQL

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre `autovacuum` est désactivé	Le paramètre autovacuum est désactivé pour les clusters de bases de données. La désactivation d'autovacuum accroît le gonflement de la table et de l'index, et a un impact sur les performances. Nous vous recommandons d'activer autovacuum dans vos groupes de paramètres de base de données.	Activez le paramètre autovacuum dans les groupes de paramètres de votre cluster de bases de données.	Non	Présentation de l'autovacuum dans les environnements Amazon RDS for PostgreSQL sur le blog de base de données AWS

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre `synchronous_commit` est désactivé	Lorsque le paramètre `synchronous_commit` est désactivé, des données peuvent être perdues lors d'une panne de base de données. La durabilité de la base de données est menacée. Nous vous recommandons d'activer le paramètre `synchronous_commit` .	Activez le paramètre `synchronous_commit` dans vos groupes de paramètres de base de données.	Oui	Paramètres Amazon Aurora PostgreSQL : réplication, sécurité et journalisation sur le blog de base de données AWS

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre `track_counts` est désactivé	Lorsque le paramètre `track_counts` est désactivé, la base de données ne collecte pas les statistiques d'activité de base de données. Autovacuum a besoin de ces statistiques pour fonctionner correctement. Nous vous recommandons de définir le paramètre `track_counts` sur 1.	Définissez le paramètre `track_counts` sur 1.	Non	Statistiques d'exécution pour PostgreSQL

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre `enable_indexonlyscan` est désactivé	Le planificateur ou l'optimiseur de requêtes ne peut pas utiliser le type de plan d'analyse d'index uniquement lorsqu'il est désactivé. Nous vous recommandons de définir la valeur du paramètre `enable_indexonlyscan` sur 1.	Définissez la valeur du paramètre `enable_indexonlyscan` sur 1.	Non	Configuration de la méthode du planificateur pour PostgreSQL
Le paramètre `enable_indexscan` est désactivé	Le planificateur ou l'optimiseur de requêtes ne peut pas utiliser le type de plan d'analyse d'index lorsqu'il est désactivé. Nous vous recommandons de définir `enable_indexscan` sur 1.	Définissez la valeur du paramètre `enable_indexscan` sur 1.	Non	Configuration de la méthode du planificateur pour PostgreSQL

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre `innodb_flush_log_at_trx` est désactivé	La valeur du paramètre `innodb_flush_log_at_trx` de votre instance de base de données n'est pas une valeur sûre. Ce paramètre contrôle la persistance des opérations de validation sur le disque. Nous vous recommandons de définir le paramètre `innodb_flush_log_at_trx` sur 1.	Définissez la valeur du paramètre `innodb_flush_log_at_trx` sur 1.	Non	Configuration de la fréquence à laquelle le tampon du journal est vidé

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre `innodb_stats_persistent` est désactivé	Votre instance de base de données n'est pas configurée pour conserver les statistiques InnoDB sur le disque. Lorsque les statistiques ne sont pas stockées, elles sont recalculées à chaque redémarrage de l'instance et à chaque accès à la table. Cela entraîne des variations dans le plan d'exécution des requêtes. Vous pouvez modifier la valeur de ce paramètre global au niveau de la table. Nous vous recommandons de définir la valeur du paramètre `innodb_stats_persistent` sur `ON`.	Définissez la valeur du paramètre `innodb_stats_persistent` sur `ON`.	Non	Groupes de paramètres pour Amazon Aurora

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre `innodb_open_files` est faible	Le paramètre `innodb_open_files` contrôle le nombre de fichiers qu'InnoDB peut ouvrir à la fois. InnoDB ouvre tous les fichiers d'espace de table journal et système lorsque mysqld est en cours d'exécution. Votre instance de base de données a une faible valeur pour le nombre maximal de fichiers qu'InnoDB peut ouvrir en même temps. Nous vous recommandons de défininir le paramètre `innodb_open_files` sur la valeur minimale 65.	Définissez le paramètre `innodb_open_files` sur une valeur minimale de 65.	Oui	InnoDB ouvre des fichiers pour MySQL

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre `max_user_connections` est faible	Votre instance de base de données a une valeur faible pour le nombre maximal de connexions simultanées pour chaque compte de base de données. Nous vous recommandons de définir le paramètre `max_user_connections` sur un nombre supérieur à 5.	Augmentez la valeur du paramètre `max_user_connections` à un nombre supérieur à 5.	Oui	Définition des limites de ressources du compte pour MySQL

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Les réplicas en lecture sont ouverts en mode accessible en écriture	Le réplica en lecture de votre instance de base de données est en mode accessible en écriture, ce qui autorise les mises à jour depuis les clients. Nous vous recommandons de définir le paramètre `read_only` sur `TrueIfReplica` telle sorte que les réplicas en lecture ne soient pas en mode accessible en écriture.	Définissez la valeur du paramètre `read_only` sur `TrueIfReplica` .	Non	Groupes de paramètres pour Amazon Aurora

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le réglage du paramètre `innodb_default_row_format` n'est pas sûr	Votre instance de base de données rencontre un problème connu : une table créée dans une version de MySQL inférieure à 8.0.26 avec le paramètre `row_format` défini sur `COMPACT` ou `REDUNDANT` est inaccessible et irrécupérable lorsque l'index dépasse 767 octets. Nous vous recommandons de définir la valeur du paramètre `innodb_default_row_format` sur `DYNAMIC`.	Définissez la valeur du paramètre `innodb_default_row_format` sur `DYNAMIC`.	Non	Changements dans MySQL 8.0.26

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Le paramètre `general_logging` est activé	La journalisation générale est activée pour votre instance de base de données. Ce paramètre est utile pour résoudre les problèmes liés à la base de données. Cependant, l'activation de la journalisation générale augmente le nombre d' I/O opérations et l'espace de stockage alloué, ce qui peut entraîner des conflits et une dégradation des performances. Vérifiez vos exigences en matière d'utilisation de la journalisation générale. Nous vous recommandons de définir la valeur du paramètre `general_logging` sur `0`.	Vérifiez vos exigences en matière d'utilisation de la journalisation générale. Si ce n'est pas obligatoire, nous vous recommandons de définir la valeur du paramètre `general_logging` sur `0`.	Non	Présentation des journaux de base de données Aurora MySQL

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Cluster de bases de données sous-provisionné pour la charge de travail de lecture	Nous vous recommandons d'ajouter une instance de base de données de lecture à votre cluster de bases de données ayant les mêmes taille et classe d'instance que l'instance de base de données de rédacteur figurant dans le cluster. La configuration actuelle comporte une instance de base de données dont la charge de base de données est constamment élevée, principalement en raison d'opérations de lecture. Répartissez ces opérations en ajoutant une autre instance de base de données au cluster et en dirigeant la charge de travail de lecture vers le point de terminaison en lecture seule du	Ajoutez une instance de base de données de lecteur au cluster.	Non	Ajout de réplicas Aurora à un cluster de bases de données Gestion des performances et dimensionnement des clusters de bases de données Aurora Tarification d'Amazon RDS

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
	cluster de bases de données.			
Instance RDS sous-provisionnée pour la capacité de mémoire du système	Nous vous recommandons de régler vos requêtes de manière à utiliser moins de mémoire ou d'utiliser un type d'instance de base de données avec une plus grande quantité de mémoire allouée. Lorsque la mémoire de l'instance est insuffisante, les performances de la base de données sont affectées.	Utilisation d'une instance de base de données avec une capacité de mémoire supérieure	Oui	Mise à l'échelle verticale et horizontale de votre instance Amazon RDS sur le blog de AWS base de données Types d'instance Amazon RDS Tarification d'Amazon RDS

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Instance RDS sous-provisionnée pour la capacité du CPU du système	Nous vous recommandons de régler vos requêtes pour utiliser moins de CPU ou de modifier votre instance de base de données pour utiliser une classe d'instance de base de données avec un v alloué plus élevéCPUs. Les performances de la base de données peuvent diminuer lorsque le processeur d'une instance de base de données est insuffisant.	Utilisation d'une instance de base de données avec une capacité d'UC supérieure	Oui	Mise à l'échelle verticale et horizontale de votre instance Amazon RDS sur le blog de AWS base de données Types d'instance Amazon RDS Tarification d'Amazon RDS

Type	Description	Recommandation	Durée d'indisponibilité requise	Informations supplémentaires
Les ressources RDS n'utilisent pas correctement le regroupement de connexions	Nous vous recommandons d'activer Proxy Amazon RDS pour regrouper et partager efficacement les connexions de base de données existantes. Si vous utilisez déjà un proxy pour votre base de données, configurez-le correctement pour améliorer le regroupement des connexions et l'équilibrage de charge entre plusieurs instances de base de données. Le proxy RDS peut contribuer à réduire le risque d'épuisement des connexions et de durée d'indisponibilité tout en améliorant la disponibilité et la capacité de mise à l'échelle.	Activation du proxy RDS ou modification de votre configuration de proxy existante	Non	Mise à l'échelle verticale et horizontale de votre instance Amazon RDS sur le blog de AWS base de données Proxy Amazon RDS pour Aurora Tarification de Proxy Amazon RDS

Affichage des métriques dans la console Amazon RDS

Amazon RDS s'intègre à Amazon CloudWatch pour afficher une variété de métriques de cluster de bases de données Aurora dans la console RDS. Certaines métriques s'appliquent au niveau du cluster, tandis que d'autres s'appliquent au niveau de l'instance.Pour obtenir des descriptions des métriques au niveau de l'instance et du cluster, voir Référence des métriques pour Amazon Aurora.

Pour votre cluster de bases de données Aurora, les catégories de métriques suivantes sont surveillées :

- CloudWatch : affiche les métriques Amazon CloudWatch pour Aurora auxquelles vous pouvez accéder depuis la console RDS. Vous pouvez également accéder à ces métriques depuis la console CloudWatch. Chaque métrique inclut un graphique affichant la métrique supervisée sur une période donnée. Pour obtenir une liste des métriques CloudWatch, consultez CloudWatch Métriques Amazon pour Amazon Aurora.
- Surveillance améliorée : affiche un récapitulatif des métriques du système d'exploitation lorsque la surveillance améliorée est activée pour le cluster de bases de données Aurora. RDS fournit les métriques de la surveillance améliorée à votre compte Amazon CloudWatch Logs. Chaque métrique du système d'exploitation comprend un graphique montrant la métrique surveillée sur un intervalle spécifique. Pour avoir une présentation, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée. Pour obtenir une liste des métriques de la surveillance améliorée, consultez Métriques du système d'exploitation dans la surveillance améliorée.
- Liste de processus du système d'exploitation : affiche les détails de chaque processus s'exécutant dans votre cluster de bases de données.
- Performance Insights : ouvre le tableau de bord Amazon RDS Performance Insights pour une instance de base de données dans votre cluster de bases de données Aurora. Performance Insights n'est pas pris en charge au niveau du cluster.Pour une présentation de Performance Insights, consultez Surveillance de la charge de la base de données avec Performance Insights sur . Pour obtenir une liste des métriques de Performance Insights, consultez Métriques Amazon CloudWatch pour Analyse des performances d'Amazon RDS.

Amazon RDS fournit désormais une vue consolidée des métriques Performance Insights et CloudWatch dans le tableau de bord Performance Insights. Performance Insights doit être activé pour que votre cluster de bases de données puisse utiliser cette vue. Vous pouvez choisir la nouvelle vue de surveillance dans l'onglet Surveillance ou Performance Insights dans le panneau de navigation.

Pour consulter les instructions relatives au choix de cette vue, consultez Affichage des métriques combinées avec le tableau de bord Performance Insights.

Affichage des métriques combinées avec le tableau de bord Performance Insights

Important

AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l'expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L'API Performance Insights continuera d'exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.

Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.

Si vous n'effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode Standard de Database Insights, vous risquez de perdre l'accès à l'historique des données de performance au-delà de 7 jours et de ne pas être en mesure d'utiliser les plans d'exécution et les fonctionnalités d'analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.

Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance à grande échelle. Pour plus d'informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

Amazon RDS fournit une vue consolidée des statistiques de performance et des CloudWatch indicateurs de votre instance de base de données dans le tableau de bord Performance Insights. Vous pouvez utiliser le tableau de bord préconfiguré ou créer un tableau de bord personnalisé.
Le tableau de bord préconfiguré fournit les métriques les plus couramment utilisées pour aider à diagnostiquer les problèmes de performances d'un moteur de base de données. Sinon, vous pouvez créer un tableau de bord personnalisé avec les métriques pour un moteur de base de données qui

répond à vos exigences en matière d'analyse. Utilisez ensuite ce tableau de bord pour toutes les instances de base de données de ce type de moteur de base de données dans votre AWS compte.

Vous pouvez choisir la nouvelle vue de surveillance dans l'onglet Surveillance ou Performance Insights dans le panneau de navigation.

Performance Insights doit être activé pour votre cluster de bases de données pour que vous puissiez afficher les métriques combinées dans le tableau de bord Performance Insights. Pour plus d'informations sur l'activation de Performance Insights, consultez Activation ou désactivation de l'Analyse des performances pour Aurora.

Dans les sections suivantes, vous apprendrez à afficher les CloudWatch statistiques et les statistiques relatives aux Performances.

Rubriques

Choix de la nouvelle vue de surveillance dans l'onglet Surveillance

Important

AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l'expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L'API Performance Insights continuera d'exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.

Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.

Si vous n'effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode

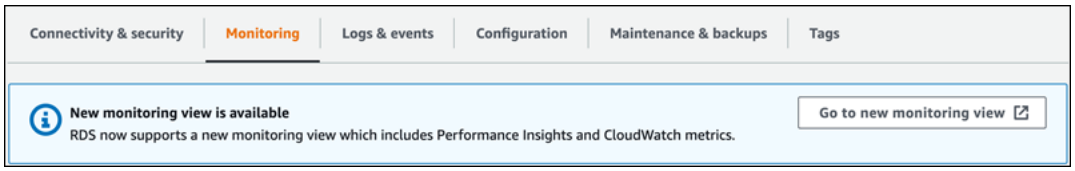
Standard de Database Insights, vous risquez de perdre l'accès à l'historique des données de performance au-delà de 7 jours et de ne pas être en mesure d'utiliser les plans d'exécution et les fonctionnalités d'analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.
Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance à grande échelle. Pour plus d'informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

Depuis la console Amazon RDS, vous pouvez choisir la nouvelle vue de surveillance pour consulter les Performances Insights et CloudWatch les métriques de votre instance de base de données.

Pour choisir la nouvelle vue de surveillance dans l'onglet Surveillance :

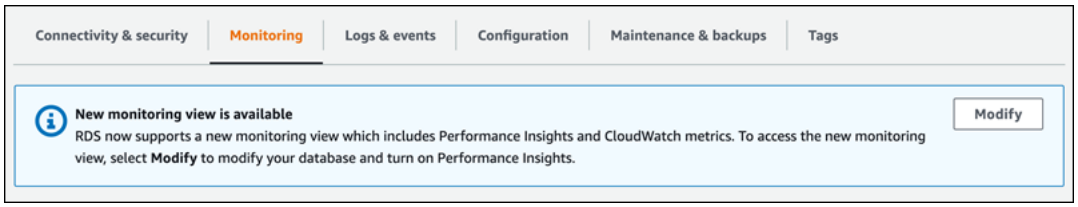
1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation de gauche, sélectionnez Bases de données.
3. Sélectionnez le cluster de bases de données Aurora que vous souhaitez surveiller.
4. Faites défiler vers le bas et choisissez l'onglet Surveillance.

   Une bannière apparaît avec l'option permettant de choisir la nouvelle vue de surveillance. L'exemple suivant montre la bannière pour choisir la nouvelle vue de surveillance.



5. Choisissez Accéder à une nouvelle vue de surveillance pour ouvrir le tableau de bord Performance Insights contenant des informations sur les performances et des CloudWatch métriques pour votre de base de données.
6. (Facultatif) Si Performance Insights est désactivé pour votre instance de base de données, une bannière apparaît avec la possibilité de modifier votre instance de base de données et d'activer Performance Insights.

L’exemple suivant montre la bannière permettant de modifier l’instance de base de données dans l’onglet Surveillance.



Choisissez Modifier pour modifier votre instance de base de données et activer Performance Insights. Pour plus d’informations sur l’activation de Performance Insights, consultez Activation ou désactivation de l’Analyse des performances pour Aurora.

Choix de la nouvelle vue de surveillance avec Performance Insights

Important

AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l’expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L’API Performance Insights continuera d’exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.

Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.

Si vous n’effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode Standard de Database Insights, vous risquez de perdre l’accès à l’historique des données de performance au-delà de 7 jours et de ne pas être en mesure d’utiliser les plans d’exécution et les fonctionnalités d’analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.

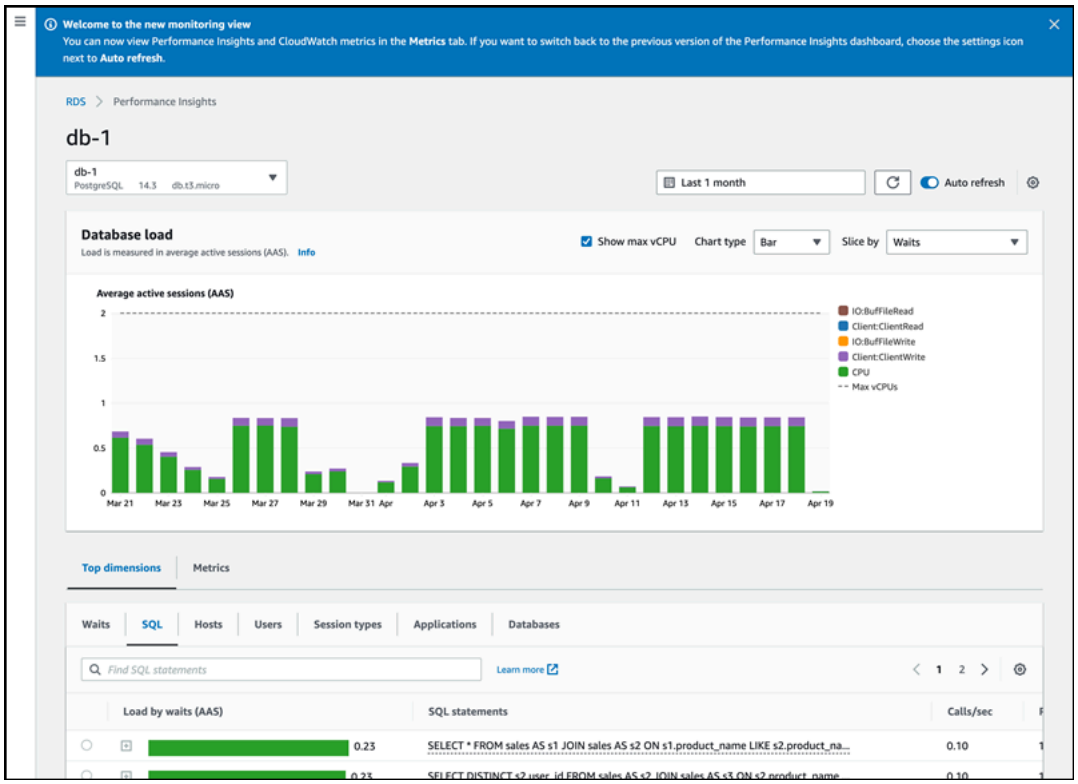
Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance

à grande échelle. Pour plus d'informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

Depuis la console Amazon RDS, vous pouvez choisir la nouvelle vue de surveillance pour consulter les Performances Insights et CloudWatch les métriques de votre instance de base de données.

Pour choisir la nouvelle vue de surveillance avec Performance Insights dans le panneau de navigation :

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données pour consulter le tableau de bord Performance Insights qui affiche à la fois les statistiques de performance et CloudWatch les métriques de votre instance de base de données.



Création d’un tableau de bord personnalisé avec Performance Insights

Important

AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l’expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L’API Performance Insights continuera d’exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.

Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.

Si vous n’effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode Standard de Database Insights, vous risquez de perdre l’accès à l’historique des données de performance au-delà de 7 jours et de ne pas être en mesure d’utiliser les plans d’exécution et les fonctionnalités d’analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.

Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance à grande échelle. Pour plus d’informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l’aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

Dans la nouvelle vue de surveillance, vous pouvez créer un tableau de bord personnalisé avec les métriques dont vous avez besoin pour répondre à vos exigences d’analyse.

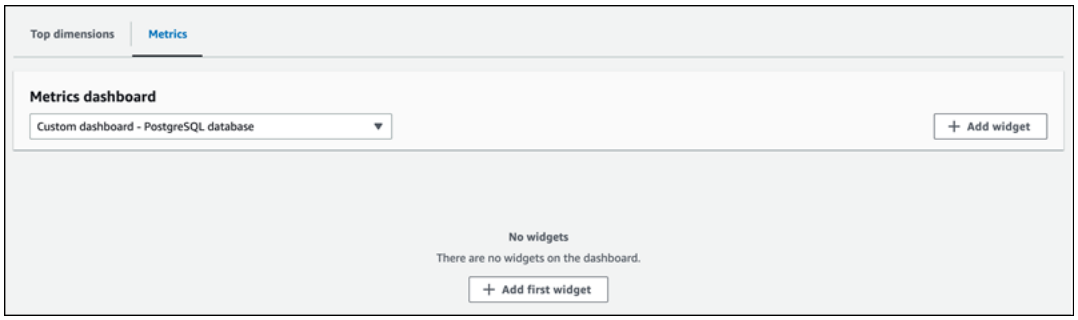
Vous pouvez créer un tableau de bord personnalisé en sélectionnant Performance Insights et CloudWatch les métriques pour votre instance de base de données. Vous pouvez utiliser ce tableau de bord personnalisé pour d'autres instances de base de données du même type de moteur de base de données dans votre AWS compte.

> **Note**
>
> Le tableau de bord personnalisé prend en charge jusqu'à 50 métriques.

Utilisez le menu des paramètres du widget pour modifier ou supprimer le tableau de bord et pour déplacer ou redimensionner la fenêtre du widget.

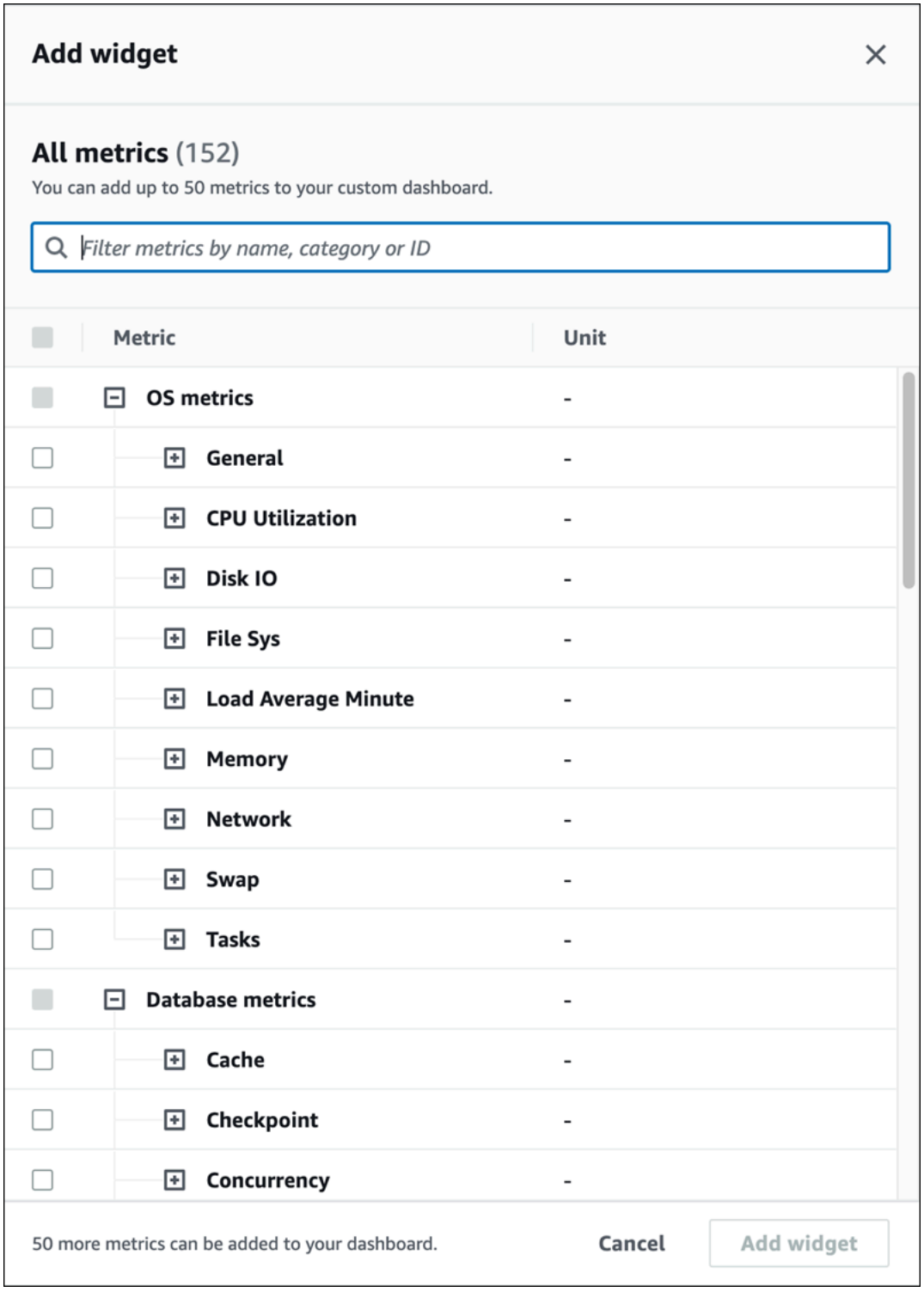
Pour créer un tableau de bord personnalisé avec Performance Insights dans le panneau de navigation :

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données.
4. Faites défiler la fenêtre vers le bas jusqu'à l'onglet Métriques.
5. Sélectionnez le tableau de bord personnalisé dans la liste déroulante. L'exemple suivant montre la création du tableau de bord personnalisé.



6. Choisissez Ajouter un widget pour ouvrir la fenêtre Ajouter un widget. Vous pouvez ouvrir et afficher les métriques du système d'exploitation (OS), les métriques de base de données et CloudWatch les métriques disponibles dans la fenêtre.

   L'exemple suivant montre la fenêtre Ajouter un widget avec les métriques.



7. Sélectionnez les métriques que vous souhaitez afficher dans le tableau de bord et sélectionnez Ajouter un widget. Vous pouvez utiliser le champ de recherche pour trouver une métrique spécifique.

   Les métriques sélectionnées s'affichent dans votre tableau de bord.

8. (Facultatif) Si vous souhaitez modifier ou supprimer votre tableau de bord, choisissez l'icône des paramètres en haut à droite du widget, puis sélectionnez l'une des actions suivantes dans le menu.

   - Modifier : modifiez la liste des métriques dans la fenêtre. Sélectionnez Mettre à jour le widget après avoir sélectionné les métriques pour votre tableau de bord.
   - Supprimer : supprime le widget. Sélectionnez Supprimer dans la fenêtre de confirmation.

Choix du tableau de bord préconfiguré avec Performance Insights

> **Important**
>
> AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l'expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L'API Performance Insights continuera d'exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.
>
> Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.
>
> Si vous n'effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode Standard de Database Insights, vous risquez de perdre l'accès à l'historique des données de performance au-delà de 7 jours et de ne pas être en mesure d'utiliser les plans d'exécution et les fonctionnalités d'analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.
>
> Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance à grande échelle. Pour plus d'informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

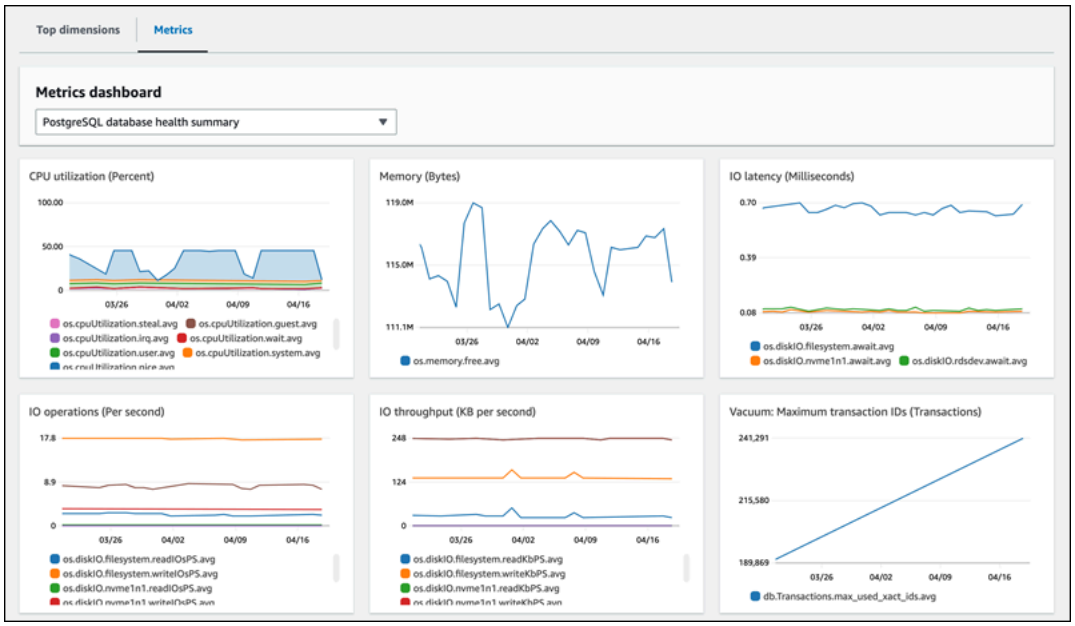
Le tableau de bord préconfiguré vous permet d'afficher les métriques les plus couramment utilisées. Ce tableau de bord permet de diagnostiquer les problèmes de performances à l'aide d'un moteur de base de données et de réduire le temps de restauration moyen de quelques heures à quelques minutes.

> **Note**
>
> Ce tableau de bord ne peut pas être modifié.

Pour choisir le tableau de bord préconfiguré avec Performance Insights dans le panneau de navigation :

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données.
4. Faites défiler la fenêtre vers le bas jusqu'à l'onglet Métriques.
5. Sélectionnez un tableau de bord préconfiguré dans la liste déroulante.

   Vous pouvez afficher les métriques pour l'instance de base de données dans le tableau de bord. L'exemple suivant présente un tableau de bord de métriques préconfiguré.



Surveillance des métriques Amazon Aurora avec Amazon CloudWatch

Amazon CloudWatch est un référentiel de métriques. Le référentiel collecte et traite les données brutes de Amazon Aurora en métriques lisibles et disponibles presque en temps réel. Pour obtenir la liste complète des métriques Amazon Aurora envoyées à CloudWatch, consultez Référence des métriques pour Amazon Aurora.

Pour analyser et résoudre les problèmes de performances de vos bases de données à grande échelle, utilisez CloudWatch Database Insights.

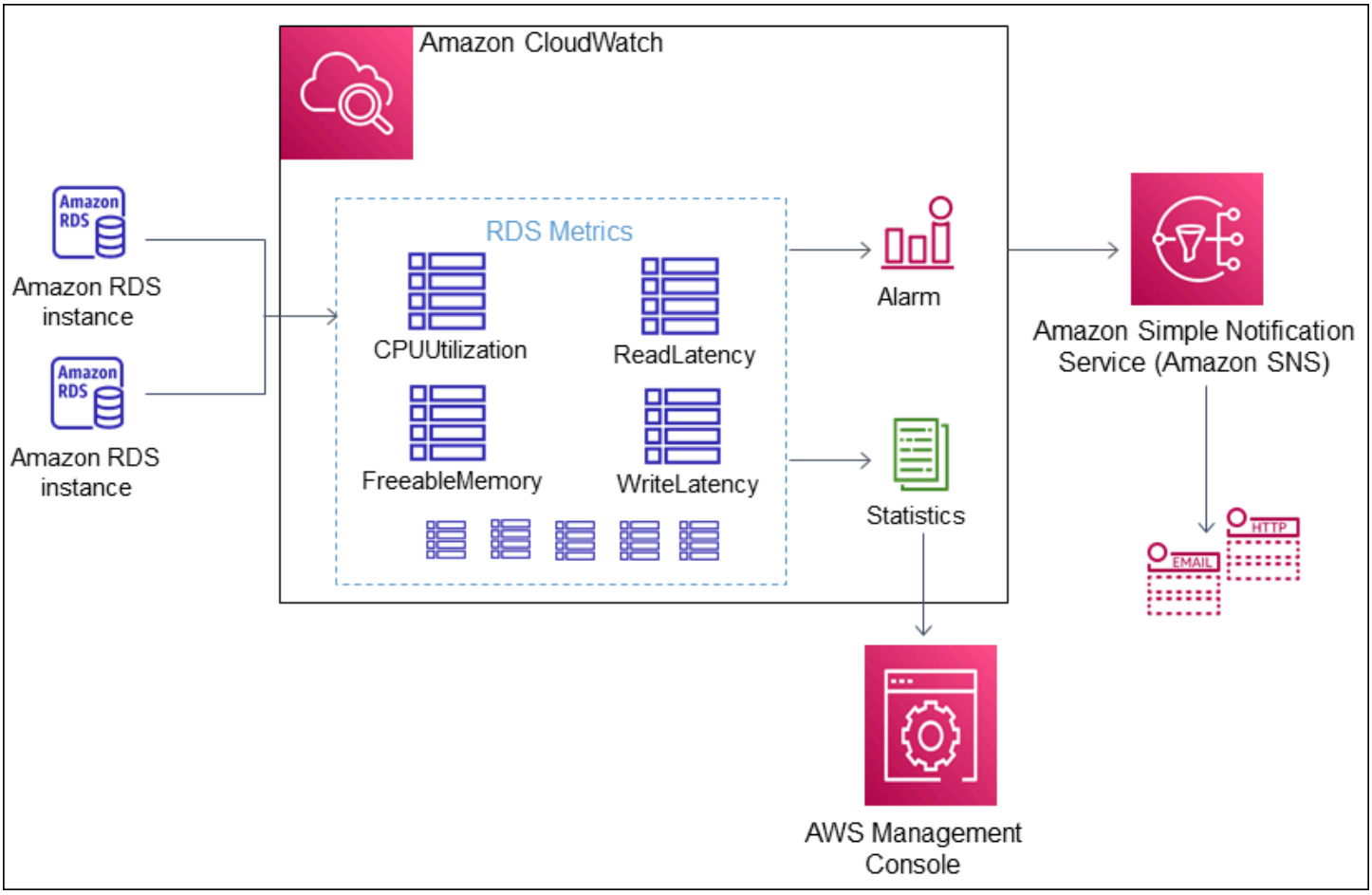
Rubriques

Présentation d’Amazon Aurora et d’Amazon CloudWatch

Par défaut, Amazon Aurora envoie automatiquement les données des métriques à CloudWatch toutes les minutes. Par exemple, la métrique `CPUUtilization` enregistre le pourcentage d’utilisation du CPU pour une instance de base de données au fil du temps. Les points de données d’une durée de 60 secondes (1 minute) sont disponibles pendant 15 jours Cela signifie que pouvez accéder aux informations historiques et voir la façon dont votre service ou application web s’exécute.

Vous pouvez désormais exporter les tableaux de bord de métriques Performance Insights d’Amazon RDS vers Amazon CloudWatch. Vous pouvez exporter les tableaux de bord de métriques préconfigurés ou personnalisés sous forme de nouveau tableau de bord ou les ajouter à un tableau de bord CloudWatch existant. Le tableau de bord exporté est visible dans la console CloudWatch. Pour plus d’informations sur la procédure d’exportation des tableaux de bord de métriques Performance Insights vers CloudWatch, consultez Exportation de métriques Performance Insights vers CloudWatch.

Comme le montre le diagramme suivant, vous pouvez configurer des alarmes pour vos métriques CloudWatch. Par exemple, vous pouvez créer une alarme qui signale que l’utilisation du CPU d’une instance est supérieure à 70 %. Vous pouvez configurer Amazon Simple Notification Service pour envoyer un e-mail lorsque le seuil est dépassé.



Amazon RDS publie les types de métriques suivants sur Amazon CloudWatch :

- Métriques Aurora au niveau des clusters et des instances

  Pour obtenir un tableau de ces métriques, consultez CloudWatch Métriques Amazon pour Amazon Aurora.

- Métriques de Performance Insights

  Pour obtenir un tableau de ces métriques, consultez Métriques Amazon CloudWatch pour Analyse des performances d'Amazon RDS et Métrique de compteur de Performance Insights.

- Métriques de surveillance améliorées (publiées dans les journaux d'Amazon CloudWatch)

  Pour obtenir un tableau de ces métriques, consultez Métriques du système d'exploitation dans la surveillance améliorée.

- Métriques d'utilisation pour les quotas du service Amazon RDS dans votre Compte AWS

Pour obtenir un tableau de ces métriques, consultez Mesures CloudWatch d'utilisation d'). Pour plus d'informations sur les quotas Amazon RDS, consultez Quotas et contraintes pour Amazon Aurora.

Pour plus d'informations sur CloudWatch, consultez Qu'est-ce qu'Amazon CloudWatch ? dans le Guide de l'utilisateur Amazon CloudWatch. Pour plus d'informations sur la conservation des métriques CloudWatch, consultez Conservation des métriques.

Affichage des métriques du cluster de bases de données dans la console CloudWatch et AWS CLI

Vous trouverez ci-dessous des détails sur l'affichage des métriques pour votre instance de base de données à l'aide de CloudWatch. Pour plus d'informations sur la surveillance des métriques du système d'exploitation de votre instance de base de données en temps réel à l'aide de CloudWatch Logs, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.

Lorsque vous utilisez des ressources Amazon Aurora, Amazon Aurora envoie des métriques et des dimensions à Amazon CloudWatch toutes les minutes.

Vous pouvez désormais exporter les tableaux de bord de métriques Performance Insights d'Amazon RDS vers Amazon CloudWatch et consulter ces métriques dans la console CloudWatch. Pour plus d'informations sur la procédure d'exportation des tableaux de bord de métriques Performance Insights vers CloudWatch, consultez Exportation de métriques Performance Insights vers CloudWatch.

Utilisez les procédures suivantes pour afficher les métriques d'Amazon Aurora dans la console et l'interface de ligne de commande CloudWatch.

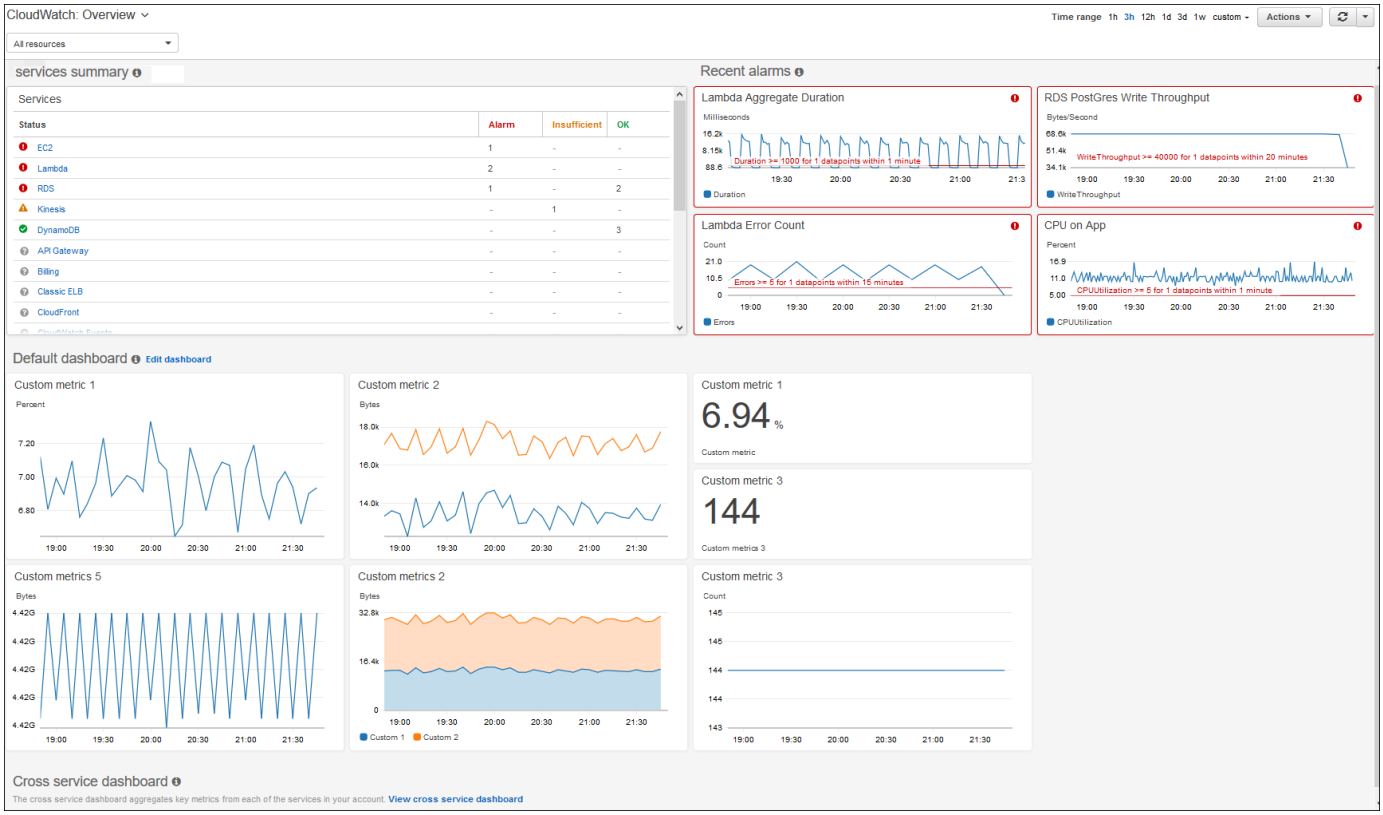
Console

Pour afficher des métriques à l'aide de la console Amazon CloudWatch

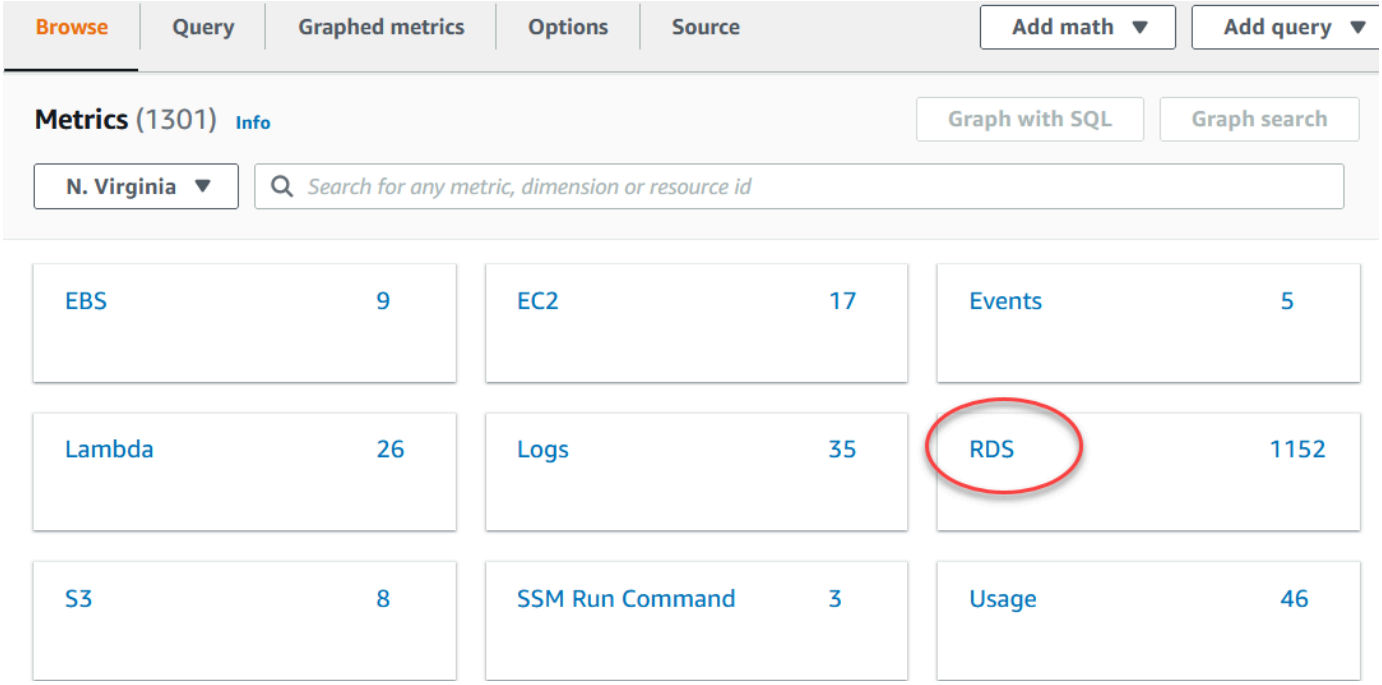
Les métriques sont d'abord regroupées par espace de noms de service, puis par les différentes combinaisons de dimension au sein de chaque espace de noms.

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.

   La page d'accueil de présentation de CloudWatch s'affiche.

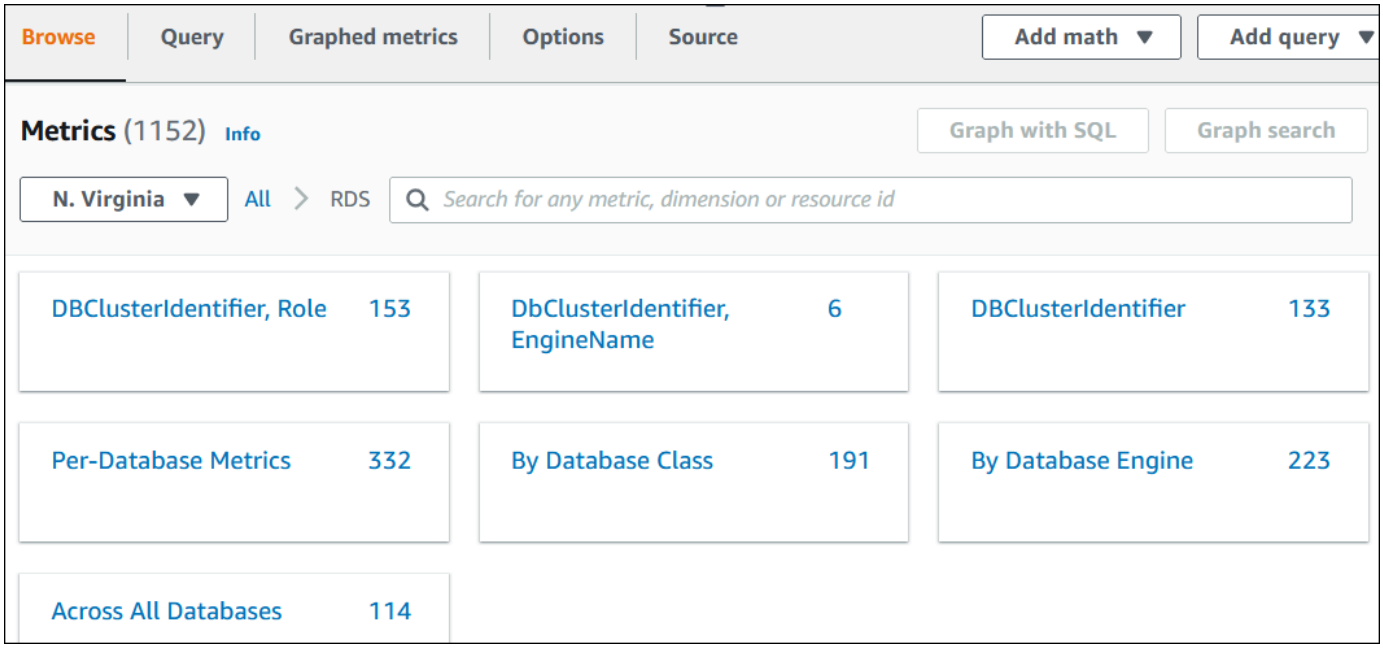


2. Si nécessaire, changez la Région AWS. Dans la barre de navigation, choisissez la Région AWS où se trouvent vos ressources AWS. Pour plus d'informations, consultez Régions et points de terminaison.
3. Dans le panneau de navigation, choisissez Metrics (Métriques), All metrics (Toutes les métriques).

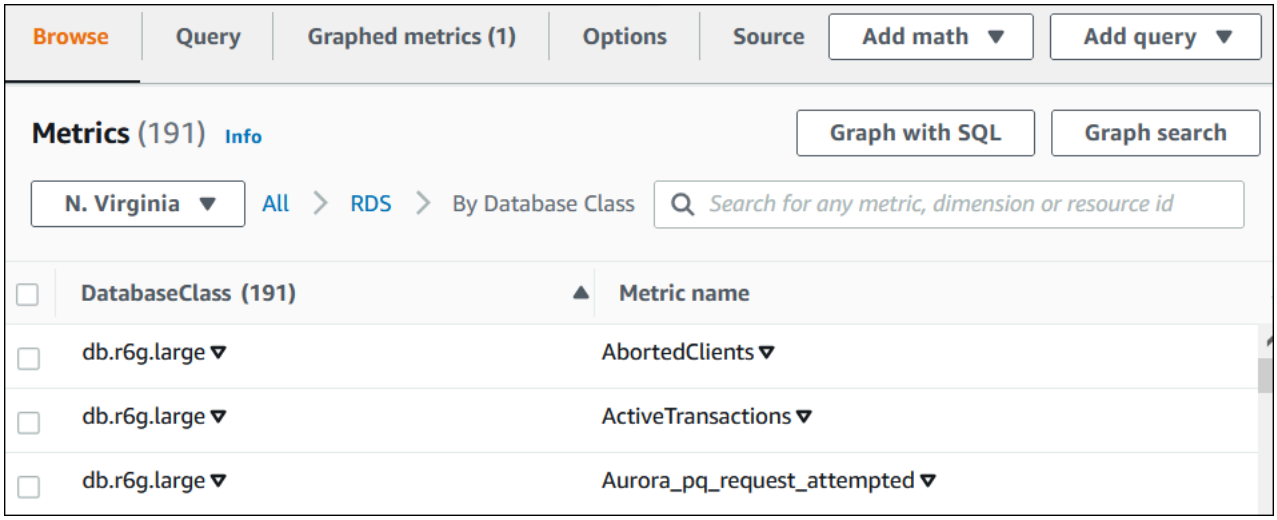


4. Faites défiler vers le bas et choisissez l'espace de nom de métrique RDS.

   La page affiche les dimensions Amazon Aurora. Pour une liste complète de ces dimensions, consultez CloudWatch Dimensions Amazon pour .



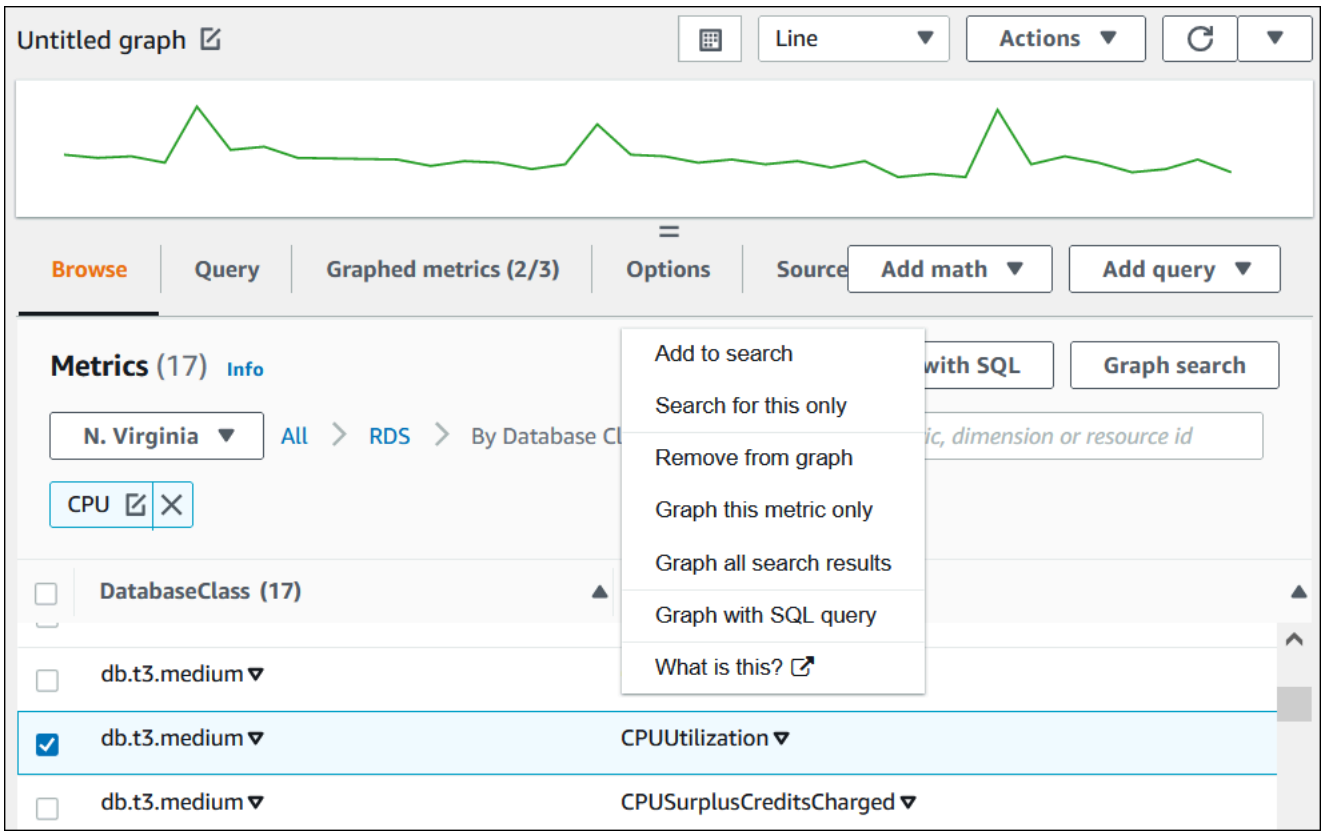
5. Sélectionnez une dimension de métrique, par exemple By Database Class (Par classe de base de données).



6. Effectuez l'une des actions suivantes :

   - Pour trier les métriques, utilisez l'en-tête de colonne.
   - Pour représenter graphiquement une métrique, cochez la case en regard de la métrique.
   - Pour filtrer par ressource, sélectionnez l'ID de ressource, puis Add to search (Ajouter à la recherche).
   - Pour filtrer par métrique, choisissez le nom de la métrique, puis Ajouter à la recherche.

   L'exemple suivant filtre sur la classe db.t3.medium et représente graphiquement la métrique CPUUtilization.



Vous trouverez des détails sur l'analyse de l'utilisation des ressources pour Aurora PostgreSQL avec des métriques CloudWatch. Pour plus d'informations, consultez Utilisation des CloudWatch métriques Amazon pour analyser l'utilisation des ressources pour Aurora PostgreSQL.

AWS CLI

Pour obtenir des informations sur les métriques à l'aide de l'AWS CLI, utilisez la commande CloudWatch `list-metrics`. Dans l'exemple indiqué ci-dessous, vous répertoriez toutes les métriques dans l'espace de noms `AWS/RDS`.

```
aws cloudwatch list-metrics --namespace AWS/RDS
```

Pour obtenir des données de métriques, utilisez la commande `get-metric-data`.

L'exemple suivant illustre l'obtention des statistiques `CPUUtilization` pour l'instance `my-instance` au cours d'une période de 24 heures donnée, avec une précision d'un niveau de 5 minutes.

Créez un fichier JSON `CPU_metric.json` avec le contenu suivant.

```
{
   "StartTime" : "2023-12-25T00:00:00Z",
   "EndTime" : "2023-12-26T00:00:00Z",
   "MetricDataQueries" : [{
     "Id" : "cpu",
     "MetricStat" : {
    "Metric" : {
        "Namespace" : "AWS/RDS",
        "MetricName" : "CPUUtilization",
        "Dimensions" : [{ "Name" : "DBInstanceIdentifier" , "Value" : my-instance}]
    },
       "Period" : 360,
       "Stat" : "Minimum"
     }
   }]
}
```

Example

Pour Linux, macOS ou Unix :

```
aws cloudwatch get-metric-data \
    --cli-input-json file://CPU_metric.json
```

Pour Windows :

```
aws cloudwatch get-metric-data ^
     --cli-input-json file://CPU_metric.json
```

L'exemple de sortie apparaît comme suit :

```
{
    "MetricDataResults": [
        {
            "Id": "cpu",
            "Label": "CPUUtilization",
            "Timestamps": [
                "2023-12-15T23:48:00+00:00",
                "2023-12-15T23:42:00+00:00",
                "2023-12-15T23:30:00+00:00",
                "2023-12-15T23:24:00+00:00",
```

```
                ...
            ],
            "Values": [
                13.299778337027714,
                13.677507543049558,
                14.24976250395827,
                13.02521708695145,
                ...
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

Pour plus d'informations, consultez Obtention de statistiques pour une métrique dans le Guide de l'utilisateur Amazon CloudWatch.

Exportation de métriques Performance Insights vers CloudWatch

Performance Insights vous permet d'exporter le tableau de bord de métriques préconfiguré ou personnalisé de votre instance de base de données vers Amazon CloudWatch. Vous pouvez exporter le tableau de bord des métriques en tant que nouveau tableau de bord ou l'ajouter à un tableau de bord CloudWatch existant. Si vous choisissez d'ajouter le tableau de bord à un tableau de bord CloudWatch existant, vous pouvez créer une étiquette d'en-tête afin que les métriques apparaissent dans une section distincte du tableau de bord CloudWatch.

Vous pouvez également afficher le tableau de bord de métriques exporté dans la console CloudWatch. Si vous ajoutez de nouvelles métriques à un tableau de bord de métriques Performance Insights après l'avoir exporté, vous devez à nouveau exporter ce tableau de bord pour afficher les nouvelles métriques dans la console CloudWatch.

Vous pouvez également sélectionner un widget de métriques dans le tableau de bord Performance Insights et afficher les données de métriques dans la console CloudWatch.

Pour plus d'informations sur l'affichage des métriques dans la console CloudWatch, consultez Affichage des métriques du cluster de bases de données dans la console CloudWatch et AWS CLI.

Dans les sections suivantes, exportez les métriques Performance Insights vers CloudWatch sous forme de tableau de bord, qu'il soit nouveau ou existant, puis consultez ces métriques dans CloudWatch.

Rubriques

- Exportation de métriques Performance Insights sous forme de nouveau tableau de bord vers CloudWatch
- Ajout de métriques Performance Insights à un tableau de bord CloudWatch existant
- Affichage d'un widget de métriques Performance Insights dans CloudWatch

Exportation de métriques Performance Insights sous forme de nouveau tableau de bord vers CloudWatch

Choisissez un tableau de bord de métriques préconfiguré ou personnalisé dans le tableau de bord Performance Insights et exportez-le en tant que nouveau tableau de bord vers CloudWatch. Vous pouvez afficher le tableau de bord exporté dans la console CloudWatch.

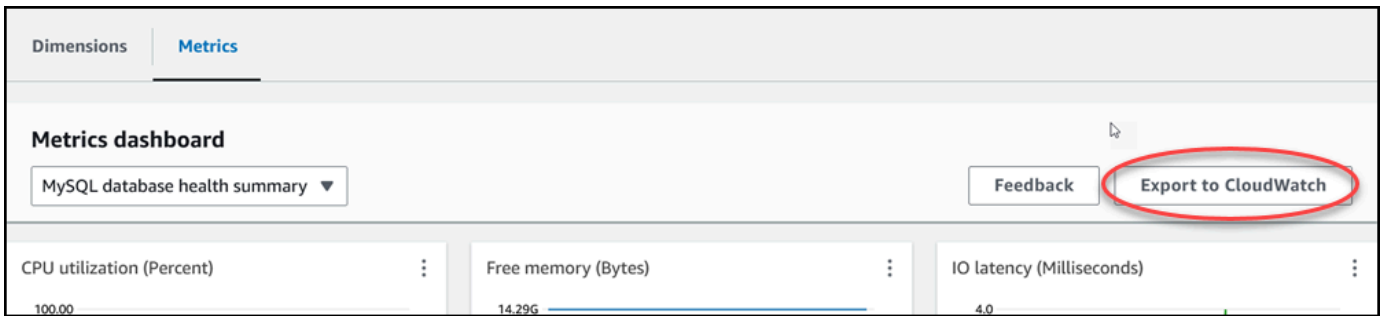
Pour exporter un tableau de bord de métriques Performance Insights sous forme de nouveau tableau de bord vers CloudWatch

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données.

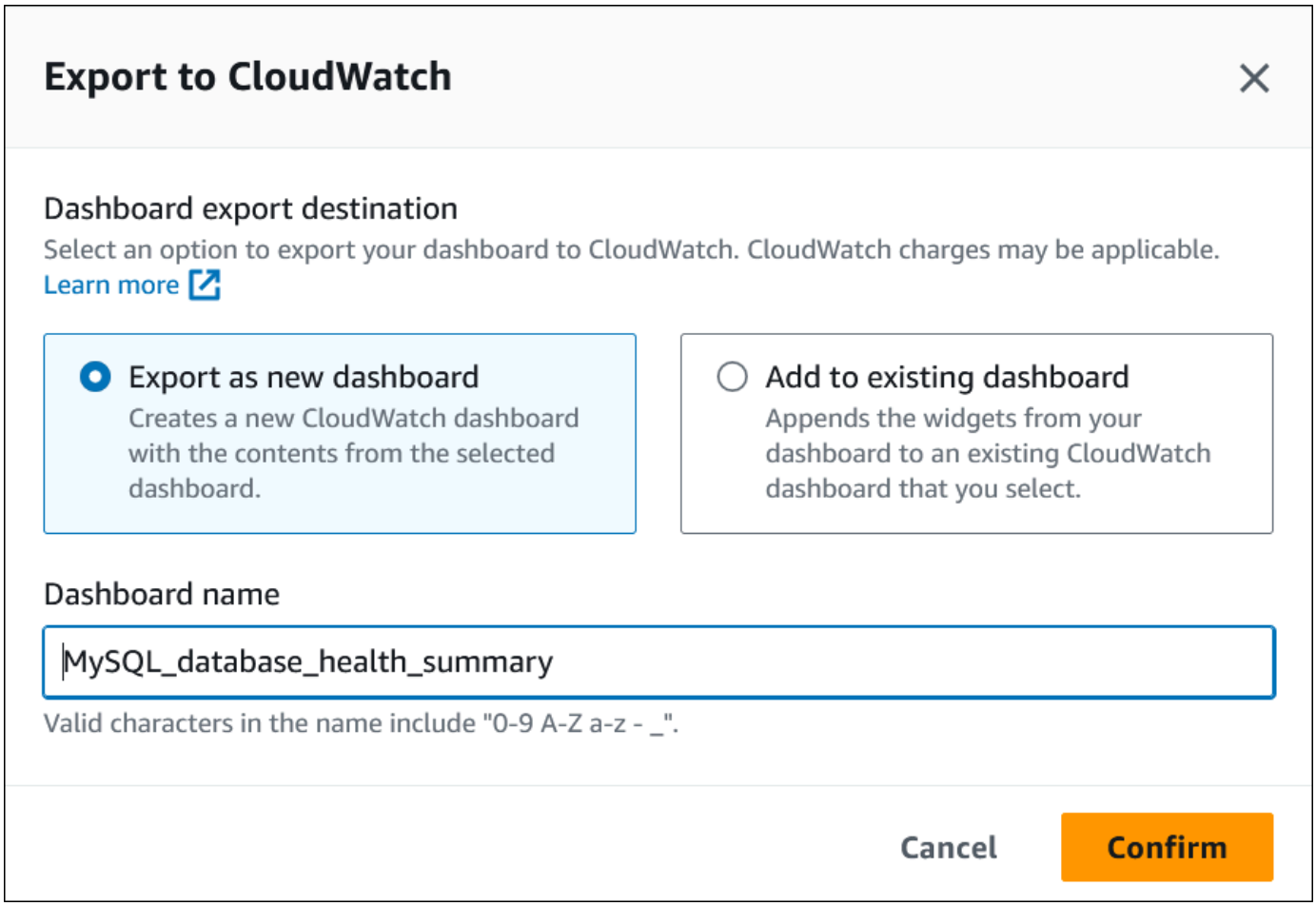
   Le tableau de bord Performance Insights correspondant à cette instance de base de données s'affiche.
4. Faites défiler l'écran vers le bas et choisissez Métriques.

   Par défaut, le tableau de bord préconfiguré avec les métriques Performance Insights s'affiche.
5. Choisissez un tableau de bord préconfiguré ou personnalisé, puis choisissez Exporter vers CloudWatch.

   La fenêtre Exporter vers CloudWatch apparaît.

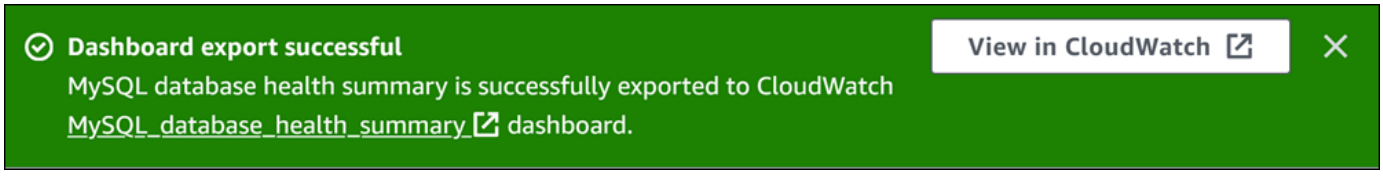


6. Choisissez Exporter en tant que nouveau tableau de bord.



7. Entrez le nom du nouveau tableau de bord dans le champ Nom du tableau de bord et choisissez Confirmer.

   Une bannière affiche un message une fois l'exportation du tableau de bord réussie.



8. Cliquez sur le lien ou sur Voir dans CloudWatch dans la bannière pour afficher le tableau de bord des métriques dans la console CloudWatch.

Ajout de métriques Performance Insights à un tableau de bord CloudWatch existant

Ajoutez un tableau de bord de métriques préconfiguré ou personnalisé à un tableau de bord CloudWatch existant. Vous pouvez ajouter une étiquette au tableau de bord de métriques pour qu'il apparaisse dans une section distincte du tableau de bord CloudWatch.

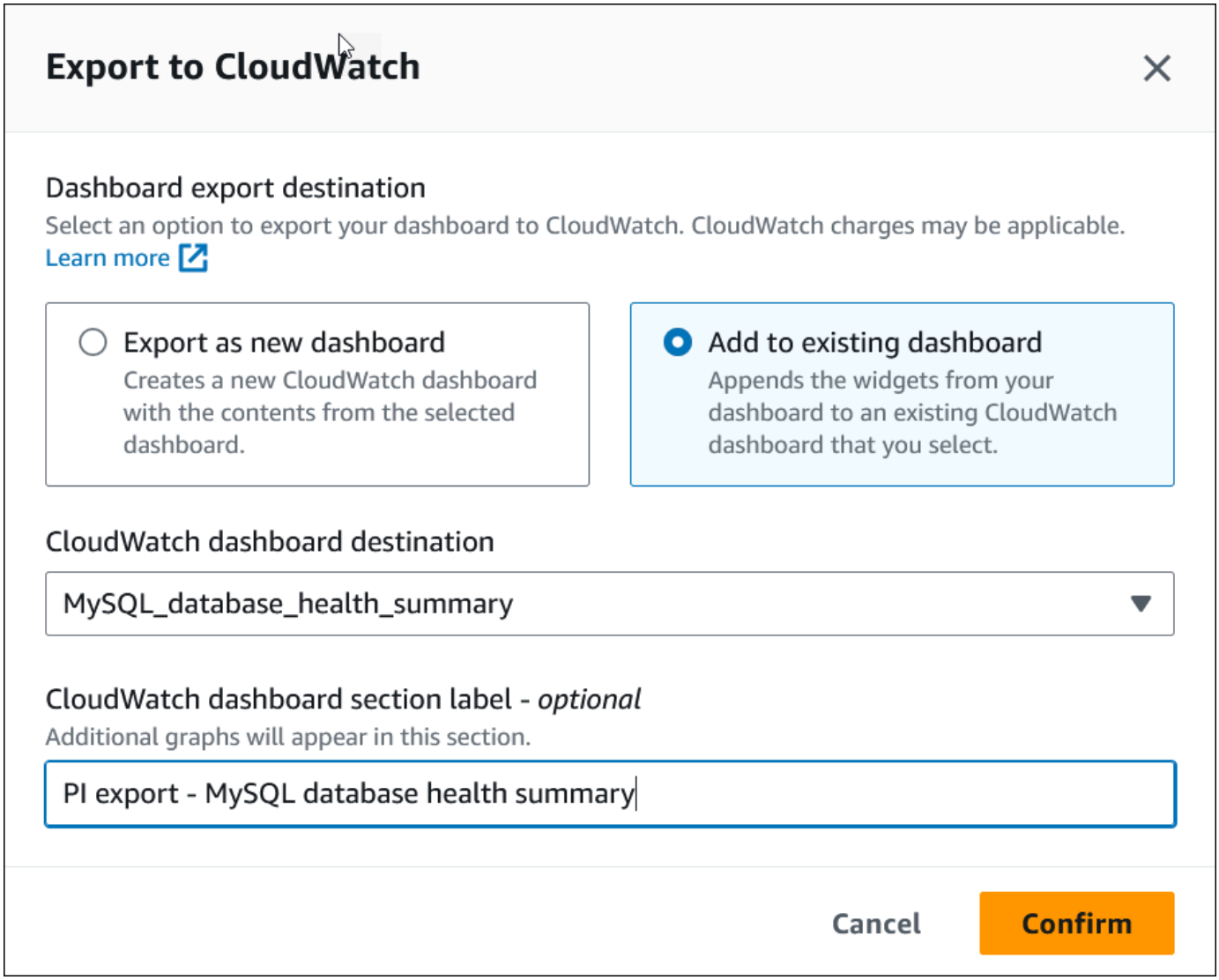
Pour exporter les métriques vers un tableau de bord CloudWatch existant

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données.

   Le tableau de bord Performance Insights correspondant à cette instance de base de données s'affiche.
4. Faites défiler l'écran vers le bas et choisissez Métriques.

   Par défaut, le tableau de bord préconfiguré avec les métriques Performance Insights s'affiche.
5. Choisissez le tableau de bord préconfiguré ou personnalisé, puis choisissez Exporter vers CloudWatch.

   La fenêtre Exporter vers CloudWatch apparaît.
6. Choisissez Ajouter au tableau de bord existant.



7. Spécifiez la destination et l'étiquette du tableau de bord, puis choisissez Confirmer.

   - Destination du tableau de bord CloudWatch : choisissez un tableau de bord CloudWatch existant.
   - Étiquette de section du tableau de bord CloudWatch (facultatif) : saisissez un nom pour les métriques Performance Insights qui apparaîtront dans cette section du tableau de bord CloudWatch.

   Une bannière affiche un message une fois l'exportation du tableau de bord réussie.

8. Cliquez sur le lien ou sur Voir dans CloudWatch dans la bannière pour afficher le tableau de bord des métriques dans la console CloudWatch.

Affichage d’un widget de métriques Performance Insights dans CloudWatch

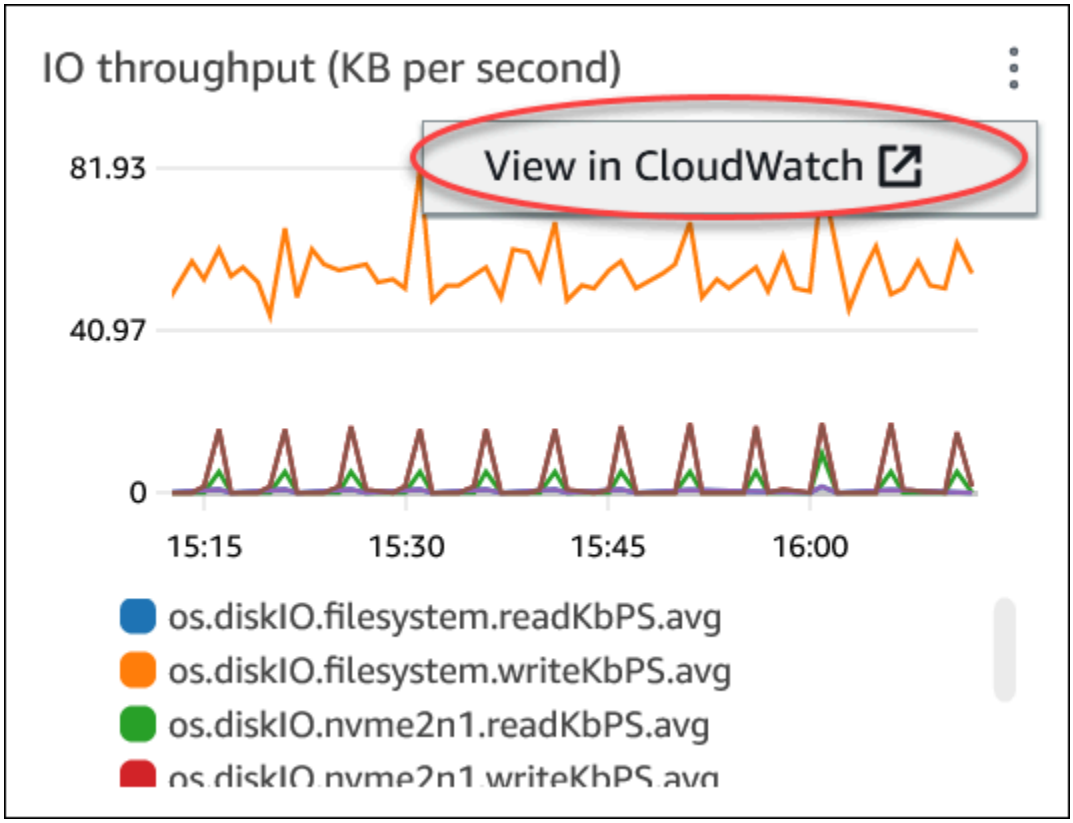
Sélectionnez un widget de métriques Performance Insights dans le tableau de bord Analyse des performances d’Amazon RDS et consultez les données des métriques dans la console CloudWatch.

Pour exporter un widget de métriques et afficher les données des métriques dans la console CloudWatch

1. Ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données.

   Le tableau de bord Performance Insights correspondant à cette instance de base de données s’affiche.
4. Faites défiler la page vers le bas jusqu’à Métriques.

   Par défaut, le tableau de bord préconfiguré avec les métriques Performance Insights s’affiche.
5. Choisissez un widget de métriques, puis sélectionnez Voir dans CloudWatch dans le menu.



   Les données des métriques s’affichent dans la console CloudWatch.

Création d’alarmes CloudWatch pour surveiller Amazon Aurora

Créez une alarme CloudWatch qui envoie un message Amazon SNS lorsque l’alarme change de statut. Une alarme surveille une seule métrique pendant la période que vous spécifiez. Elle peut également réaliser une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes. Cette action est une notification envoyée vers une rubrique Amazon SNS ou une stratégie Amazon EC2 Auto Scaling.

Les alarmes appellent les actions pour les changements d’état soutenus uniquement. Les alarmes CloudWatch ne déclenchent pas d’actions simplement parce qu’elles se trouvent dans un état particulier. L’état doit avoir changé et avoir été maintenu pendant un nombre de périodes spécifié.

> **Note**
>
> Pour Aurora, utilisez les métriques de rôle `WRITER` ou `READER` pour configurer des alarmes plutôt que de s’appuyer sur des métriques associées à des instances de base de données spécifiques. Les rôles d’instance de base de donnéesAurora peuvent changer de rôles au fil du temps. Vous pouvez trouver ces métriques basées sur les rôles dans la consoleCloudWatch.
> L’Auto Scaling Aurora définit automatiquement les alarmes en fonction des métriques de rôle`READER`. Pour plus d’informations sur l’Auto Scaling Aurora, consultez Amazon Aurora Auto Scaling avec des réplicas Aurora.

Vous pouvez utiliser la fonction mathématique de métrique DB_PERF_INSIGHTS dans la console CloudWatch afin d’interroger Amazon RDS sur les métriques de compteur Performance Insights. La fonction DB_PERF_INSIGHTS inclut également la métrique DBLoad à des intervalles inférieurs à la minute. Vous pouvez également définir des alarmes CloudWatch sur ces métriques.

Pour en savoir plus sur la création d’une alarme, consultez Création d’une alarme sur les métriques de compteur Performance Insights à partir d’une base de données AWS.

Pour définir une alarme à l’aide de l AWS CLI

- Appelez `put-metric-alarm`. Pour plus d’informations, consultez la référence de la commande AWS CLI.

Pour définir une alarme à l’aide de l’API CloudWatch

- Appelez `PutMetricAlarm`. Pour plus d’informations, consultez la Référence de l’API Amazon CloudWatch.

Pour plus d’informations sur la configuration des rubriques Amazon SNS et la création d’alarmes, consultez Utilisation des alarmes Amazon CloudWatch.

Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights

Surveillez la charge de base de données (charge BD) de votre flotte d'instances de base de données Amazon Aurora à l'aide de Database Insights. La charge de la base de données mesure le niveau d'activité de la session de votre base de données. Vous pouvez utiliser Database Insights pour analyser et résoudre les problèmes liés aux performances de mise à l'échelle de vos bases de données Amazon Aurora.

Sur le tableau de bord de Performance Insights, vous pouvez visualiser la charge de base de données de votre flotte de bases de données et la filtrer par attentes, instructions SQL, hôtes ou utilisateurs.

Par défaut, RDS active le mode Standard de Database Insights pour vos bases de données Amazon Aurora Lorsque vous activez le mode Avancé de Database Insights pour un cluster de bases de données, RDS active Database Insights pour chaque instance de base de données du cluster.

Pour plus d'informations sur l'utilisation de Database Insights dans la console Amazon CloudWatch, consultez CloudWatch Database Insights dans le Amazon CloudWatch User Guide.

Tarification

Pour plus d'informations sur la tarification, consultez Tarification Amazon CloudWatch.

Rubriques

- Prise en charge du moteur de base de données, de la région et de la classe d'instance Amazon Aurora pour Database Insights
- Activation du mode Avancé de Database Insights pour Amazon Aurora
- Activation du mode Standard de Database Insights pour Amazon Aurora
- Configuration de votre base de données pour surveiller les requêtes SQL lentes avec Database Insights pour Amazon Aurora
- Considérations relatives à Database Insights pour Amazon Aurora

Prise en charge du moteur de base de données, de la région et de la classe d’instance Amazon Aurora pour Database Insights

Le tableau suivant fournit les moteurs de base de données Amazon Aurora qui prennent en charge Database Insights.

Moteur de base de données Amazon Aurora	Versions et régions soumises à la gestion des versions du moteur	Restrictions de classe d’instance
Amazon Aurora MySQL-Compatible Edition	Pour plus d’informations sur la disponibilité des versions et des régions de Database Insights avec Aurora MySQL, consultez Performance Insights avec Aurora MySQL.	Database Insights présente les restrictions de classe de moteur suivantes : • db.t2 : non pris en charge • db.t3 : non pris en charge • db.t4g.micro et db.t4g.small : non pris en charge
Amazon Aurora PostgreSQL-Compatible Edition	Pour plus d’informations sur la disponibilité des versions et des régions de Database Insights avec Aurora PostgreSQL, consultez Performance Insights avec Aurora PostgreSQL.	Ne s’applique pas
Base de données Aurora PostgreSQL Limitless	Pour plus d’informations sur l’utilisation de Database Insights avec la base de données Aurora PostgreSQL Limitless, consultez Surveillance d’Aurora PostgreSQL Limitless Database avec CloudWatch Database Insights.	Ne s’applique pas

Database Insights prend en charge Amazon Aurora Serverless v2.

Prise en charge des moteurs de base de données, des régions et des classes d’instance Amazon Aurora, pour les fonctionnalités Database Insights

Le tableau suivant fournit les moteurs de base de données Amazon Aurora qui prennent en charge les fonctionnalités Database Insights.

Fonctionnalité	Niveau de tarification	Régions prises en charge	Moteurs de base de données pris en charge	Classes d’instance prises en charge
Statistiques SQL pour Performance Insights	Tous	Tous	Tous	Tous
Analyse des performances de base de données pour une période donnée	Niveau payant uniquement	Tous	Tous	Toutes sauf db.serverless (Aurora Serverless v2)
Affichage des recommandations proactives de Performance Insights	Niveau payant uniquement	• USA Est (Ohio) • USA Est (Virginie du Nord) • USA Ouest (Californie du Nord) • USA Ouest (Oregon) • Asie-Pacifique (Mumbai) • Asie-Pacifique (Séoul)	Tous	Toutes sauf db.serverless (Aurora Serverless v2)

Fonctionnalité	Niveau de tarification	Régions prises en charge	Moteurs de base de données pris en charge	Classes d'instance prises en charge
		• Asie-Pacifique (Singapour) • Asie-Pacifique (Sydney) • Asie-Pacifique (Tokyo) • Canada (Centre) • Europe (Francfort) • Europe (Irlande) • Europe (Londres) • Europe (Paris) • Europe (Stockholm) • Amérique du Sud (São Paulo)		

Prise en charge des régions Amazon Aurora pour Database Insights

Aurora prend en charge Database Insights dans les Régions AWS suivantes.

- USA Est (Virginie du Nord)
- USA Est (Ohio)
- USA Ouest (Californie du Nord)
- USA Ouest (Oregon)

- Afrique (Le Cap)
- Asie-Pacifique (Hong Kong)
- Asie-Pacifique (Hyderabad)
- Asie-Pacifique (Jakarta)
- Asie-Pacifique (Malaisie)
- Asie-Pacifique (Melbourne)
- Asie-Pacifique (Mumbai)
- Asie-Pacifique (Osaka)
- Asia Pacific (Seoul)
- Asie-Pacifique (Singapour)
- Asie-Pacifique (Sydney)
- Asie-Pacifique (Tokyo)
- Canada (Centre)
- Canada-Ouest (Calgary)
- Europe (Francfort)
- Europe (Irlande)
- Europe (Londres)
- Europe (Milan)
- Europe (Paris)
- Europe (Espagne)
- Europe (Stockholm)
- Europe (Zurich)
- Israël (Tel Aviv)
- Moyen-Orient (Bahreïn)
- Moyen-Orient (EAU)
- Amérique du Sud (São Paulo)
- AWS GovCloud (US, côte est)
- AWS GovCloud (US-West)

Activation du mode Avancé de Database Insights pour Amazon Aurora

Pour activer le mode Avancé de Database Insights pour Amazon Aurora, utilisez les procédures qui suivent.

Activation du mode Avancé de Database Insights lors de la création d'une d'un cluster de bases de données

Activez le mode Avancé de Database Insights lors de la création d'une base de données pour Amazon Aurora.

Console

Dans la console, vous pouvez activer le mode Avancé de Database Insights lorsque vous créez un cluster de bases de données. Les paramètres de Database Insights s'appliquent à toutes les instances de base de données de votre cluster de bases de données.

Pour activer le mode Avancé de Database Insights lors de la création d'un cluster de bases de données à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez Create database (Créer une base de données).
4. Dans la section Database Insights, sélectionnez le mode Avancé. Choisissez ensuite les options suivantes :
   - Conservation – Durée de conservation des données de Performance Insights. La période de conservation doit être de 15 mois pour le mode avancé de Database Insights.
   - AWS KMS key – Spécifiez votre clé KMS. Performance Insights chiffre toutes les données potentiellement sensibles à l'aide votre clé KMS. Les données sont chiffrées en transit et au repos. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.
5. Choisissez Create database (Créer une base de données).

AWS CLI

Pour activer le mode Avancé de Database Insights lors de la création d'un cluster de bases de données, appelez la commande create-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :

- `--database-insights-mode advanced` pour activer le mode Avancé de Database Insights.
- `--engine` : moteur de base de données pour le cluster de bases de données.
- `--db-cluster-identifier` : identifiant du cluster de bases de données .
- `--enable-performance-insights` pour activer Performance Insights pour Database Insights.
- `--performance-insights-retention-period` : période de conservation des données de votre cluster de bases de données . Pour activer Database Insights, la période de conservation doit être d'au moins 465 jours.

Dans l'exemple suivant, le mode Avancé de Database Insights est activé lors de la création d'un cluster de bases de données.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
    --database-insights-mode advanced \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Pour Windows :

```
aws rds create-db-cluster ^
    --database-insights-mode advanced ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Pour activer le mode Avancé de Database Insights lors de la création d'un cluster de bases de données , définissez les paramètres suivants pour votre opération CreateDBCluster de l'API Amazon RDS.

- `DatabaseInsightsMode` sur `advanced`
- `EnablePerformanceInsights` sur `True`
- `PerformanceInsightsRetentionPeriod` sur au moins 465 jours

Activation du mode Avancé de Database Insights lors de la modification d'un cluster de bases de données

Activation de Database Insights lors de la modification d'une base de données pour Amazon Aurora. La modification d'un cluster de bases de données pour activer le mode Avancé de Database Insights ne provoque pas de durée d'indisponibilité.

> **Note**
>
> Pour activer Database Insights, il est nécessaire que chaque instance de base de données d'un cluster de bases de données ait les mêmes paramètres Performance Insights et Enhanced Monitoring.

Console

Dans la console, vous pouvez activer le mode Avancé de Database Insights lorsque vous modifiez un cluster de bases de données. Les paramètres de Database Insights s'appliquent à toutes les instances de base de données de votre cluster de bases de données.

Pour activer le mode Avancé de Database Insights lors de la modification d'un cluster de bases de données à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez un cluster de bases de données, puis choisissez Modifier.

4. Dans la section Database Insights, sélectionnez le mode Avancé. Choisissez ensuite les options suivantes :

   - Conservation – Durée de conservation des données de Performance Insights. La période de conservation doit être de 15 mois pour le mode avancé de Database Insights.
   - AWS KMS key – Spécifiez votre clé KMS. Performance Insights chiffre toutes les données potentiellement sensibles à l'aide votre clé KMS. Les données sont chiffrées en transit et au repos. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.

5. Choisissez Continuer.
6. Pour Scheduling of Modifications (Planification des modifications), choisissez Appliquer immédiatement. Si vous choisissez Appliquer pendant la fenêtre de maintenance planifiée suivante, votre base de données ignore ce paramètre et active immédiatement le mode Avancé de Performance Insights.
7. Choisissez Modifier le cluster.

AWS CLI

Pour activer le mode Avancé de Database Insights lors de la modification d'un cluster de bases de données, appelez la commande modify-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :

- `--database-insights-mode advanced` pour activer le mode Avancé de Database Insights.
- `--db-cluster-identifier` : identifiant du cluster de bases de données .
- `--enable-performance-insights` pour activer Performance Insights pour Database Insights.
- `--performance-insights-retention-period` : période de conservation des données de votre cluster de bases de données. Pour activer le mode Avancé de Database Insights, la période de conservation doit être d'au moins 465 jours.

Dans l'exemple suivant, le mode Avancé Database Insights est activé lors de la modification d'un cluster de bases de données.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
```

```
    --database-insights-mode advanced \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --database-insights-mode advanced ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Pour activer le mode Avancé de Database Insights lorsque vous modifiez un cluster de bases de données, spécifiez les paramètres suivants pour votre opération d'API Amazon RDS ModifyDBCluster.

- `DatabaseInsightsMode` sur `advanced`
- `EnablePerformanceInsights` sur `True`
- `PerformanceInsightsRetentionPeriod` sur au moins 465 jours

Activation du mode Standard de Database Insights pour Amazon Aurora

Pour activer le mode Standard de Database Insights pour Amazon Aurora, suivez les procédures suivantes.

Activation du mode Standard de Database Insights lors de la création d'un cluster de bases de données

Activez le mode Standard de Database Insights lors de la création d'une base de données pour Amazon Aurora.

Console

Dans la console, vous pouvez activer le mode Standard de Database Insights lorsque vous créez un cluster de bases de données. Les paramètres de Database Insights s'appliquent à toutes les instances de base de données de votre cluster de bases de données.

Pour activer le mode Standard de Database Insights lors de la création d'un cluster de bases de données à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez Create database (Créer une base de données).
4. Dans la section Database Insights, sélectionnez le mode Standard. Choisissez ensuite l'une des options suivantes pour activer ou désactiver Performance Insights :
   - Pour désactiver Performance Insights, désélectionnez Activer Performance Insights.
   - Pour activer Performance Insights, sélectionnez Activer Performance Insights. Pour configurer Performance Insights, spécifiez les options suivantes :
     - Conservation – Durée de conservation des données de Performance Insights. La période de conservation doit être d'au moins 7 jours.
     - AWS KMS key – Spécifiez votre clé KMS. Performance Insights chiffre toutes les données potentiellement sensibles à l'aide votre clé KMS. Les données sont chiffrées en transit et au repos. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.
5. Choisissez Create database (Créer une base de données).

AWS CLI

Pour activer le mode Standard de Database Insights lors de la création d'un cluster de bases de données, appelez la commande create-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :

- `--database-insights-mode standard` pour activer le mode Standard de Database Insights.
- `--engine` : moteur de base de données pour le cluster de bases de données.
- `--db-cluster-identifier` : identifiant du cluster de bases de données .
- `--enable-performance-insights` ou `--no-enable-performance-insights` pour activer ou désactiver Performance Insights. Si vous spécifiez `--enable-performance-insights`, vous devez également spécifier `--performance-insights-retention-period`, la période de conservation des données de votre cluster de bases de données. La période de conservation doit être d'au moins 7 jours.

L’exemple suivant active le mode Standard de Database Insights et Performance Insights lors de la création d’un cluster de bases de données.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
    --database-insights-mode standard \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Pour Windows :

```
aws rds create-db-cluster ^
    --database-insights-mode standard ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

L’exemple suivant active le mode Standard de Database Insights et désactive Performance Insights lors de la création d’un cluster de bases de données.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
    --database-insights-mode standard \
    --engine aurora-postgresql \
    --db-cluster-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Pour Windows :

```
aws rds create-db-cluster ^
    --database-insights-mode standard ^
    --engine aurora-postgresql ^
    --db-cluster-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

Pour activer le mode Standard de Database Insight lors de la création d'un cluster de bases de données, spécifiez les paramètres suivants pour votre opération d'API Amazon RDS CreateDBCluster.

- `DatabaseInsightsMode` sur `standard`
- `EnablePerformanceInsights` sur `True` ou `False`. Si vous définissez `EnablePerformanceInsights` sur `True`, vous devez définir `PerformanceInsightsRetentionPeriod` sur au moins 7 jours.

Activation du mode Standard de Database Insights lors de la modification d'un cluster de bases de données

Activez le mode Standard de Database Insights lors de la modification d'une base de données pour Amazon Aurora. La modification d'un cluster de bases de données pour activer le mode Standard de Database Insights ne provoque pas de durée d'indisponibilité.

> **Note**
>
> Pour activer Database Insights, il est nécessaire que chaque instance de base de données d'un cluster de bases de données ait les mêmes paramètres Performance Insights et Enhanced Monitoring.

Console

Dans la console, vous pouvez activer le mode Standard de Database Insights lorsque vous modifiez un cluster de bases de données. Les paramètres de Database Insights s'appliquent à toutes les instances de base de données de votre cluster de bases de données.

Pour activer le mode Standard de Database Insights lors de la modification d'un cluster de bases de données à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez un cluster de bases de données, puis choisissez Modifier.

4. Dans la section Database Insights, sélectionnez le mode Standard. Ensuite, faites votre choix parmi les options suivantes :

   - Pour désactiver Performance Insights, désélectionnez Activer Performance Insights.
   - Pour activer Performance Insights, sélectionnez Activer Performance Insights. Pour configurer Performance Insights, spécifiez les options suivantes :
     - Conservation – Durée de conservation des données de Performance Insights. La période de conservation doit être d'au moins 7 jours.
     - AWS KMS key – Spécifiez votre clé KMS. Performance Insights chiffre toutes les données potentiellement sensibles à l'aide votre clé KMS. Les données sont chiffrées en transit et au repos. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.
5. Choisissez Continuer.
6. Pour Planification des modifications, choisissez Appliquer immédiatement. Si vous choisissez Appliquer lors de la prochaine fenêtre de maintenance planifiée, votre base de données ignore ce paramètre et active immédiatement le mode Standard de Database Insights.
7. Choisissez Modifier le cluster.

AWS CLI

Pour activer le mode Standard de Database Insights lors de la modification d'un cluster de bases de données, appelez la commande modify-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :

- `--database-insights-mode standard` pour activer le mode Standard de Database Insights.
- `--db-cluster-identifier` : identifiant du cluster de bases de données .
- `--enable-performance-insights` ou `--no-enable-performance-insights` pour activer ou désactiver Performance Insights. Si vous spécifiez `--enable-performance-insights`, vous devez également spécifier `--performance-insights-retention-period`, la période de conservation des données de votre cluster de bases de données . La période de conservation doit être d'au moins 7 jours.

L'exemple suivant active le mode Standard de Database Insights et active Performance Insights lors de la modification d'un cluster de bases de données.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 7
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 7
```

L'exemple suivant active le mode Standard de Database Insights et désactive Performance Insights lors de la modification d'un cluster de bases de données.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --no-enable-performance-insights
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --no-enable-performance-insights
```

RDS API

Pour activer le mode Standard de Database Insights lors de la modification d'un cluster de bases de données, spécifiez les paramètres suivants pour votre opération d'API Amazon RDS ModifyDBCluster.

- `DatabaseInsightsMode` sur `standard`

- `EnablePerformanceInsights` sur `True` ou `False`. Si vous définissez `EnablePerformanceInsights` sur `True`, vous devez définir `PerformanceInsightsRetentionPeriod` sur au moins 7 jours.

Configuration de votre base de données pour surveiller les requêtes SQL lentes avec Database Insights pour Amazon Aurora

Pour surveiller les requêtes SQL lentes pour votre base de données, vous pouvez utiliser la section Requêtes SQL lentes du tableau de bord Database Insights. Avant la configuration de votre base de données pour surveiller les requêtes SQL lentes, la section Requêtes SQL lentes est vide.

Pour plus d'informations sur la surveillance des requêtes SQL lentes dans le tableau de bord Database Insights, consultez Affichage du tableau de bord des instances de base de données pour CloudWatch Database Insights dans le Guide de l'utilisateur Amazon CloudWatch.

Pour configurer votre base de données afin de surveiller les requêtes SQL lentes avec Database Insights, procédez comme suit :

1. Activez les exportations de journaux vers CloudWatch Logs.
2. Créez ou modifiez le groupe de paramètres de cluster de bases de données pour votre cluster de bases de données.

Pour plus d'informations sur la configuration des exportations de journaux, consultez Publication des journaux de base de données sur Amazon CloudWatch Logs dans le Guide de l'utilisateur Amazon Aurora.

Pour créer ou modifier votre groupe de paramètres de cluster de bases de données, consultez les rubriques suivantes.

- Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora
- Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora

Amazon Aurora MySQL

Pour configurer votre cluster de bases de données Amazon Aurora MySQL afin de surveiller les requêtes SQL lentes, vous pouvez utiliser par exemple la combinaison de paramètres suivante :

- `slow_query_log` : défini sur `1`
- `long_query_time` : défini sur `1.0`
- `log_output` : défini sur `FILE`

Il s'agit d'une configuration possible. Pour un guide complet sur les paramètres du journal des requêtes lentes de MySQL et les options de configuration supplémentaires, consultez la documentation MySQL relative au journal des requêtes lentes.

Amazon Aurora PostgreSQL

Pour configurer votre cluster de bases de données Amazon Aurora PostgreSQL afin de surveiller les requêtes SQL lentes, vous pouvez utiliser par exemple la combinaison de paramètres suivante : Notez que la définition de ces paramètres peut réduire les performances de votre cluster de bases de données.

- `log_min_duration_statement` : défini sur `1000`
- `log_statement` : défini sur `none`
- `log_destination` : défini sur `stderr`

Il s'agit d'une configuration possible. Pour un guide complet sur les paramètres de journalisation de PostgreSQL et les options de configuration supplémentaires, consultez la documentation PostgreSQL relative à la configuration de journalisation.

> **Note**
>
> Pour Aurora MySQL, vous pouvez configurer le paramètre `long_query_time` avec une granularité d'une microseconde. Par exemple, vous pouvez définir ce paramètre sur `0.000001`. En fonction du nombre de requêtes sur l'instance de base de données, la valeur du paramètre `long_query_time` peut réduire les performances. Commencez par la valeur `1.0` et ajustez-la en fonction de votre charge de travail. Lorsque vous définissez ce paramètre sur `0`, Database Insights journalise toutes les requêtes.

Pour plus d'informations sur les journaux Aurora MySQL et Aurora PostgreSQL, consultez ce qui suit.

- Fichiers journaux de base de données Aurora MySQL

- Fichiers journaux de base de données Aurora PostgreSQL

Considérations relatives à Database Insights pour Amazon Aurora

Voici les considérations relatives à Database Insights pour Amazon Aurora.

- Vous ne pouvez pas gérer Database Insights pour une instance de base de données dans un cluster de bases de données.
- Pour activer le mode Avancé de Database Insights, vous devez activer Performance Insights et fixer la période de conservation de Performance Insights à au moins 465 jours (15 mois). Il n'y a aucun coût supplémentaire pour fixer la période de conservation de Performance Insights à 15 mois en plus du coût de Database Insights. Pour plus d'informations sur la tarification de Database Insights, consultez Tarification Amazon CloudWatch.
- Pour activer Database Insights, il est nécessaire que chaque instance de base de données d'un cluster de bases de données ait les mêmes paramètres Performance Insights et Enhanced Monitoring.
- La modification d'un cluster de bases de données pour activer l'un ou l'autre mode de Database Insights ne provoque pas de durée d'indisponibilité.

Surveillance de la charge de la base de données avec Performance Insights sur

Important

AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l'expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L'API Performance Insights continuera d'exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.

Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.

Si vous n'effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode Standard de Database Insights, vous risquez de perdre l'accès à l'historique des données de performance au-delà de 7 jours et de ne pas être en mesure d'utiliser les plans d'exécution et les fonctionnalités d'analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.

Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance à grande échelle. Pour plus d'informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

Performance Insights développe les fonctions de surveillance existantes d' Amazon Aurora pour illustrer et vous aider à analyser les performances de votre cluster. Avec le tableau de bord Performance Insights, vous pouvez visualiser la charge de la base de données de votre Charge de cluster Amazon Aurora et filtrer la charge par attentes, instructions SQL, hôtes ou utilisateurs. Pour plus d'informations sur l'utilisation de Performance Insights avec Amazon DocumentDB, consultez le Guide du développeur Amazon DocumentDB.

Rubriques

Présentation de Performance Insights sur Amazon Aurora

Important

AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l’expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L’API Performance Insights continuera d’exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.

Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.

Si vous n’effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode Standard de Database Insights, vous risquez de perdre l’accès à l’historique des données de performance au-delà de 7 jours et de ne pas être en mesure d’utiliser les plans d’exécution et les fonctionnalités d’analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.

Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance

à grande échelle. Pour plus d'informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

Par défaut, RDS active Performance Insights dans l'assistant de création de console pour les moteurs Amazon RDS. Si vous activez Performance Insights au niveau du cluster de bases de données, RDS active Performance Insights pour chaque instance de base de données du cluster. Si vous disposez de plusieurs bases de données sur une instance de base de données, Performance Insights regroupe les données de performance.

Vous trouverez un aperçu de Performance Insights pour Amazon Aurora dans la vidéo suivante.

Utilisation de Performance Insights pour analyser les performances de Amazon Aurora PostgreSQL

Rubriques

- Charge de base de données
- Utilisation maximale de l'UC
- Prise en charge de la classe d'instances, de la région et du moteur de base de données Amazon Aurora pour Performance Insights
- Tarification et conservation des données pour Performance Insights

Charge de base de données

Charge de base de données mesure le niveau d'activité de la session de votre base de données. `DBLoad` est la métrique clé de Performance Insights, et Performance Insights collecte la charge de base de données chaque seconde.

Rubriques

- Sessions actives
- Sessions actives en moyenne
- Exécutions actives moyennes
- Dimensions

Sessions actives

Une session de base de données représente le dialogue d'une application avec une base de données relationnelle. Une session active est une connexion qui a transmis du travail au moteur de base de données et qui attend une réponse.

Une session est active lorsqu'elle s'exécute sur le processeur (CPU) ou attend qu'une ressource devienne disponible pour pouvoir continuer. Par exemple, une session active peut attendre qu'une page (ou un bloc) soit lue en mémoire avant d'utiliser le processeur pendant la lecture des données de la page.

Sessions actives en moyenne

Les sessions actives en moyenne (AAS) représentent l'unité de la métrique `DBLoad` de Performance Insights. Elle mesure le nombre de sessions actives simultanément sur la base de données.

Toutes les secondes, Performance Insights échantillonne le nombre de sessions exécutant simultanément une requête. Pour chaque session active, Performance Insights collecte les données suivantes :

- Instruction SQL
- État de la session (en cours d'exécution sur le processeur ou en attente)
- Host (Hôte)
- Utilisateur exécutant le SQL

Performance Insights calcule les AAS en divisant le nombre total de sessions par le nombre d'échantillons pour une période déterminée. Par exemple, la table suivante présente 5 échantillons consécutifs d'une requête en cours d'exécution, prélevés à des intervalles d'une seconde.

Exemple	Nombre de sessions exécutant la requête	AAS	Calcul
1	2	2	2 sessions au total/1 échantillon
2	0	1	2 sessions au total/2 échantillons

Exemple	Nombre de sessions exécutant la requête	AAS	Calcul
3	4	2	6 sessions au total/3 échantillons
4	0	1.5	6 sessions au total/4 échantillons
5	4	2	10 sessions au total/5 échantillons

Dans l'exemple précédent, la charge de la base de données pour l'intervalle de temps était de 2 AAS. Cette mesure signifie qu'en moyenne, deux sessions étaient actives à la fois à n'importe quel moment au cours de la période où les cinq échantillons ont été prélevés.

Exécutions actives moyennes

La moyenne des exécutions actives (AAE) par seconde est liée à l'AAS. Pour calculer l'AAE, Performance Insights divise la durée totale d'exécution d'une requête par l'intervalle de temps. Le tableau suivant présente le calcul de l'AAE pour la même requête que dans le tableau précédent.

Temps écoulé (en secondes)	Durée totale d'exécution (en secondes)	AAE	Calcul
60	120	2	120 secondes d'exécution/60 secondes écoulées
120	120	1	120 secondes d'exécution/120 secondes écoulées
180	380	2.11	380 secondes d'exécution/180 secondes écoulées

Temps écoulé (en secondes)	Durée totale d'exécution (en secondes)	AAE	Calcul
240	380	1.58	380 secondes d'exécution/240 secondes écoulées
300	600	2	600 secondes d'exécution/300 secondes écoulées

Dans la plupart des cas, l'AAS et l'AAE d'une requête sont approximativement identiques. Cela dit, comme les données utilisées pour les calculs proviennent de sources différentes, les calculs varient souvent légèrement.

Dimensions

La métrique `db.load` est différente des autres métriques de séries chronologiques, car vous pouvez la décomposer en sous-composants appelés dimensions. Vous pouvez considérer les dimensions comme des catégories de « tranches » pour les différentes caractéristiques de la métrique `DBLoad`.

Lorsque vous diagnostiquez des problèmes de performances, les dimensions suivantes sont souvent les plus utiles :

Rubriques

- Événements d'attente
- Principaux éléments SQL

Pour obtenir la liste complète des dimensions des moteurs Aurora, consultez Charge de base de données tranchée par dimensions.

Événements d'attente

Un événement d'attente fait qu'une instruction SQL attend qu'un événement spécifique se produise avant de pouvoir continuer à s'exécuter. Les événements d'attente constituent une dimension (ou catégorie) importante pour la charge de la base de données, car ils indiquent les points de blocage du travail.

Chaque session active est soit en cours d'exécution au niveau du processeur soit en attente. Par exemple, les sessions sollicitent le processeur lorsqu'elles recherchent un tampon dans la mémoire, effectuent un calcul ou exécutent du code procédural. Lorsque les sessions ne sollicitent pas le processeur, c'est peut-être qu'elles attendent qu'un tampon de mémoire se libère, qu'un fichier de données soit lu ou qu'un journal soit écrit. Le temps que passe une session à attendre des ressources est autant de temps en moins qu'elle passe à s'exécuter au niveau du processeur.

Lorsque vous réglez une base de données, vous cherchez souvent à identifier les ressources que les sessions attendent. Par exemple, deux ou trois événements d'attente peuvent représenter 90 % de la charge de la base de données. Cette mesure signifie qu'en moyenne, les sessions actives passent la majeure partie de leur temps à attendre un petit nombre de ressources. Si vous trouvez la cause de ces attentes, vous pouvez tenter une solution.

Les événements d'attente varient en fonction du moteur de base de données :

- Pour obtenir la liste des événements d'attente courants pour Aurora MySQL, consultez Événements d'attente Aurora MySQL. Pour savoir comment régler l'utilisation de ces événements d'attente, consultez Réglage d'Aurora MySQL.
- Pour plus d'informations sur tous les événements d'attente MySQL, consultez Wait Event Summary Tables dans la documentation MySQL.
- Pour obtenir la liste des événements d'attente courants pour Aurora PostgreSQL, consultez Événements d'attente Amazon Aurora PostgreSQL. Pour savoir comment régler l'utilisation de ces événements d'attente, consultez Réglage des événements d'attente pour Aurora PostgreSQL.
- Pour plus d'informations sur tous les événements d'attente PostgreSQL, consultez The Statistics Collector > Wait Event tables dans la documentation de PostgreSQL.

Principaux éléments SQL

Là où les événements d'attente présentent des goulots d'étranglement, les principaux éléments SQL indiquent quelles requêtes contribuent le plus à la charge de la base de données. Par exemple, de nombreuses requêtes peuvent être en cours d'exécution sur la base de données, mais une seule d'entre elles peut consommer 99 % de la charge de la base de données. Dans ce cas, la charge élevée peut indiquer un problème avec la requête.

Par défaut, la console Performance Insights affiche les principales requêtes SQL qui contribuent à la charge de la base de données. La console affiche également des statistiques pertinentes pour chaque instruction. Pour diagnostiquer les problèmes de performances d'une instruction spécifique, vous pouvez examiner son plan d'exécution.

Utilisation maximale de l’UC

Dans le tableau de bord, le graphique Database load (Charge de base de données)collecte, regroupe et affiche les informations de session. Pour voir si les sessions actives dépassent l’utilisation maximale de l’UC, examinez leur relation sur la ligne Max vCPU (UC virtuelle max). Performance Insights détermine la valeur de vCPU maximum par le nombre de cœurs de vCPU (UC virtuelle) pour votre instance de base de données. Pour Aurora sans serveur v2, le vCPU maximum représente le nombre estimé de vCPU.

Un seul processus peut être exécuté sur un vCPU à la fois. Si le nombre de processus dépasse le nombre de vCPU, les processus sont mis en file d’attente. Lorsque le nombre de processus en file d’attente augmente, les performances de la base de données diminuent. Si la charge de la base de données est souvent au-dessus de la ligne Max vCPU (UC virtuelle max)et que l’état d’attente principal est CPU, cela signifie que l’UC est surchargée. Dans ce cas, vous pouvez décider de limiter les connexions à l’instance, de régler les requêtes SQL avec une charge d’UC élevée, ou envisager l’utilisation d’une classe d’instance plus grande. Quel que soit leur état d’attente, les instances élevées et régulières indiquent que des problèmes de goulots d’étranglement ou de conflits de ressources devront peut-être être résolus. Cela peut être vrai même si la charge de la base de données ne dépasse pas la ligne Max vCPU (UC virtuelle max).

Prise en charge de la classe d’instances, de la région et du moteur de base de données Amazon Aurora pour Performance Insights

Important

AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l’expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L’API Performance Insights continuera d’exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.

Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.

Si vous n’effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode

Standard de Database Insights, vous risquez de perdre l'accès à l'historique des données de performance au-delà de 7 jours et de ne pas être en mesure d'utiliser les plans d'exécution et les fonctionnalités d'analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.
Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance à grande échelle. Pour plus d'informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

Le tableau suivant fournit les moteurs de base de données Amazon Aurora qui prennent en charge l'analyse des performances.

Moteur de base de données Amazon Aurora	Versions et régions soumises à la gestion des versions du moteur	Restrictions de classe d'instance
Amazon Aurora MySQL-Compatible Edition	Pour plus d'informations sur la disponibilité des versions et des régions de Performance Insights avec Aurora MySQL, consultez Performance Insights avec Aurora MySQL.	Performance Insights présente les restrictions de classe de moteur suivantes : • db.t2 : non pris en charge • db.t3 : non pris en charge • db.t4g.micro et db.t4g.small : non pris en charge
Amazon Aurora PostgreSQL-Compatible Edition	Pour plus d'informations sur la disponibilité des versions et des régions de Performance Insights avec Aurora PostgreSQL, consultez Performance Insights avec Aurora PostgreSQL.	N/A

Prise en charge de la classe d'instances, de la région et du moteur de base de données Amazon Aurora pour les fonctionnalités d'analyse des performances

Le tableau suivant fournit les moteurs de base de données Amazon Aurora qui prennent en charge les fonctionnalités d'analyse des performances.

Fonctionnalité	Niveau de tarification	Régions prises en charge	Moteurs de base de données pris en charge	Classes d'instance prises en charge
Statistiques SQL pour Performance Insights	Tous	Tous	Tous	Tous
Analyse des performances de base de données pour une période donnée	Niveau payant uniquement	Tous	Tous	Toutes sauf db.serverless (Aurora Serverless v2)
Affichage des recommandations proactives de Performance Insights	Niveau payant uniquement	• USA Est (Ohio) • USA Est (Virginie du Nord) • USA Ouest (Californie du Nord) • USA Ouest (Oregon) • Asie-Pacifique (Mumbai) • Asie-Pacifique (Séoul) • Asie-Pacifique (Singapour)	Tous	Toutes sauf db.serverless (Aurora Serverless v2)

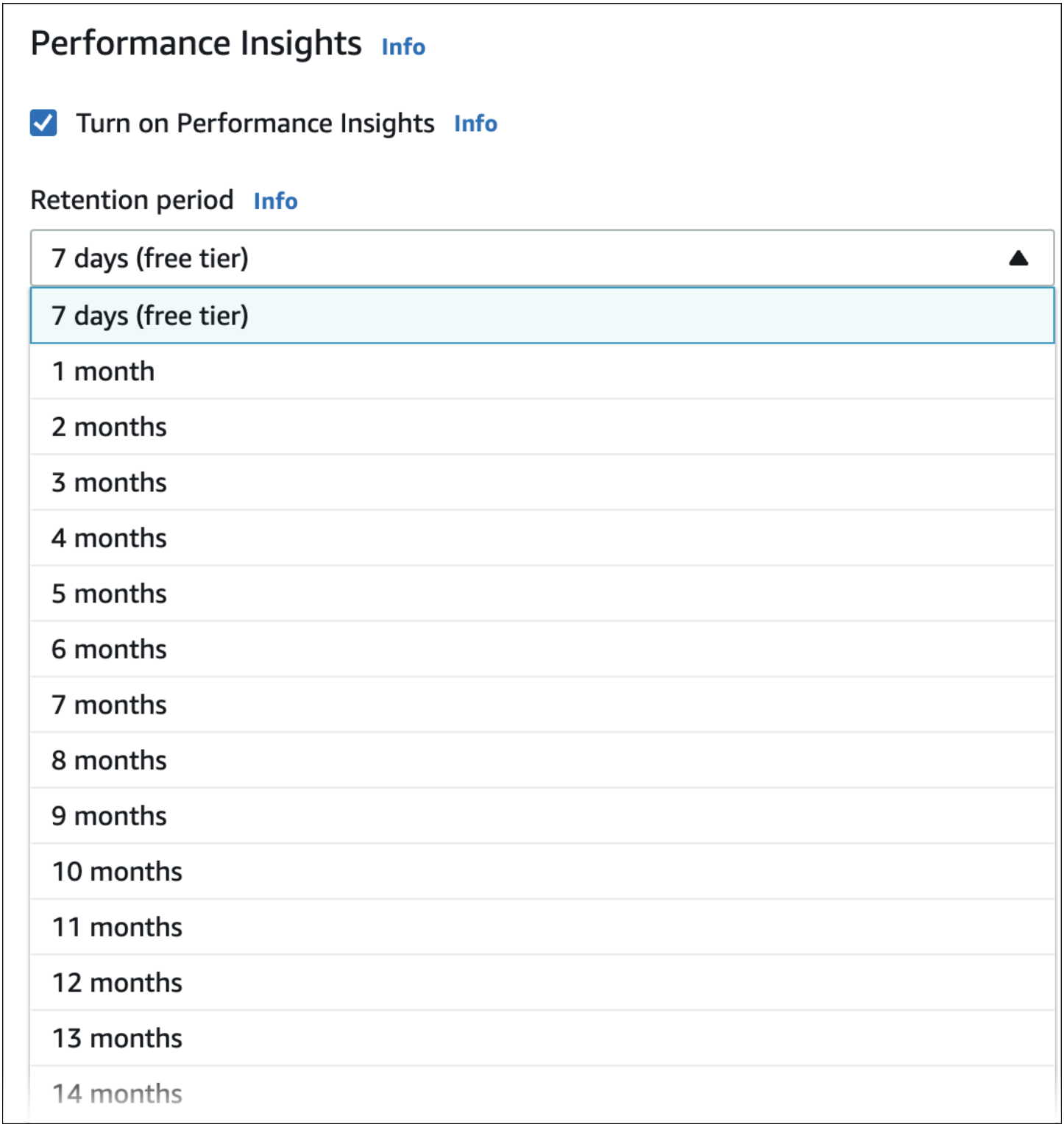
Fonctionnalité	Niveau de tarification	Régions prises en charge	Moteurs de base de données pris en charge	Classes d'instance prises en charge
		• Asie-Pacifique (Sydney) • Asie-Pacifique (Tokyo) • Canada (Centre) • Europe (Francfort) • Europe (Irlande) • Europe (Londres) • Europe (Paris) • Europe (Stockholm) • Amérique du Sud (São Paulo)		

Tarification et conservation des données pour Performance Insights

Par défaut, Performance Insights qui comprend 7 jours d'historique des données de performance et 1 million de demandes API par mois. Vous pouvez également acheter des périodes de conservation plus longues. Pour des informations complètes sur les prix, consultez la section Performance Insights Pricing (Tarification de Performance Insights).

Dans la console RDS, vous pouvez choisir l'une des périodes de conservation suivantes pour vos données Performance Insights :

- Par défaut (7 jours)
- ***n*** mois, où ***n*** est un nombre allant de 1 à 24



Pour savoir comment définir une période de conservation à l'aide de AWS CLI, consultez Activation ou désactivation de l'Analyse des performances pour Aurora.

> **Note**
>
> L'arrêt d'un cluster de bases de données alors que Performance Insights est activé n'a aucune incidence sur la conservation des données. En cas d'arrêt d'un cluster de bases de données, Performance Insights ne collecte aucune donnée.

Activation ou désactivation de l'Analyse des performances pour Aurora

> **Important**
>
> AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l'expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L'API Performance Insights continuera d'exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.
>
> Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.
>
> Si vous n'effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode Standard de Database Insights, vous risquez de perdre l'accès à l'historique des données de performance au-delà de 7 jours et de ne pas être en mesure d'utiliser les plans d'exécution et les fonctionnalités d'analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.
>
> Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance à grande échelle. Pour plus d'informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

Vous pouvez activer Performance Insights pour votre cluster de bases de données lors de sa création. Si nécessaire, vous pouvez le désactiver ultérieurement en modifiant le cluster de bases de données depuis la console. L'activation ou la désactivation de Performance Insights ne provoquent pas de durée d'indisponibilité, de redémarrage ni de basculement.

> **Note**
>
> Le schéma de performance est un outil de performance facultatif utilisé par Aurora MySQL. Si vous activez ou désactivez le schéma de performance, un redémarrage est requis. Toutefois, si vous activez ou désactivez Performance Insights, aucun redémarrage n'est requis. Pour de plus amples informations, veuillez consulter Présentation du schéma de performance pour Performance Insights sur Aurora MySQL.

Si vous utilisez Performance Insights avec des bases de données globales Aurora, activez Performance Insights sur chaque base de données dans chaque Région AWS. Pour en savoir plus, consultez Surveillance d'une base de données Amazon Aurora globale avec Amazon RDS Performance Insights.

L'agent Performance Insights consomme une quantité limitée d'UC et de mémoire sur l'hôte de base de données. Lorsque la charge de base de données est élevée, l'agent limite l'impact sur les performances en collectant des données moins fréquemment.

Console

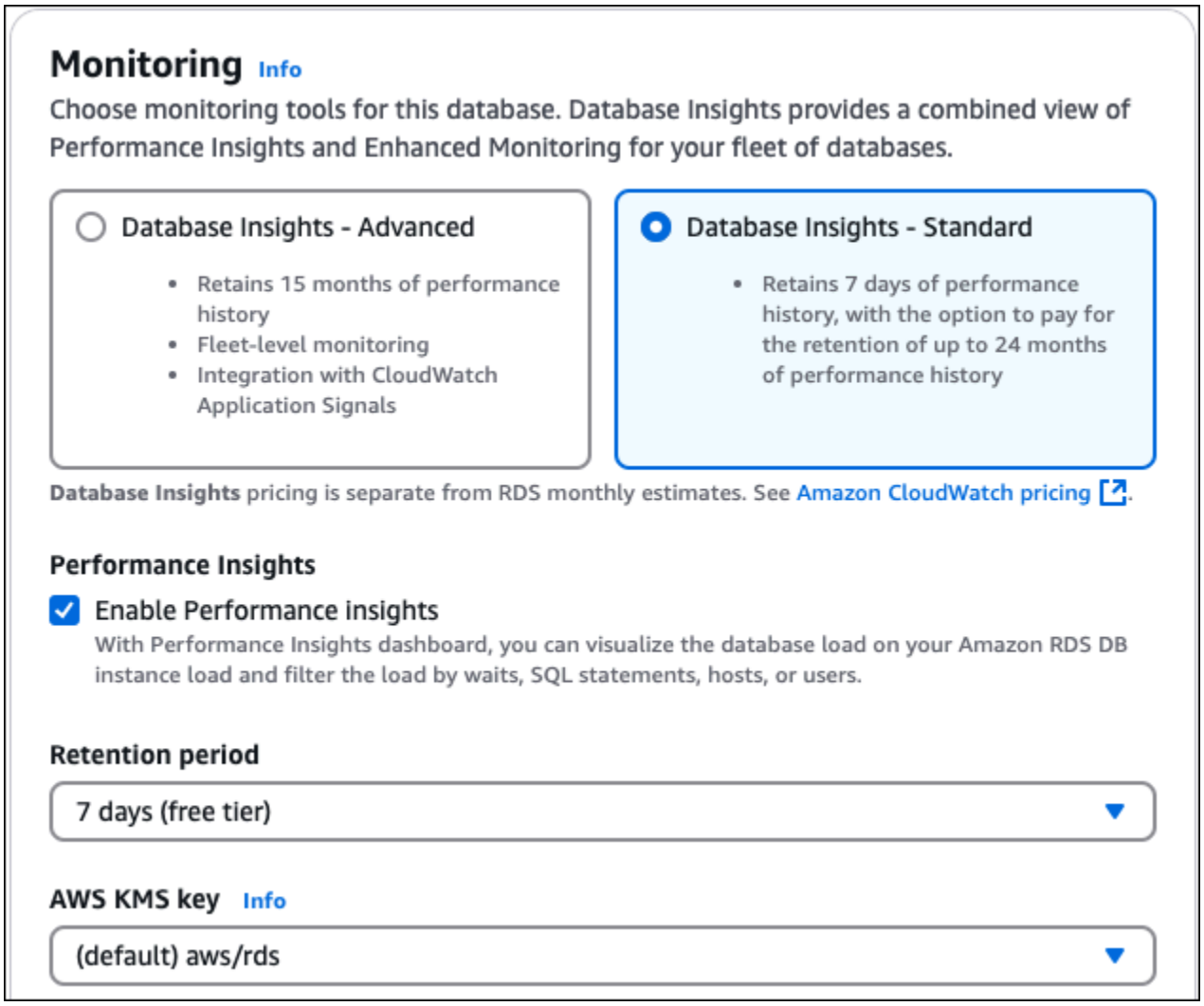
Dans la console, vous pouvez activer ou désactiver la fonctionnalité Analyse des performances lorsque vous créez un cluster de bases de données. L'activation de Performance Insights vous permet de gérer les paramètres et les options de Performance Insights pour votre cluster de bases de données. Les paramètres au niveau du cluster s'appliquent à toutes les instances de base de données du cluster.

Activation ou désactivation de Performance Insights lors de la création d'un cluster de bases de données

Après avoir créé un cluster de bases de données, Amazon RDS active Performance Insights par défaut. Pour désactiver Performance Insights pour votre cluster de bases de données, choisissez l'option Database Insights – Standard et désélectionnez l'option Activer Performance Insights.

Pour créer un cluster de bases de données, suivez les instructions pour votre moteur de base de données dans Création d'un cluster de bases de données Amazon Aurora.

L’image suivante représente la section Performance Insights.



Vous disposez des options suivantes lorsque vous choisissez Activer Performance Insights :

- Conservation (pour le mode Standard de Database Insights uniquement) : durée de conservation des données de Performance Insights. Le paramètre de conservation est Par défaut (7 jours). Pour conserver vos données de performance plus longtemps, indiquez 1 à 24 mois. Pour obtenir plus d’informations sur les périodes de conservation, consultez Tarification et conservation des données pour Performance Insights.
- AWS KMS key— Spécifiez votre AWS KMS key. Performance Insights chiffre toutes les données potentiellement sensibles à l’aide votre clé KMS. Les données sont chiffrées en transit et au repos. Pour plus d’informations, consultez Modification d’une politique AWS KMS pour Performance Insights.

Activation ou désactivation de Performance Insights lors de la modification d'un cluster de bases de données

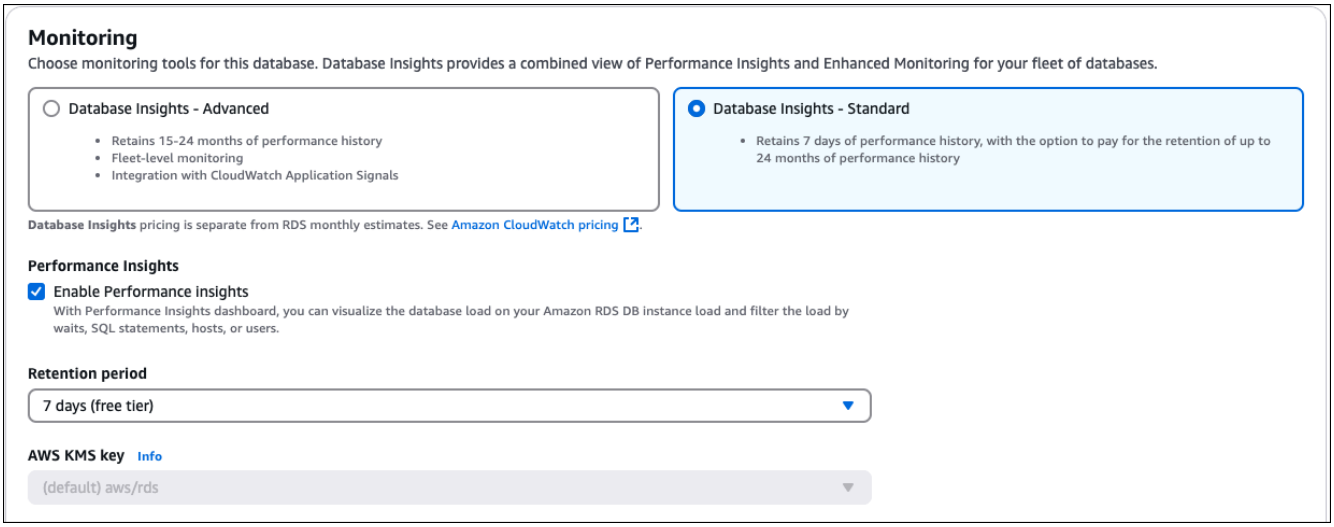
Dans la console, vous pouvez modifier un cluster de bases de données pour gérer Performance Insights.

Pour activer ou désactiver Performance Insights pour un cluster de bases de données en utilisant la console

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Choisissez Databases (Bases de données).
3. Choisissez un cluster de bases de données, et choisissez Modifier.
4. Pour activer Performance Insights, sélectionnez Activer Performance Insights. Pour désactiver Performance Insights pour votre cluster de bases de données, choisissez l'option Database Insights – Standard et désélectionnez l'option Activer Performance Insights.

   Vous disposez des options suivantes lorsque vous choisissez Activer Performance Insights :

   - Conservation (pour le mode Standard de Database Insights uniquement) : durée de conservation des données de Performance Insights. Le paramètre de conservation est Par défaut (7 jours). Pour conserver vos données de performance plus longtemps, indiquez 1 à 24 mois. Pour obtenir plus d'informations sur les périodes de conservation, consultez Tarification et conservation des données pour Performance Insights.
   - AWS KMS key – Spécifiez votre clé KMS. Performance Insights chiffre toutes les données potentiellement sensibles à l'aide votre clé KMS. Les données sont chiffrées en transit et au repos. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.



5. Choisissez Continuer.
6. Pour Scheduling of Modifications (Planification des modifications), choisissez Apply immediately (Appliquer immédiatement). Si vous choisissez Apply during the next scheduled maintenance window (Appliquer lors de la prochaine fenêtre de maintenance planifiée), votre instance ignore ce paramètre et active immédiatement Performance Insights.
7. Choisissez Modify instance (Modifier l'instance).

AWS CLI

Lorsque vous utilisez la create-db-instance AWS CLI commande, activez Performance Insights en spécifiant `--enable-performance-insights` et en `--database-insights-mode` réglant sur `advanced` ou`standard`. Pour désactiver Performance Insights, spécifiez `--no-enable-performance-insights` et définissez `database-insights-mode` sur `standard`.

Vous pouvez également spécifier ces valeurs à l'aide des AWS CLI commandes suivantes :

- create-db-cluster
- modify-db-cluster
- create-db-instance-read-réplique
- modify-db-instance
- restore-db-instance-from-s3

Pour gérer Performance Insights pour un cluster de bases de données à l'aide du AWS CLI

- Appelez la modify-db-cluster AWS CLI commande et entrez les valeurs suivantes :
  - `--db-cluster-identifier` : le nom de l'instance de base de données dans votre cluster de bases de données.
  - `--enable-performance-insights` pour l'activer ou `--no-enable-performance-insights` pour le désactiver
  - `database-insights-mode` : le mode de Database Insights pour le cluster de bases de données. Pour désactiver Performance Insights, définissez cette valeur sur `standard`.

  L'exemple suivant active Performance Insights pour `sample-db-cluster`.

  Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights
```

  Pour Windows :

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights
```

Lorsque vous activez Performance Insights dans l'interface CLI, vous pouvez éventuellement spécifier le nombre de jours pour conserver les données de Performance Insights avec l'option `--performance-insights-retention-period`. Vous pouvez spécifier 7 *`month`* * 31 (où *`month`* est un nombre compris entre 1 et 23) ou731. Par exemple, si vous souhaitez conserver vos données de performance pendant 3 mois, indiquez 93, soit 3 * 31. La durée par défaut est de 7 jours. Pour obtenir plus d'informations sur les périodes de conservation, consultez Tarification et conservation des données pour Performance Insights.

L'exemple suivant active Performance Insights pour `sample-db-cluster` et spécifie que les données de Performance Insights sont conservées pendant 93 jours (3 mois).

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
 --database-insights-mode standard \
    --db-cluster-identifier sample-db-instance \
    --enable-performance-insights \
    --performance-insights-retention-period 93
```

Pour Windows :

```
aws rds modify-db-cluster ^
 --database-insights-mode standard ^
    --db-cluster-identifier sample-db-instance ^
    --enable-performance-insights ^
    --performance-insights-retention-period 93
```

Si vous spécifiez une période de conservation telle que 94 jours, qui n'est pas une valeur valide, RDS émet une erreur.

```
An error occurred (InvalidParameterValue) when calling the CreateDBInstance
 operation:
Invalid Performance Insights retention period. Valid values are: [7, 31, 62, 93,
 124, 155, 186, 217,
248, 279, 310, 341, 372, 403, 434, 465, 496, 527, 558, 589, 620, 651, 682, 713, 731]
```

> **Note**
>
> Vous ne pouvez activer Performance Insights que pour une instance d'un cluster de bases de données où Performance Insights n'est pas géré au niveau du cluster.

RDS API

Lorsque vous créez une nouvelle instance de base de données dans votre cluster de bases de données à l'aide de l'DBInstanceopération Create operation Amazon RDS API, activez Performance Insights en définissant `EnablePerformanceInsights` sur. `True` Pour désactiver Performance Insights, définissez `EnablePerformanceInsights` sur `False` et `DatabaseInsightsMode` sur `standard`.

Vous pouvez également spécifier la valeur `EnablePerformanceInsights` à l'aide des opérations d'API suivantes :

- CréerDBCluster
- ModifyDBCluster
- ModifyDBInstance
- CréerDBInstanceReadReplica
- Restaurer à DBInstance partir de S3

Lorsque vous activez Performance Insights, vous pouvez facultativement spécifier la durée, en jours, de conservation des données de Performance Insights avec le paramètre `PerformanceInsightsRetentionPeriod`. Vous pouvez spécifier 7 *`month`* * 31 (où *`month`* est un nombre compris entre 1 et 23) ou`731`. Par exemple, si vous souhaitez conserver vos données de performance pendant 3 mois, indiquez `93`, soit 3 * 31. La durée par défaut est de 7 jours. Pour obtenir plus d'informations sur les périodes de conservation, consultez Tarification et conservation des données pour Performance Insights.

Présentation du schéma de performance pour Performance Insights sur Aurora MySQL

Le schéma de performance est une fonctionnalité facultative pour la surveillance des performances d'exécution d'Aurora MySQL à un faible niveau de détails. Le schéma de performance est conçu pour avoir un impact minimal sur les performances de base de données. Performance Insights est une fonctionnalité distincte que vous pouvez utiliser avec ou sans le schéma de performance.

Rubriques

- Présentation du schéma de performance
- Performance Insights et le schéma de performance
- Gestion automatique du schéma de performance par Performance Insights
- Effet d'un redémarrage sur le schéma de performance
- Déterminer si Performance Insights gère le schéma de performance
- Activation du schéma de performance sur Aurora MySQL

Présentation du schéma de performance

Le schéma de performance surveille les événements dans les bases de données Aurora MySQL. Un événement est une action du serveur de base de données qui consomme du temps et qui a été instrumentée de manière à ce que des informations temporelles puissent être collectées. Voici quelques exemples d'événements :

- Appels de fonction
- Attend le système d'exploitation
- Étapes de l'exécution SQL
- Groupes d'instructions SQL

Le moteur de stockage `PERFORMANCE_SCHEMA` est un mécanisme de mise en œuvre de la fonctionnalité de schéma de performance. Ce moteur collecte les données d'événement à l'aide d'une instrumentation dans le code source de la base de données. Le moteur stocke les événements dans des tables à mémoire uniquement dans la base de données `performance_schema`. Vous pouvez interroger `performance_schema` tout comme vous pouvez interroger d'autres tables. Pour plus d'informations, consultez MySQL Performance Schema (Schéma de performance MySQL) dans le Manuel de référence de MySQL.

Performance Insights et le schéma de performance

Performance Insights et Performance Schema sont des fonctionnalités distinctes, mais elles sont liées. Le comportement de Performance Insights pour Aurora MySQL dépend de l'activation ou non du schéma de performance, et si oui, si Performance Insights gère automatiquement le schéma de performance. Le tableau suivant décrit le comportement.

Schéma de performance activé	Mode de gestion de Performance Insights	Comportement de Performance Insights
Oui	Automatique	• Collecte des informations de surveillance détaillées et de bas niveau • Collecte des métriques de la session active toutes les secondes

Schéma de performance activé	Mode de gestion de Performance Insights	Comportement de Performance Insights
		• Affiche la charge de la base de données classée par événements d’attente détaillés, que vous pouvez utiliser pour identifier les goulots d’étranglement
Oui	Manuelle	• Collecte les événements d’attente et les métriques par SQL • Collecte des métriques de la session active toutes les secondes • Les rapports sur les états des utilisateurs, tels que l’insertion et l’envoi, ne vous aident pas à identifier les goulots d’étranglement
Non	N/A	• Il ne collecte pas les événements d’attente, les métriques par SQL ou d’autres informations de surveillance détaillées et de bas niveau • Il collecte des métriques de la session active toutes les cinq secondes au lieu de toutes les secondes • Les rapports sur les états des utilisateurs, tels que l’insertion et l’envoi, ne vous aident pas à identifier les goulots d’étranglement

Gestion automatique du schéma de performance par Performance Insights

Le schéma de performance est également activé lorsque vous créez une instance de base de données Aurora MySQL avec Performance Insights activé. Le cas échéant, Performance Insights gère automatiquement vos paramètres de schéma de performance. Il s’agit de la configuration recommandée.

Lorsque Performance Insights gère automatiquement le schéma de performance, la source de `performance_schema` est `System default`.

> **Note**
>
> La gestion automatique du schéma de performance n'est pas prise en charge pour la classe d'instance t4g.medium.

Vous pouvez également gérer le schéma de performance manuellement. Si vous choisissez cette option, réglez les paramètres selon les valeurs du tableau suivant.

Nom du paramètre	Valeur de paramètre
`performance_schema`	1 (La colonne Source a la valeur `Modified`)
`performance-schema-consumer-events-waits-current`	`ON`
`performance-schema-instrument`	`wait/%=ON`
`performance_schema_consumer_global_instrumentation`	`1`
`performance_schema_consumer_thread_instrumentation`	`1`

Si vous modifiez la valeur du paramètre `performance_schema` manuellement et que vous souhaitez ensuite passer à la gestion automatique, consultez Activation du schéma de performance sur Aurora MySQL.

> **Important**
>
> Lorsque Performance Insights active le schéma de performance, il ne modifie pas les valeurs du groupe de paramètres. Toutefois, les valeurs sont modifiées sur les instances de base de données en cours d'exécution. La seule façon de voir les valeurs modifiées est d'exécuter la commande `SHOW GLOBAL VARIABLES`.

Effet d’un redémarrage sur le schéma de performance

Performance Insights et le schéma de performance diffèrent dans leurs exigences en matière de redémarrage des instances de base de données :

Schéma de performance

Pour activer ou désactiver cette fonction, vous devez redémarrer l’instance de base de données.

Performance Insights

Pour activer ou désactiver cette fonction, il n’est pas nécessaire de redémarrer l’instance de base de données.

Si le schéma de performance n’est pas activé et que vous activez Performance Insights sans redémarrer l’instance de base de données, le schéma de performance ne sera pas activé.

Déterminer si Performance Insights gère le schéma de performance

Pour savoir si Performance Insights gère actuellement le schéma de performance pour toutes les versions majeures du moteur prises en charge, consultez le tableau suivant.

Définition du paramètre performance_schema	Paramétrage de la colonne Source	Performance Insights gère le schéma de performance
`0`	`System default`	Oui
`0` ou `1`	`Modified`	Non

Dans la procédure suivantes, vous déterminez si Performance Insights gère automatiquement le schéma de performance.

Pour déterminer si Performance Insights gère automatiquement le schéma de performance

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Choisissez Groupes de paramètres.
3. Sélectionnez le nom du groupe de paramètres pour votre instance de base de données.

4. Entrez **`performance_schema`** dans la barre de recherche.
5. Vérifiez que Source est la valeur par défaut du système et que le champ Valeurs est défini sur 0. Si c'est le cas, Performance Insights gère automatiquement le schéma de performance.

   Dans cet exemple, Performance Insights ne gère pas automatiquement le schéma de performance.

Activation du schéma de performance sur Aurora MySQL

Supposons que Performance Insights soit activé pour votre instance de base de données mais qu'il ne gère pas actuellement le schéma de performance. Si vous voulez permettre à Performance Insights de gérer automatiquement le schéma de performance, suivez les étapes suivantes.

Pour configurer le schéma de performance pour une gestion automatique

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Groupes de paramètres.
3. Sélectionnez le nom du groupe de paramètres pour votre instance de base de données.
4. Choisissez Modifier.
5. Entrez **`performance_schema`** dans la barre de recherche.
6. Sélectionnez le paramètre `performance_schema`.
7. Choisissez Définir sur la valeur par défaut.
8. Confirmez en choisissant Réinitialiser les valeurs par défaut.
9. Choisissez Save Changes (Enregistrer les modifications).
10. Redémarrage de l'instance de base de données.

> **Important**
>
> Chaque fois que vous activez ou désactivez le schéma de performance, veillez à redémarrer l'instance de base de données.

Pour plus d'informations sur la modification des paramètres d'instance de base de données, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora. Pour plus d'informations sur le tableau de bord, consultez Analyse des métriques à l'aide du tableau de bord de Performance Insights. Pour plus d'informations sur le schéma de performance MySQL, consultez les sections MySQL Performance Schema (pour 8.0) et MySQL Performance Schema (pour 8.4) dans la documentation MySQL.

Configuration des politiques d'accès pour Performance Insights

Pour accéder à Performance Insights, un directeur doit disposer des autorisations appropriées auprès de Gestion des identités et des accès AWS (IAM).

> **Note**
>
> Pour utiliser Performance Insights avec une clé gérée par le client, accordez aux utilisateurs les `kms:GenerateDataKey` autorisations `kms:Decrypt` et autorisations associées à votre AWS AWS KMS clé.

Accédez à Performance Insights à l'aide des méthodes suivantes :

- Joindre la politique `AmazonRDSPerformanceInsightsReadOnly` gérée pour un accès en lecture seule
- Joignez la politique `AmazonRDSPerformanceInsightsFullAccess` gérée pour accéder à toutes les opérations de l'API Performance Insights
- Créez une politique IAM personnalisée avec des autorisations spécifiques
- Configuration AWS KMS des autorisations pour les données Performance Insights chiffrées
- Configurez un accès détaillé à l'aide d'autorisations au niveau des ressources
- Utilisez le contrôle d'accès basé sur des balises pour gérer les autorisations via des balises de ressources

Associer la politique AmazonRDSPerformanceInsightsReadOnly à un principal IAM

`AmazonRDSPerformanceInsightsReadOnly`est une politique AWS gérée qui donne accès à toutes les opérations en lecture seule de l'API Amazon RDS Performance Insights.

Si vous vous associez `AmazonRDSPerformanceInsightsReadOnly` à un ensemble d'autorisations ou à un rôle, vous devez également associer les CloudWatch autorisations suivantes :

- `GetMetricStatistics`
- `ListMetrics`
- `GetMetricData`

Avec ces autorisations, le destinataire peut utiliser Performance Insights avec d'autres fonctionnalités de console.

Pour plus d'informations sur CloudWatch les autorisations, consultez la référence CloudWatch des autorisations Amazon.

Pour plus d'informations sur `AmazonRDSPerformanceInsightsReadOnly`, consultez Politique AWS gérée : AmazonRDSPerformanceInsightsReadOnly.

Associer la politique AmazonRDSPerformanceInsightsFullAccess à un principal IAM

`AmazonRDSPerformanceInsightsFullAccess`est une politique AWS gérée qui donne accès à toutes les opérations de l'API Amazon RDS Performance Insights.

Si vous vous associez `AmazonRDSPerformanceInsightsFullAccess` à un ensemble d'autorisations ou à un rôle, vous devez également associer les CloudWatch autorisations suivantes :

- `GetMetricStatistics`
- `ListMetrics`
- `GetMetricData`

Avec ces autorisations, le destinataire peut utiliser Performance Insights avec d'autres fonctionnalités de console.

Pour plus d'informations sur CloudWatch les autorisations, consultez la référence CloudWatch des autorisations Amazon.

Pour plus d'informations sur `AmazonRDSPerformanceInsightsFullAccess`, consultez Politique gérée AWS: AmazonRDSPerformanceInsightsFullAccess.

Création d’une politique IAM personnalisée pour Performance Insights

Pour les utilisateurs qui ne bénéficient pas de la politique `AmazonRDSPerformanceInsightsReadOnly` ou `AmazonRDSPerformanceInsightsFullAccess`, vous pouvez accorder l’accès à l’analyse des performances en créant ou modifiant une politique IAM gérée par l’utilisateur. Quand vous attachez la politique à un jeu d’autorisations ou à un rôle IAM, le destinataire peut utiliser Performance Insights.

Pour créer une politique personnalisée

1. Ouvrez la console IAM à l’adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation, choisissez Politiques.
3. Choisissez Créer une politique.
4. Sur la page Créer une politique, choisissez l’option JSON.
5. Copiez et collez le texte fourni dans la section du document de politique JSON du Guide de référence des politiques AWS gérées pour notre politique AmazonRDSPerformanceInsightsReadOnly ou AmazonRDSPerformanceInsightsFullAccess.
6. Choisissez Examiner une politique.
7. Indiquez un nom pour la stratégie et éventuellement une description, puis choisissez Créer une stratégie.

Vous pouvez désormais attacher la politique à un jeu d’autorisations ou à un rôle. La procédure suivante suppose que vous disposez déjà d’un utilisateur à cette fin.

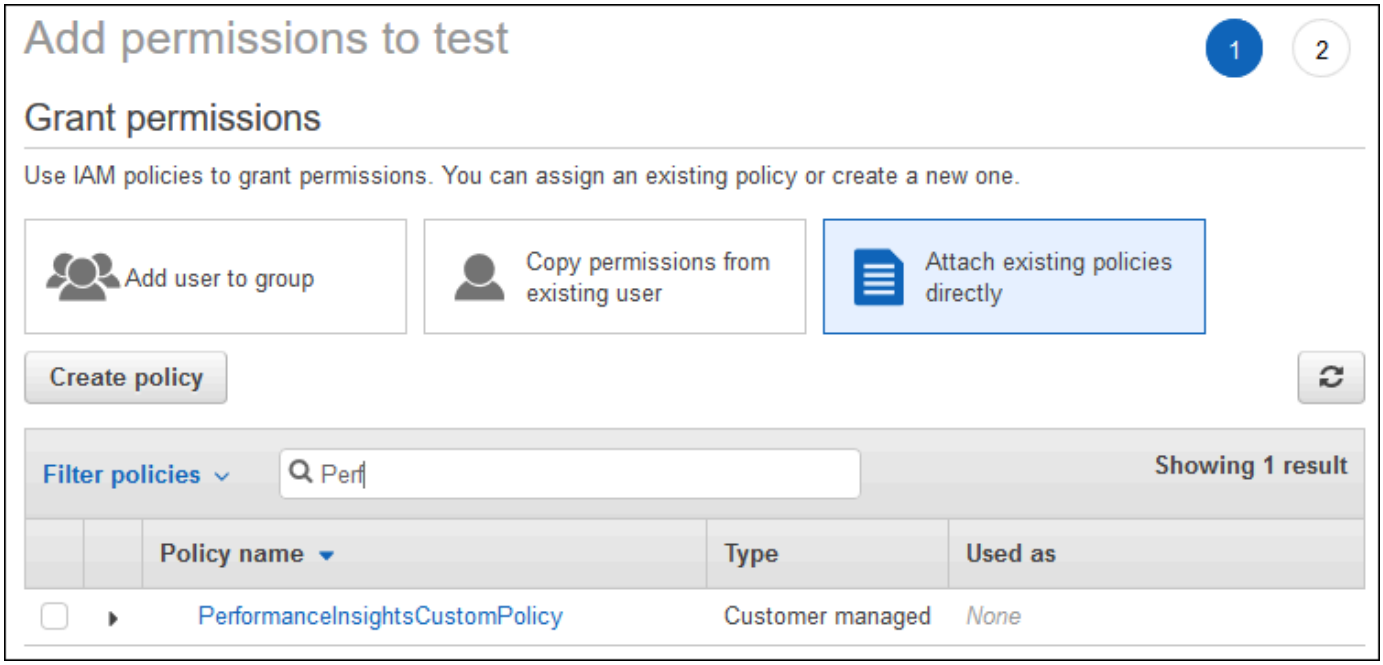
Pour attacher la politique à un utilisateur

1. Ouvrez la console IAM à l’adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation, sélectionnez Users.
3. Choisissez un utilisateur existant dans la liste.

> **Important**
>
> Pour utiliser Performance Insights, assurez-vous que vous avez accès à Amazon RDS en plus de l’accès à la politique personnalisée. Par exemple, la stratégie prédéfinie `AmazonRDSPerformanceInsightsReadOnly` fournit un accès en lecture seule

à Amazon RDS. Pour plus d'informations, consultez Gestion de l'accès à l'aide de politiques.

4. Sur la page Récapitulatif, choisissez Ajouter des autorisations.
5. Choisissez Attacher directement les stratégies existantes. Pour Recherche, tapez les premiers caractères du nom de votre politique, comme illustré dans l'image suivante.



6. Choisissez votre stratégie, puis sélectionnez Suivant : Vérification.
7. Choisissez Add permissions.

Modification d'une politique AWS KMS pour Performance Insights

Performance Insights utilise un AWS KMS key pour chiffrer les données sensibles. Lorsque vous activez Performance Insights via l'API ou la console, vous pouvez effectuer l'une ou l'autre des opérations suivantes :

- Choisissez la valeur par défautClé gérée par AWS.

  Amazon RDS utilise le Clé gérée par AWS pour votre nouvelle instance de base de données. Amazon RDS crée une Clé gérée par AWS pour votre Compte AWS. Vous Compte AWS avez un Amazon RDS différent Clé gérée par AWS pour chacun Région AWS d'entre eux.

- Choisissez une clé gérée par le client.

Si vous spécifiez une clé gérée par le client, les utilisateurs de votre compte qui appellent l'API Performance Insights ont besoin des autorisations `kms:Decrypt` et `kms:GenerateDataKey` sur la clé KMS. Vous pouvez configurer ces autorisations via des politiques IAM. Toutefois, nous vous recommandons de gérer ces autorisations via votre politique de clé KMS. Pour plus d'informations, consultez Politiques de clé dans AWS KMS dans le Guide du développeur AWS Key Management Service.

Example

L'exemple suivant montre comment ajouter des instructions à votre politique KMS. Ces instructions permettent d'accéder à Performance Insights. Selon la façon dont vous utilisez la clé KMS, vous pouvez modifier certaines restrictions. Avant d'ajouter des instructions à votre politique, supprimez tous les commentaires.

JSON

```
{
    "Version":"2012-10-17",
    "Id" : "your-policy",
    "Statement" : [
        {
            "Sid" : "AllowViewingRDSPerformanceInsights",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::444455556666:role/Role1"
                ]
                },
             "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
                ],
            "Resource": "*",
            "Condition" : {
            "StringEquals" : {
                "kms:ViaService" : "rds.us-east-1.amazonaws.com"
                },
            "ForAnyValue:StringEquals": {
                "kms:EncryptionContext:aws:pi:service": "rds",
                "kms:EncryptionContext:service": "pi",
```

```
                "kms:EncryptionContext:aws:rds:db-id": "db-
AAAAABBBBBCCCCDDDDDEEEEE"
                }
            }
        }
    ]
}
```

Comment Performance Insights utilise les clés gérées par le AWS KMS client

L'analyse des performances utilise les clés gérées par le client pour chiffrer les données sensibles. Lorsque vous activez l'analyse des performances, vous pouvez fournir une clé AWS KMS via l'API. Performance Insights crée AWS KMS des autorisations sur cette clé. Il utilise la clé et effectue les opérations nécessaires au traitement des données sensibles. Les données sensibles incluent des champs tels que l'utilisateur, la base de données, l'application et le texte de requête SQL. L'analyse des performances garantit que les données restent chiffrées à la fois au repos et en transit.

Comment fonctionne Performance Insights IAM AWS KMS

IAM accorde des autorisations à des personnes spécifiques APIs. Performance Insights possède le public suivant APIs, que vous pouvez restreindre à l'aide de politiques IAM :

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetadata`
- `GetResourceMetrics`
- `ListAvailableResourceDimensions`
- `ListAvailableResourceMetrics`

Vous pouvez utiliser les demandes d'API suivantes pour obtenir des données sensibles.

- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`
- `GetResourceMetrics`

Lorsque vous utilisez l'API pour obtenir des données sensibles, l'analyse des performances exploite les informations d'identification de l'appelant. Cette vérification garantit que l'accès aux données sensibles est limité aux personnes ayant accès à la clé KMS.

Lorsque vous les appelez APIs, vous avez besoin d'autorisations pour appeler l'API via la politique IAM et d'autorisations pour invoquer l'`kms:decrypt`action via la politique AWS KMS clé.

L'API `GetResourceMetrics` peut renvoyer des données sensibles et non sensibles. Les paramètres de demande déterminent si la réponse doit inclure des données sensibles. L'API renvoie des données sensibles lorsque la demande inclut une dimension sensible dans les paramètres de filtre ou de regroupement.

Pour plus d'informations sur les dimensions que vous pouvez utiliser avec l'`GetResourceMetrics`API, consultez DimensionGroup.

Example Exemples

L'exemple suivant demande les données sensibles pour le groupe `db.user` :

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
  "ServiceType": "RDS",
  "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
  "MetricQueries": [
    {
      "Metric": "db.load.avg",
      "GroupBy": {
        "Group": "db.user",
        "Limit": 2
      }
    }
  ],
```

```
  "StartTime": 1693872000,
  "EndTime": 1694044800,
  "PeriodInSeconds": 86400
}
```

Example

L'exemple suivant demande les données non sensibles pour la métrique `db.load.avg` :

```
POST / HTTP/1.1
Host: <Hostname>
Accept-Encoding: identity
X-Amz-Target: PerformanceInsightsv20180227.GetResourceMetrics
Content-Type: application/x-amz-json-1.1
User-Agent: <UserAgentString>
X-Amz-Date: <Date>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "MetricQueries": [
        {
            "Metric": "db.load.avg"
        }
    ],
    "StartTime": 1693872000,
    "EndTime": 1694044800,
    "PeriodInSeconds": 86400
}
```

Octroi d'accès affiné pour Performance Insights

Le contrôle d'accès affiné offre des moyens supplémentaires de contrôler l'accès à vos données sur Performance Insights. Ce contrôle d'accès permet d'autoriser ou de refuser l'accès à des dimensions individuelles pour les actions Performance Insights `GetResourceMetrics`, `DescribeDimensionKeys` et `GetDimensionKeyDetails`. Pour utiliser un accès affiné, spécifiez les dimensions dans la politique IAM à l'aide de clés de condition. L'évaluation de l'accès suit la

logique d'évaluation de la politique IAM. Pour plus d'informations, consultez Logique d'évaluation des politiques dans le Guide de l'utilisateur IAM. Si la déclaration de politique IAM ne spécifie aucune dimension, elle contrôle l'accès à toutes les dimensions pour l'action spécifiée. Pour obtenir la liste des dimensions disponibles, consultez DimensionGroup.

Pour connaître les dimensions auxquelles vos informations d'identification sont autorisées à accéder, utilisez le paramètre `AuthorizedActions` dans `ListAvailableResourceDimensions` et spécifiez l'action. Les valeurs autorisées pour `AuthorizedActions` sont les suivantes :

- `GetResourceMetrics`
- `DescribeDimensionKeys`
- `GetDimensionKeyDetails`

Par exemple, si vous spécifiez `GetResourceMetrics` pour le paramètre `AuthorizedActions`, `ListAvailableResourceDimensions` renvoie la liste des dimensions auxquelles l'action `GetResourceMetrics` est autorisée à accéder. Si vous spécifiez plusieurs actions dans le paramètre `AuthorizedActions`, il `ListAvailableResourceDimensions` renvoie une intersection de dimensions auxquelles ces actions sont autorisées à accéder.

Example

L'exemple suivant fournit l'accès aux dimensions pour les actions `GetResourceMetrics` et `DescribeDimensionKeys`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
```

```
        },
        {
            "Sid": "SingleAllow",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.id",
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }


    ]
}
```

Voici la réponse pour la dimension demandée :

```
 // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
```

```
        "Groups": [
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                  //  { "Identifier": "db.sql_tokenized.db_id" }, // not included
 because not allows in the IAM Policy
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            }

        ] }
    ]
}
```

L'exemple suivant indique une autorisation et deux refus d'accès pour les dimensions.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
          {
            "Sid": "AllowToDiscoverDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:ListAvailableResourceDimensions"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
          },

          {
            "Sid": "O01AllowAllWithoutSpecifyingDimensions",
            "Effect": "Allow",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
```

```
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ]
        },

        {
            "Sid": "O01DenyAppDimensionForAll",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics",
                "pi:DescribeDimensionKeys"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.application.name"
                    ]
                }
            }
        },

        {
            "Sid": "O01DenySQLForGetResourceMetrics",
            "Effect": "Deny",
            "Action": [
                "pi:GetResourceMetrics"
            ],
            "Resource": [
                "arn:aws:pi:us-east-1:123456789012:metrics/rds/db-
ABC1DEFGHIJKL2MNOPQRSTUV3W"
            ],
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "pi:Dimensions": [
                        "db.sql_tokenized.statement"
                    ]
                }
            }
        }
    ]
```

```
}
```

Voici les réponses pour les dimensions demandées :

```
   // ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["GetResourceMetrics"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

```
// ListAvailableResourceDimensions API
// Request
{
    "ServiceType": "RDS",
    "Identifier": "db-ABC1DEFGHIJKL2MNOPQRSTUV3W",
    "Metrics": [ "db.load" ],
    "AuthorizedActions": ["DescribeDimensionKeys"]
}

// Response
{
    "MetricDimensions": [ {
        "Metric": "db.load",
        "Groups": [
            {
                "Group": "db.application",
                "Dimensions": [
                  // removed from response because denied by the IAM Policy
                  //  { "Identifier": "db.application.name" }
                ]
            },
            {
                "Group": "db.sql_tokenized",
                "Dimensions": [
                    { "Identifier": "db.sql_tokenized.id" },
                    { "Identifier": "db.sql_tokenized.db_id" },

                  // allowed for DescribeDimensionKeys because our IAM Policy
                  // denies it only for GetResourceMetrics
                    { "Identifier": "db.sql_tokenized.statement" }
                ]
            },
            ...
        ] }
    ]
}
```

Utilisation du contrôle d'accès basé sur des balises pour Performance Insights

Vous pouvez contrôler l'accès aux métriques de Performance Insights à l'aide de balises héritées de l'instance de base de données parent. Pour contrôler l'accès aux opérations Performance Insights,

utilisez les politiques IAM. Ces politiques peuvent vérifier les balises de votre instance de base de données pour déterminer les autorisations.

Comment les tags fonctionnent avec Performance Insights

Performance Insights applique automatiquement les balises de votre instance de base de données pour autoriser les métriques Performance Insights. Lorsque vous ajoutez des balises à votre instance de base de données, vous pouvez immédiatement les utiliser pour contrôler l'accès aux données Performance Insights.

- Pour ajouter ou mettre à jour des balises pour les métriques Performance Insights, modifiez les balises de votre instance de base de données.
- Pour consulter les balises des métriques Performance Insights, faites appel `ListTagsForResource` à la ressource métrique Performance Insights. Il renverra les balises de l'instance de base de données associée à la métrique.

> **Note**
>
> Les `UntagResource` opérations `TagResource` et renvoient une erreur si vous essayez de les utiliser directement sur les métriques Performance Insights.

Création de politiques IAM basées sur des balises

Pour contrôler l'accès aux opérations Performance Insights, utilisez la clé de `aws:ResourceTag` condition dans vos politiques IAM. Ces politiques vérifient les balises de votre instance de base de données.

Example

Cette politique empêche l'accès aux métriques Performance Insights pour les bases de données de production. La politique interdit l'`pi:GetResourceMetrics`opération dans Performance Insights pour toute ressource de base de données étiquetée avec`env:prod`.

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Effect": "Deny",
           "Action": "pi:GetResourceMetrics",
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/env": "prod"
                }
            }
        }
    ]
}
```

Analyse des métriques à l'aide du tableau de bord de Performance Insights

Important

AWS a annoncé la end-of-life date de Performance Insights : le 30 juin 2026. Après cette date, Amazon RDS ne prendra plus en charge l'expérience de la console Performance Insights, les périodes de conservation flexibles (1 à 24 mois) et les tarifs associés. L'API Performance Insights continuera d'exister sans modification de prix. Les coûts de l'API Performance Insights apparaîtront sur votre AWS facture avec le coût de CloudWatch Database Insights.

Nous vous recommandons de mettre à niveau toutes les de base de données de clusters de bases de données utilisant le niveau payant de Performance Insights vers le mode avancé de Database Insights avant le 30 juin 2026. Pour en savoir plus sur la mise à niveau vers le mode avancé de Database Insights, consultez Activation du mode Avancé de Database Insights pour Amazon Aurora.

Si vous n'effectuez aucune action, les clusters de bases de données utilisant Performance Insights utiliseront par défaut le mode Standard de Database Insights. Avec le mode Standard de Database Insights, vous risquez de perdre l'accès à l'historique des données de performance au-delà de 7 jours et de ne pas être en mesure d'utiliser les plans d'exécution et les fonctionnalités d'analyse à la demande dans la console Amazon RDS. Après le 30 juin 2026, seul le mode avancé de Database Insights prendra en charge les plans d'exécution et les analyses à la demande.

Avec CloudWatch Database Insights, vous pouvez surveiller la charge de base de données de votre parc de bases de données et analyser et résoudre les problèmes de performance à grande échelle. Pour plus d'informations sur Database Insights, consultez Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights. Pour plus d'informations sur les tarifs, consultez Amazon CloudWatch Pricing.

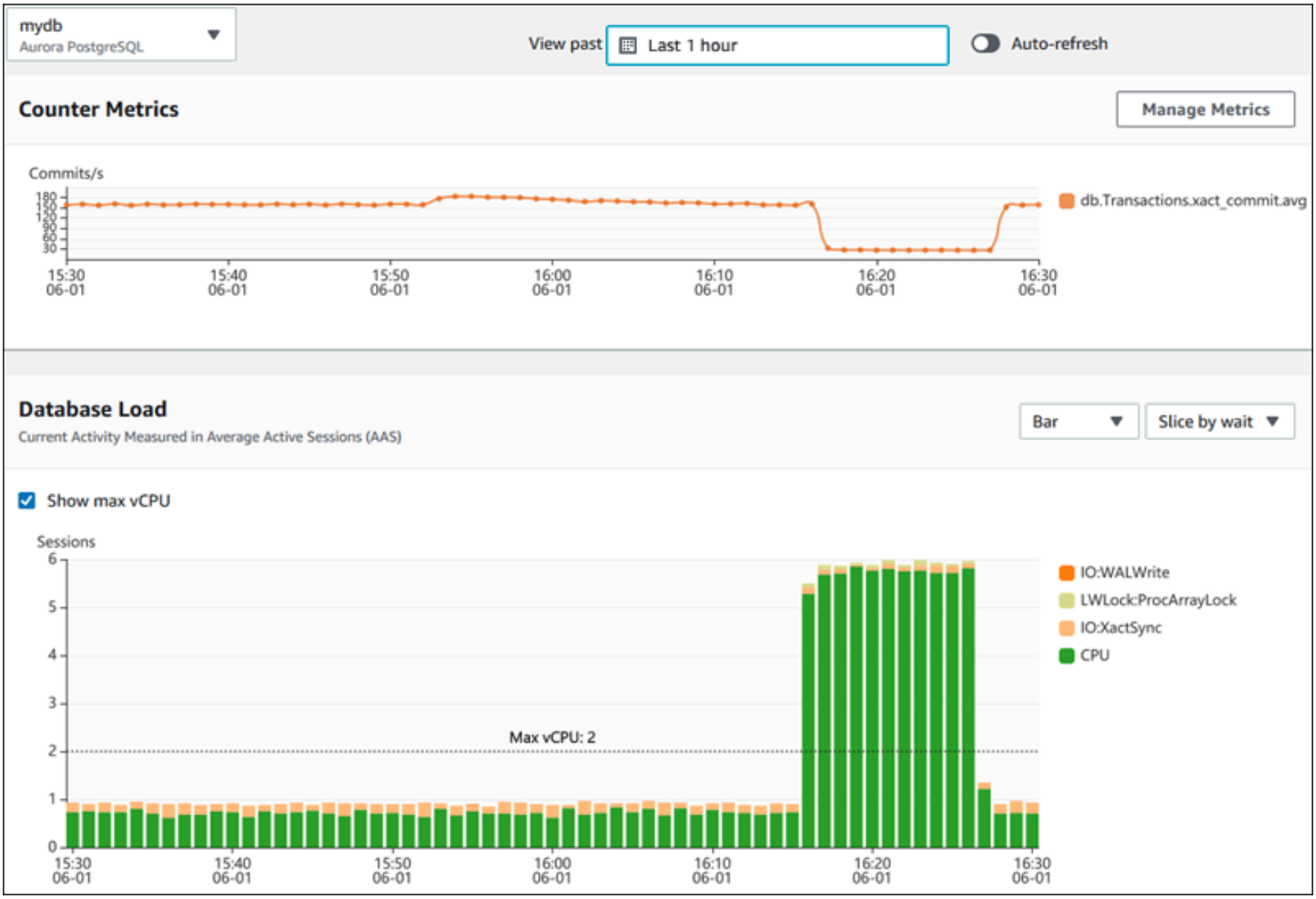
Le tableau de bord de Performance Insights contient des informations sur les performances des bases de données qui vous aideront à analyser et à résoudre les problèmes de performances. Sur la page de tableau de bord principale, vous pouvez afficher des informations concernant la charge de la base de données. Vous pouvez « trancher » la charge de base de données en différentes dimensions, telles que les événements d'attente ou SQL.

Tableau de bord Performance Insights

- Présentation du tableau de bord Performance Insights
- Accès au tableau de bord Performance Insights
- Analyse de la charge de base de données par événements d'attente
- Analyse des performances de base de données pour une période donnée
- Analyse des requêtes à l'aide de l'onglet Top SQL dans Performance Insights

Présentation du tableau de bord Performance Insights

Le tableau de bord est le moyen le plus simple d'interagir avec Performance Insights. L'exemple suivant présente le tableau de bord pour une instance de base de données PostgreSQL.



Rubriques

- Filtre de plage de temps
- Graphique Counter Metrics (Métriques de compteur)
- Graphique Database Load (Charge de la base de données)
- Tableau des dimensions principales

Filtre de plage de temps

Par défaut, le tableau de bord de Performance Insights affiche la charge de la base de données pour la dernière heure. Vous pouvez régler cette plage pour qu'elle soit aussi courte que 5 minutes ou aussi longue que 2 ans. Vous pouvez également sélectionner une plage relative personnalisée.

Vous pouvez sélectionner une plage absolue avec une date et une heure de début et de fin. L'exemple suivant montre la plage horaire commençant à minuit le 25/09/24 et se terminant à 23h59 le 28/09/24.

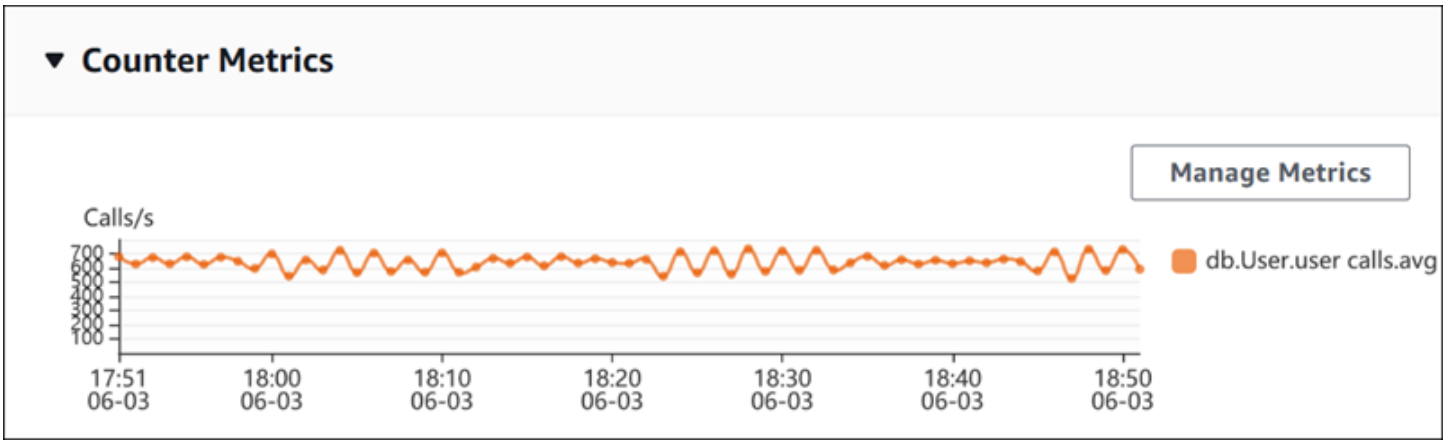
Par défaut, le fuseau horaire du tableau de bord Performance Insights est UTC (Coordinated Universal Time). Vous pouvez également choisir le fuseau horaire local.

Graphique Counter Metrics (Métriques de compteur)

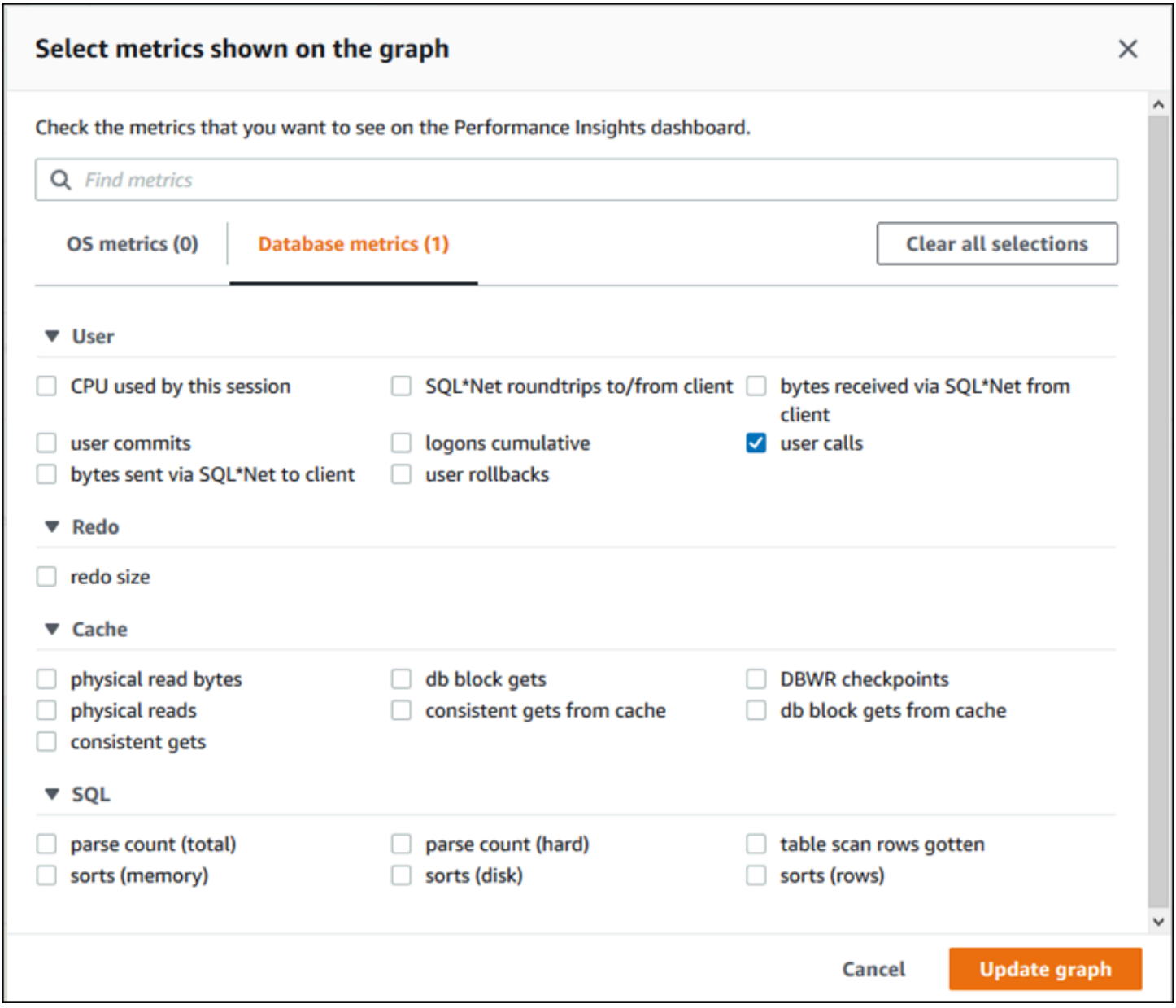
Grâce aux métriques de compteur, vous pouvez personnaliser le tableau de bord de Performance Insights de sorte à inclure jusqu'à 10 graphiques supplémentaires. Ces graphiques présentent une dizaine de métriques de performances de base de données et de système d'exploitation. Vous pouvez établir des corrélations entre ces informations et la charge de la base de données pour identifier et analyser les problèmes de performances.

Le graphique Counter Metrics (Métriques de compteur) affiche les données des compteurs de performances. Les métriques par défaut varient en fonction du moteur de base de données :

- Aurora MySQL– `db.SQL.Innodb_rows_read.avg`
- Aurora PostgreSQL – `db.Transactions.xact_commit.avg`



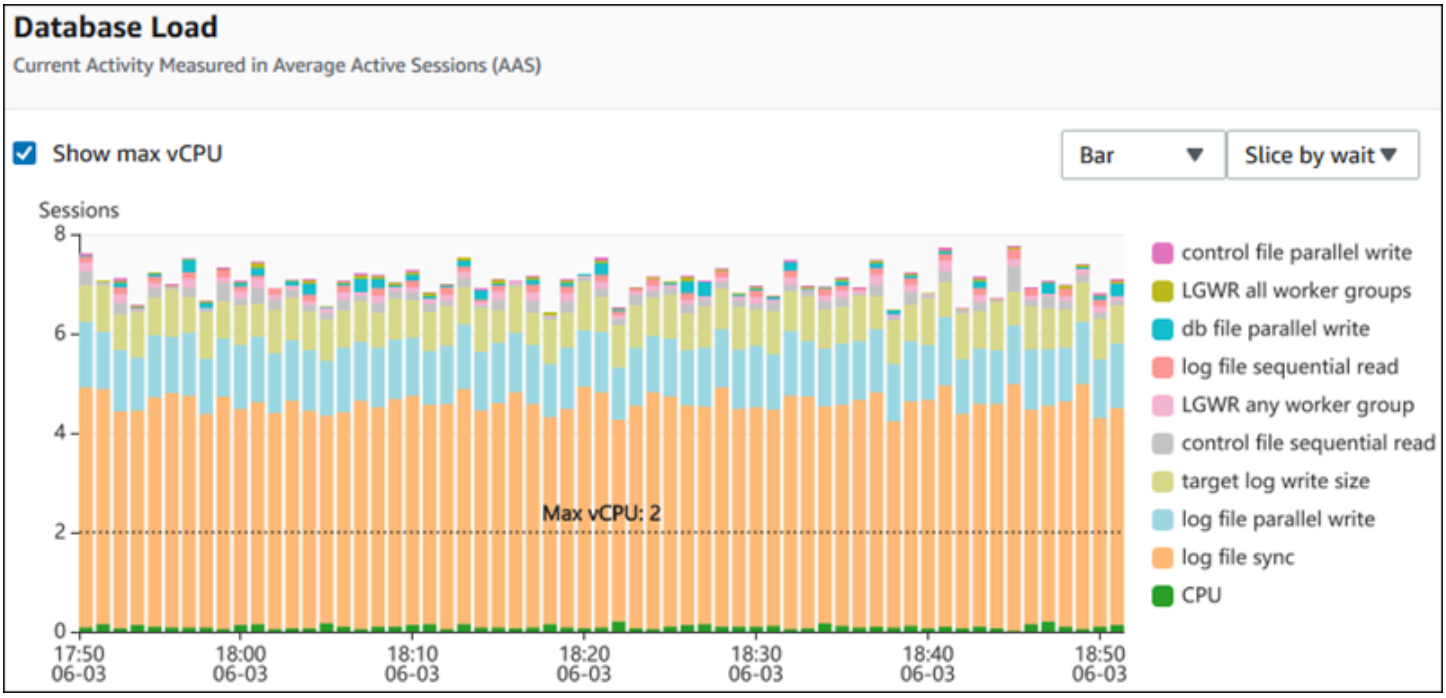
Pour changer de compteurs de performance, choisissez Manage Metrics (Gérer les métriques). Vous pouvez sélectionner plusieurs métriques de système d'exploitation ou métriques de base de données, comme illustré dans la capture d'écran suivante. Pour afficher les détails relatifs à une métrique, passez la souris sur le nom de la métrique.



Pour obtenir une description des métriques de compteur que vous pouvez ajouter pour chaque moteur de base de données, voir Métrique de compteur de Performance Insights.

Graphique Database Load (Charge de la base de données)

Le graphique Database Load (Charge de la base de données) montre l'activité de la base de données par rapport à la capacité de l'instance de base de données représentée par la ligne Max vCPU (vCPU max). Par défaut, le graphique en courbes empilées représente la charge de la base de données sous forme de sessions actives en moyenne par unité de temps. La charge de la base de données est découpée (groupée) par états d'attente.

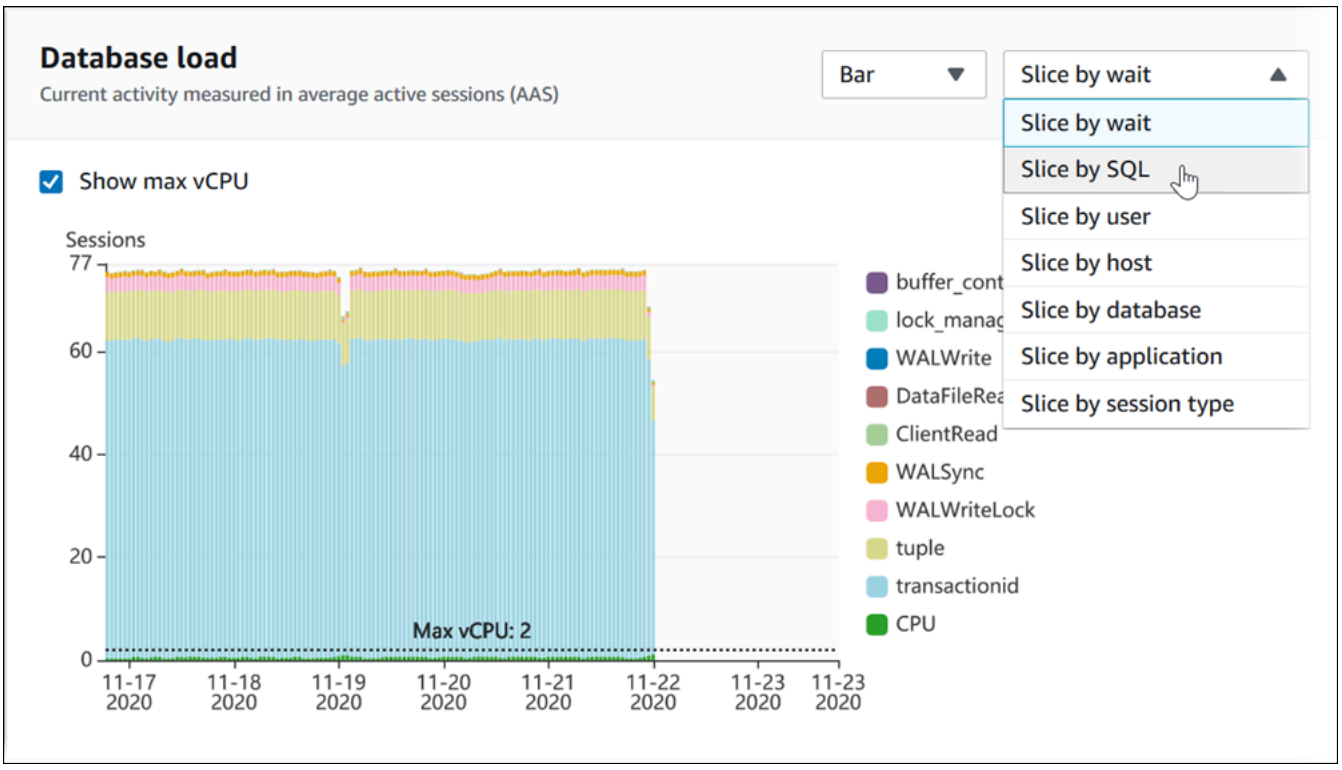


Charge de base de données tranchée par dimensions

Vous pouvez afficher la charge sous la forme de sessions actives regroupées par dimensions prises en charge. Le tableau suivant montre les dimensions prises en charge pour les différents moteurs.

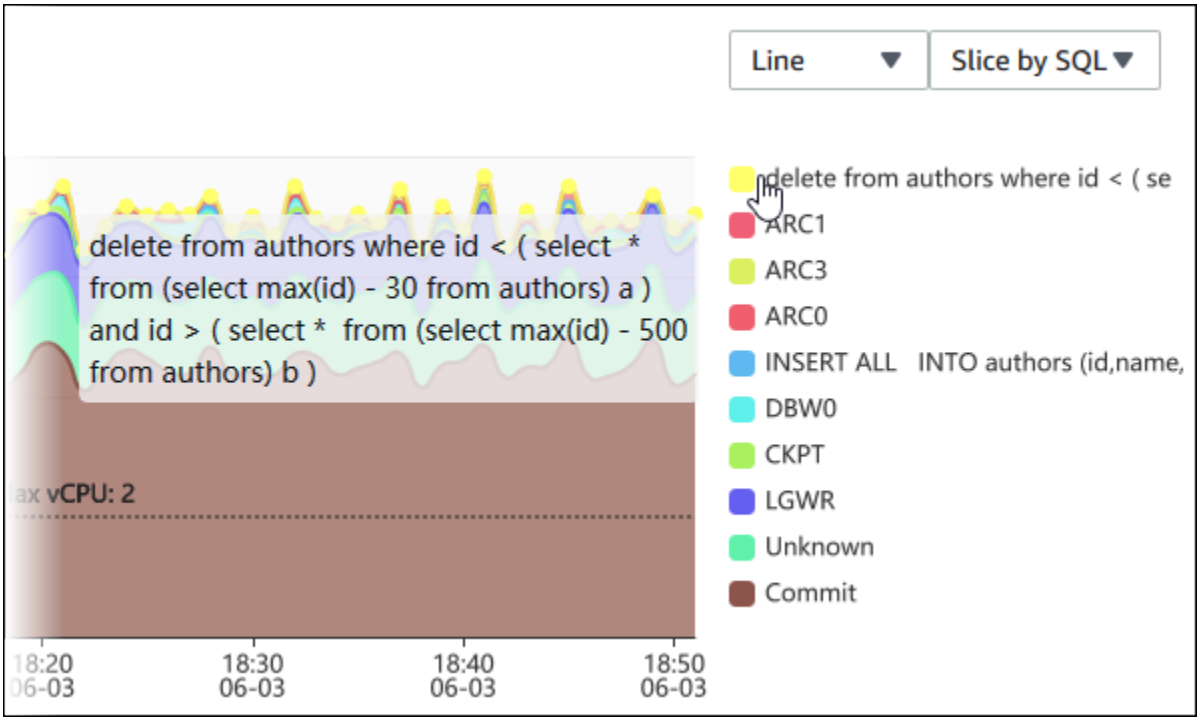
Dimension	Aurora PostgreSQL	Aurora MySQL
Host (Hôte)	Oui	Oui
SQL	Oui	Oui
Utilisateur	Oui	Oui
Éléments d’attente	Oui	Oui
Application	Oui	Non
Base de données	Oui	Oui
Type de session	Oui	Non

L’image suivante illustre les dimensions d’une instance de base de données PostgreSQL.



Détails de charge de base de données pour un élément de dimension

Pour afficher les détails d'un élément de charge de base de données dans une dimension, passez la souris sur le nom d'élément. L'image suivante illustre les détails d'une instruction SQL.



Pour afficher les détails d'un élément pour la période sélectionnée dans la légende, survolez cet élément.

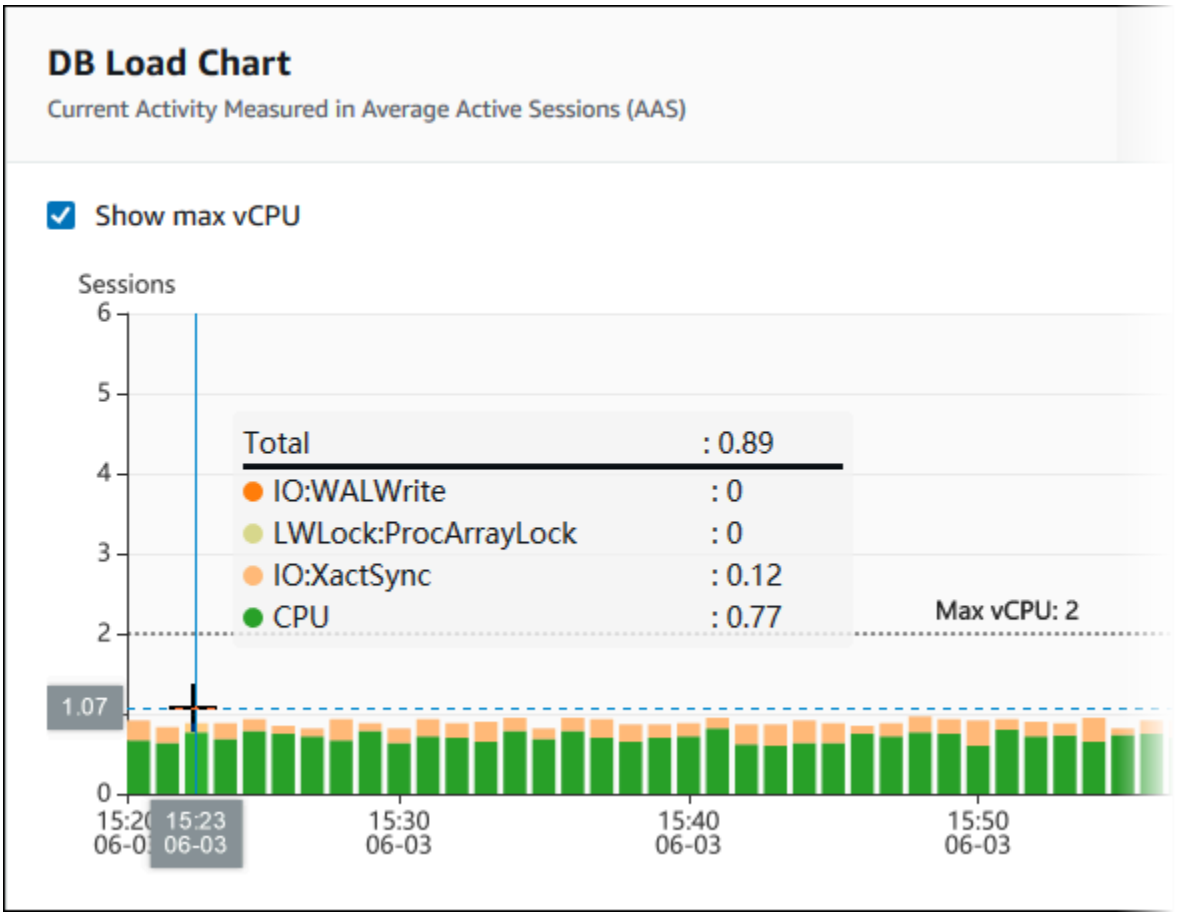
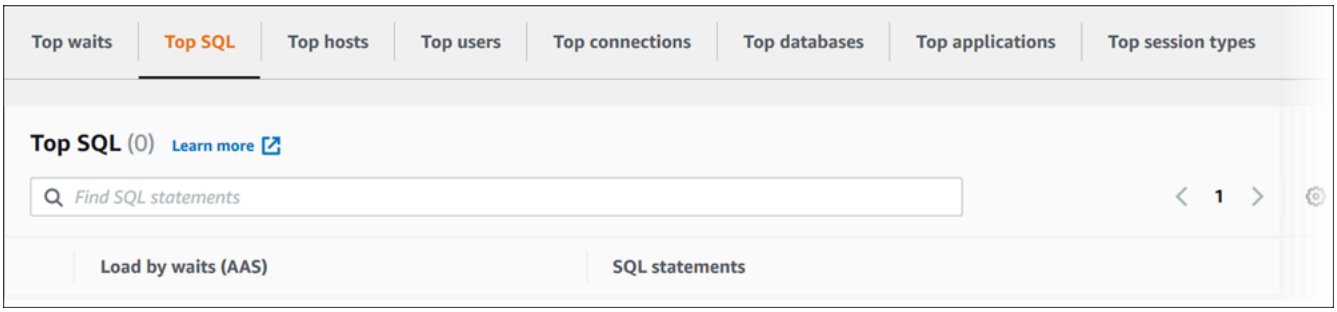


Tableau des dimensions principales

Le tableau des dimensions principales découpe la charge de la base de données selon différentes dimensions. Une dimension est une catégorie ou « tranche » qui représente l'une des différentes caractéristiques de la charge de la base de données. Si la dimension est SQL, Top SQL (Principaux éléments SQL) affiche les instructions SQL qui contribuent le plus à la charge de la base de données.



Choisissez l'un des onglets de dimension suivants.

Onglet	Description	Moteurs pris en charge
Top SQL (Principaux éléments SQL)	Instructions SQL en cours d'exécution	Tous
Principaux éléments d'attente	Événement pour lequel le backend de la base de données attend	Tous
Principaux hôtes	Nom d'hôte du client connecté	Tous
Principaux utilisateurs	Utilisateur connecté à la base de données	Tous
Principales applications	Nom de l'application connectée à la base de données	Aurora PostgreSQL uniquement
Principaux types de session	Type de la session en cours	Aurora PostgreSQL uniquement

Pour savoir comment analyser les requêtes à partir de l'onglet Top SQL (Principaux éléments SQL), consultez Présentation de l'onglet Top SQL (Principaux éléments SQL).

Accès au tableau de bord Performance Insights

Amazon RDS fournit une vue consolidée des métriques Performance Insights et CloudWatch dans le tableau de bord Performance Insights.

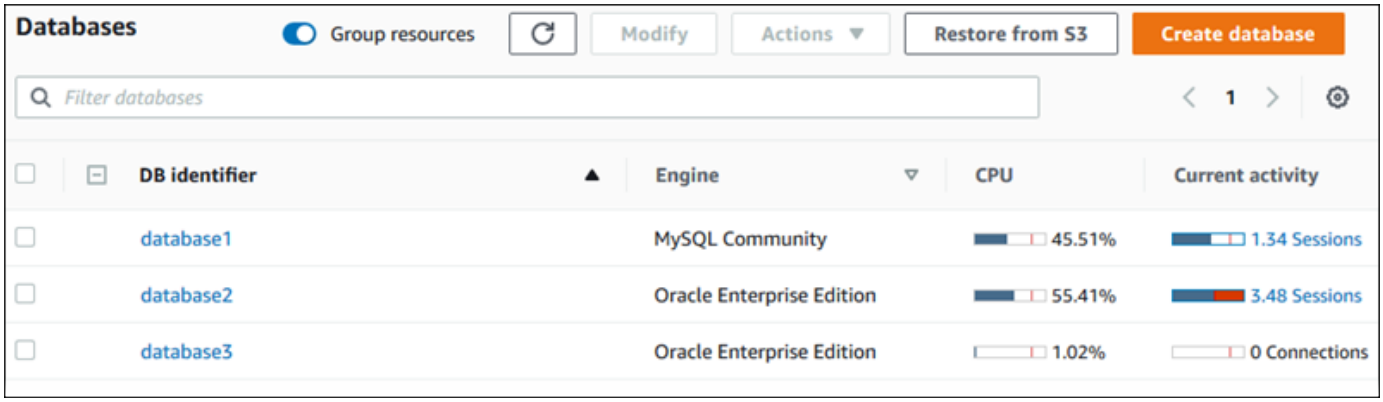
Pour accéder au tableau de bord de Performance Insights, procédez comme suit.

Pour afficher le tableau de bord Performance Insights dans la Console de gestion AWS

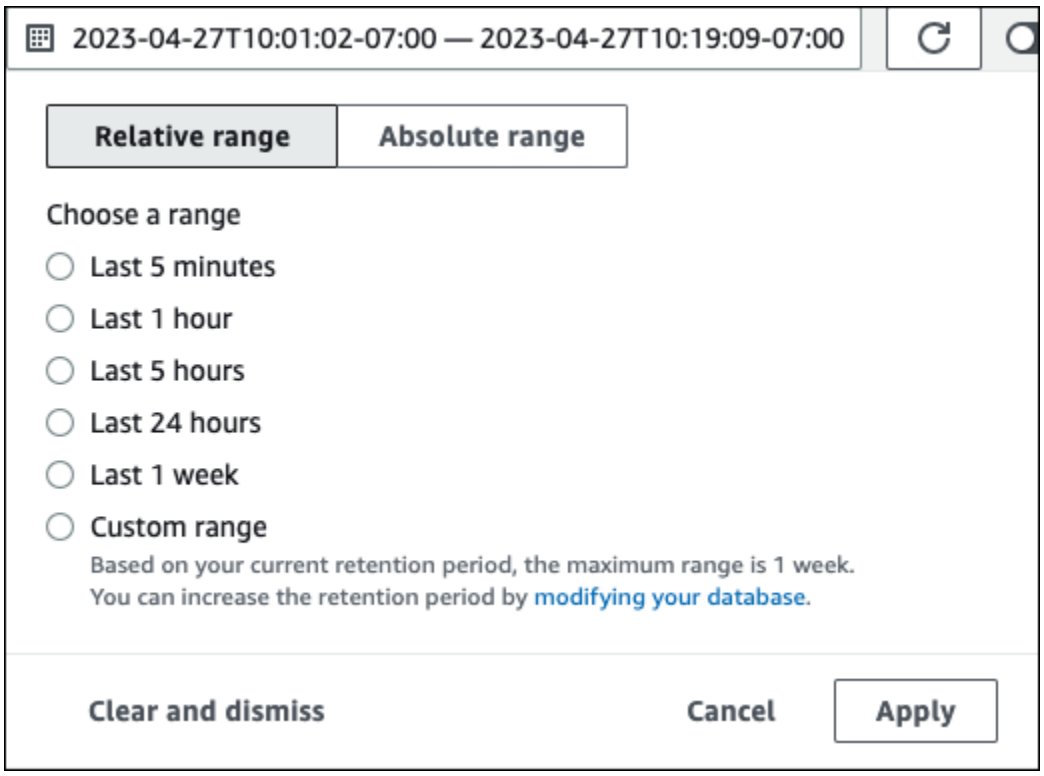
1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données.

   Pour les instances de base de données avec Performance Insights activé, vous pouvez également accéder au tableau de bord Performance Insights en choisissant l'élément Sessions

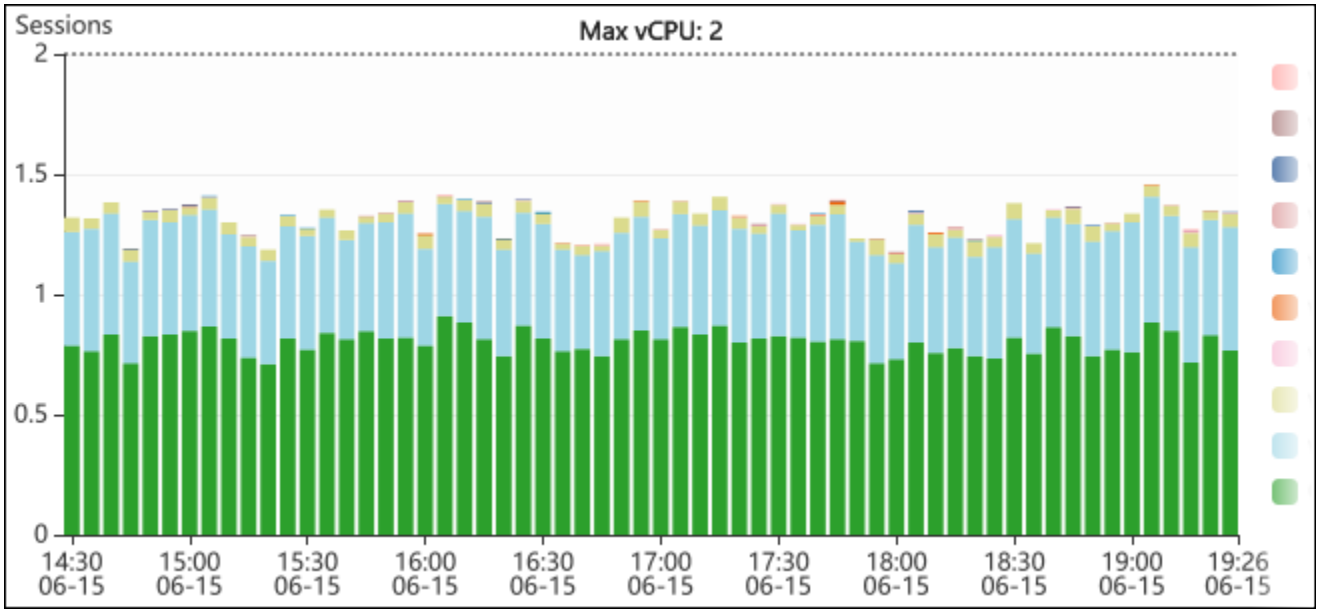
dans la liste des instances de base de données. Sous Activité actuelle, l'élément Sessions affiche la charge de base de données dans les sessions actives moyennes lors des cinq dernières minutes. La barre affiche visuellement le chargement. Votre instance de base de données est à l'arrêt lorsque la barre est vide. La barre se remplit de bleu à mesure que le chargement augmente. Une fois que le chargement dépasse le nombre d'UC virtuels (vUC) dans la classe d'instance de base de données, la barre vire au rouge, ce qui indique un engorgement potentiel.



4. (Facultatif) Choisissez la plage de dates ou de temps en haut à droite et spécifiez un autre intervalle de temps relatif ou absolu. Vous pouvez désormais spécifier une période et générer un rapport d'analyse des performances de base de données. Ce rapport fournit les informations et recommandations identifiées. Pour plus d'informations, consultez Création d'un rapport d'analyse des performances dans Performance Insights.

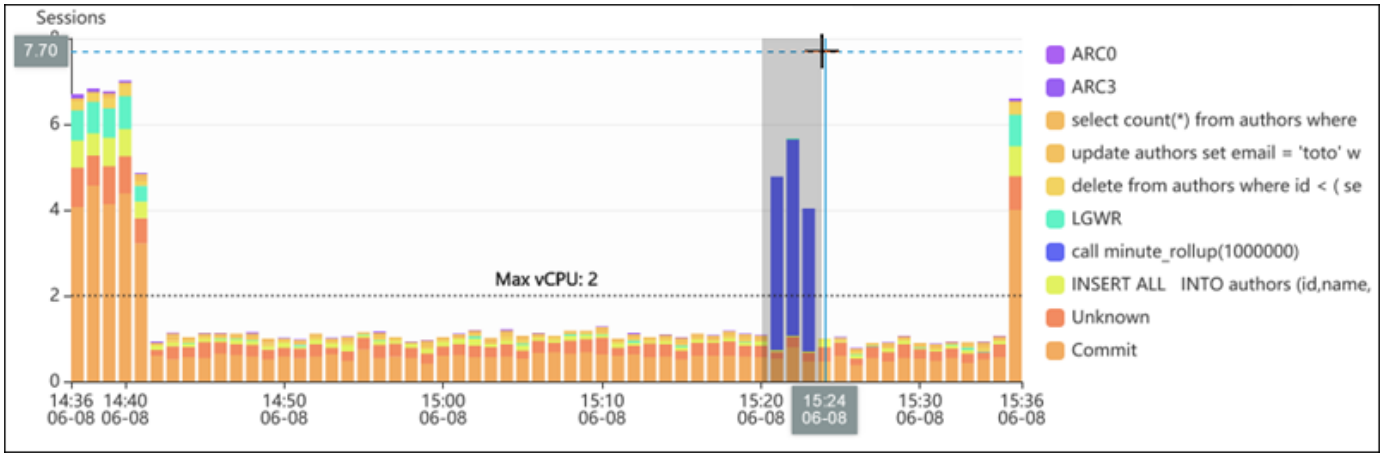


Dans la capture d'écran suivante, l'intervalle de charge de base de données est de 5 heures.

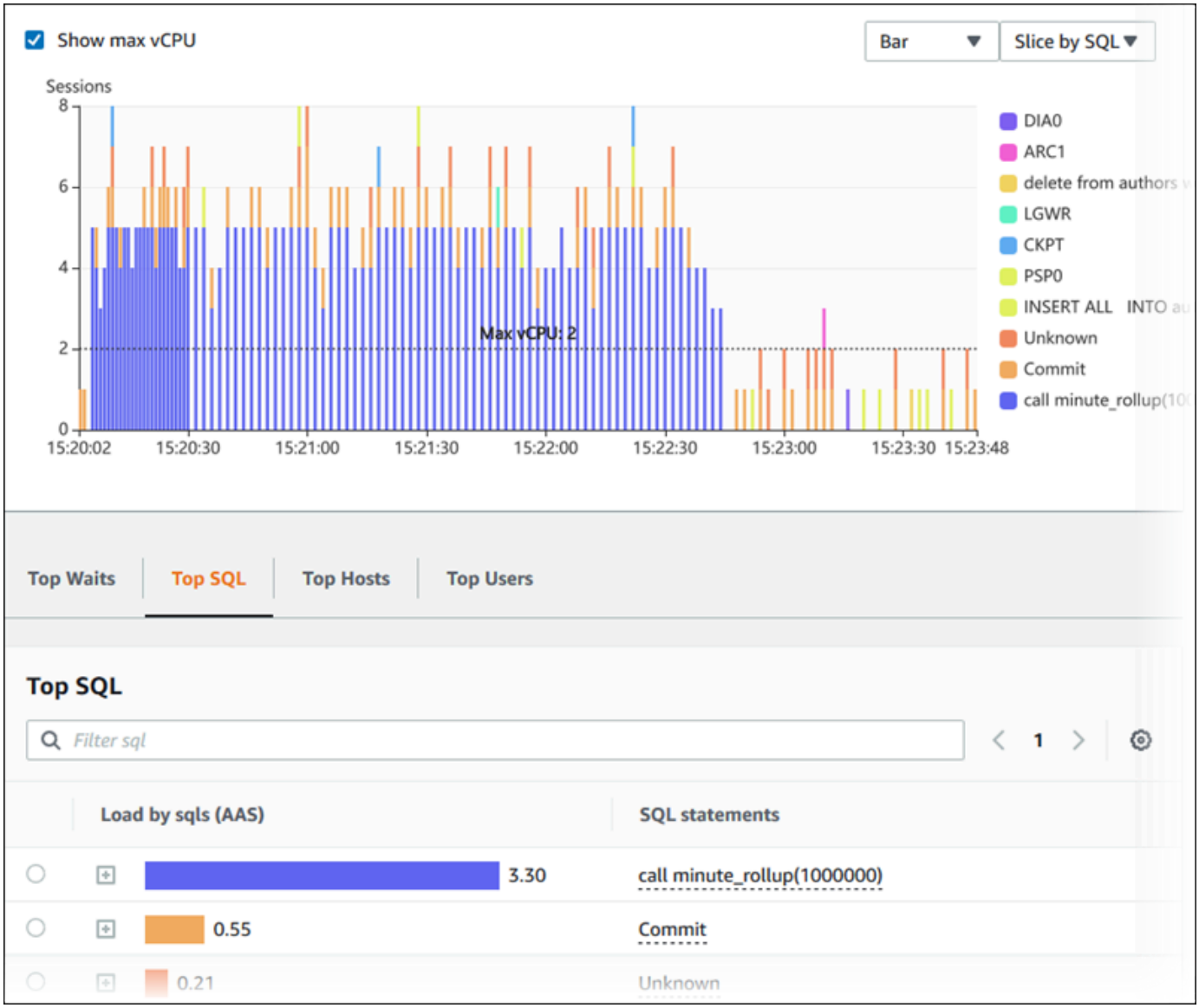


5. (Facultatif) Pour effectuer un zoom avant sur une partie du graphique de charge de la base de données, choisissez l'heure de début et faites glisser jusqu'à la fin de la période souhaitée.

   La zone sélectionnée est mise en évidence dans le tableau de charge de la base de données.



Lorsque vous relâchez la souris, le graphique de charge de la base de données fait un zoom avant sur la région AWS sélectionnée et la table Top dimensions (Principales dimensions) est recalculée.



6. (Facultatif) Pour actualiser automatiquement vos données, sélectionnez Actualisation automatique.

Le tableau de bord Performance Insights s'actualise automatiquement avec de nouvelles données. Le taux de rafraîchissement dépend de la quantité de données affichées :

- Pour 5 minutes, les données seront actualisées toutes les 10 secondes.
- Pour 1 heure, les données seront actualisées toutes les 5 minutes.
- Pour 5 heures, les données seront actualisées toutes les 5 minutes.
- Pour 24 heures, les données seront actualisées toutes les 30 minutes.

- Pour 1 semaine, les données seront actualisées tous les jours.
- Pour 1 mois, les données seront actualisées tous les jours.

Analyse de la charge de base de données par événements d’attente

Si le graphique Database load (Charge de la base de données) présente un goulot d’étranglement, vous pouvez déterminer la provenance de la charge. Pour ce faire, examinez le tableau des principaux éléments de charge en dessous du graphique Database load (Charge de la base de données). Choisissez un élément précis (une requête SQL ou un utilisateur par exemple) pour approfondir son analyse et afficher les détails le concernant.

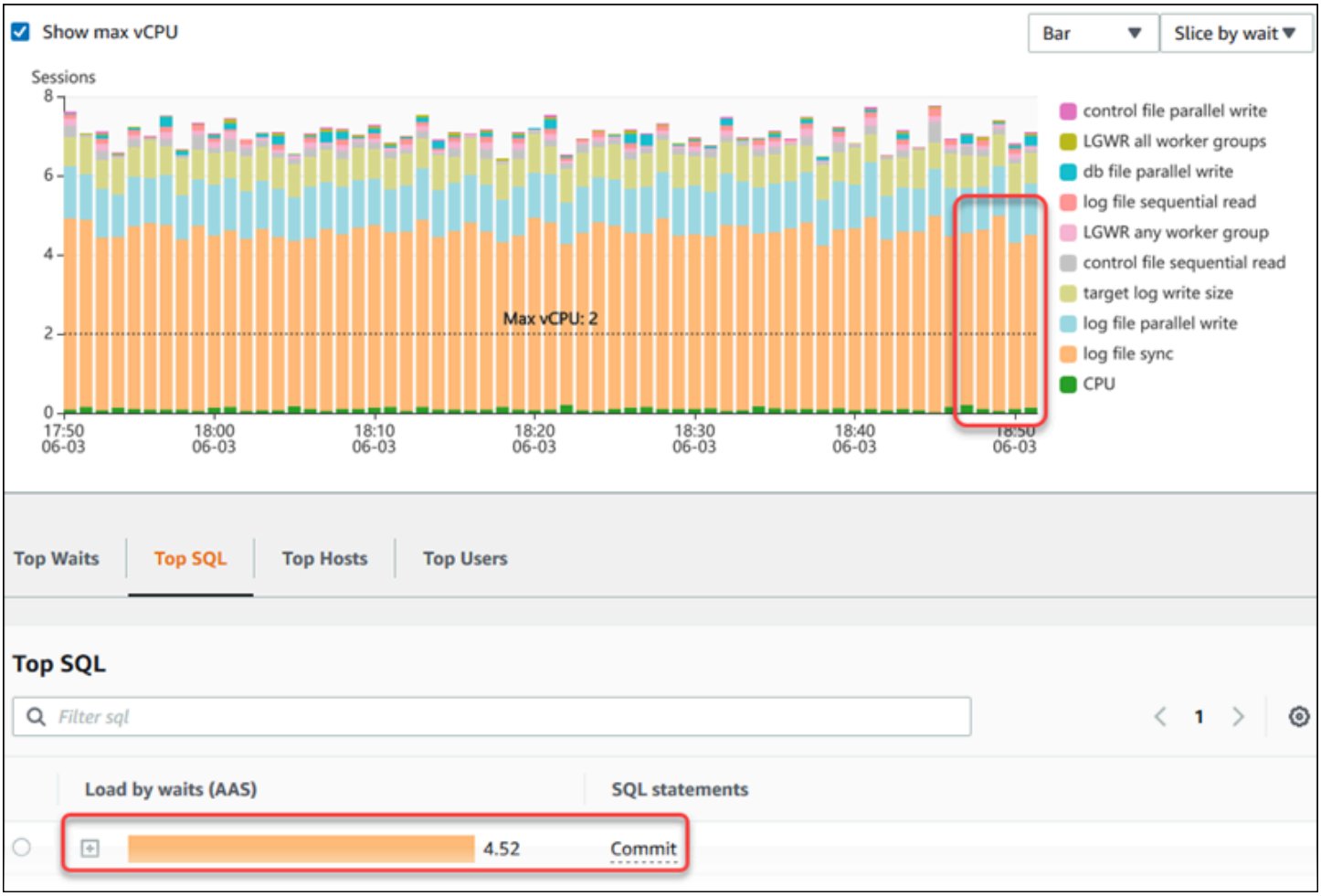
La vue par défaut du tableau de bord de Performance Insights affiche la charge de la base de données en fonction de l’attente et les principales requêtes SQL. Cette combinaison fournit en général la meilleure compréhension des problèmes de performances. L’affichage de la charge de la base de données en fonction de l’attente indique s’il existe des goulots d’étranglement liés aux ressources ou à des actions simultanées dans la base de données. Dans ce cas, l’onglet SQL du tableau Top Load Items (Principaux éléments de charge) indique les requêtes à l’origine de cette charge.

Votre flux de travail standard pour diagnostiquer les problèmes de performances se présente comme suit :

1. Dans le graphique Database load (Charge de la base de données), regardez s’il existe des incidents de charge de base de données qui dépassent la ligne Max CPU (CPU max).
2. Si c’est le cas, observez le graphique Database load (Charge de la base de données) et identifiez le ou les états d’attente qui sont les principaux responsables.
3. Identifiez les requêtes de hachage à l’origine de la charge en déterminant les requêtes du tableau Top Load Items (Principaux éléments de charge) de l’onglet SQL qui contribuent le plus à ces états d’attente. Vous pouvez les identifier dans la colonne DB Load by Wait (Charge de base de données par attente).
4. Choisissez l’une de ces requêtes de hachage dans l’onglet SQL pour la développer et afficher les requêtes enfants qui la composent.

Par exemple, dans le tableau de bord suivant, les attentes log file sync (synchronisation de fichier journal) constituent la majeure partie de la charge de base de données. Les attentes LGWR all worker groups sont également élevées. Le graphique Top SQL (Principaux éléments SQL) montre ce

qui provoque les attentes log file sync (synchronisation de fichier journal) : les instructions `COMMIT` fréquentes. Dans ce cas, une validation moins fréquente permet de réduire la charge de base de données.



Analyse des performances de base de données pour une période donnée

Analysez les performances des bases de données à l'aide d'une analyse à la demande en créant un rapport d'analyse des performances pour une période donnée. Affichez un rapport d'analyse des performances pour découvrir les problèmes de performances tels que les goulots d'étranglement ou les modifications d'une requête dans votre instance de base de données. Le tableau de bord d'analyse des performances vous permet de sélectionner une période et de créer un rapport d'analyse des performances. Vous pouvez également ajouter une ou plusieurs balises au rapport.

Pour utiliser cette fonctionnalité, vous devez utiliser la période de conservation du niveau payant. Pour plus d'informations, consultez Tarification et conservation des données pour Performance Insights.

Le rapport est disponible dans l'onglet Rapports d'analyse des performances – nouveau pour être sélectionné et affiché. Ce rapport contient les informations, les métriques associées et les recommandations permettant de résoudre le problème de performances. Le rapport peut être consulté pendant toute la durée de conservation de l'analyse des performances.

Le rapport est supprimé si l'heure de début de la période d'analyse du rapport se situe en dehors de la période de rétention. Vous pouvez également supprimer le rapport avant la fin de la période de conservation.

Pour détecter les problèmes de performances et générer le rapport d'analyse pour votre instance de base de données, vous devez activer l'analyse des performances. Pour plus d'informations sur l'activation de Performance Insights, consultez Activation ou désactivation de l'Analyse des performances pour Aurora.

Pour obtenir les informations de prise en charge de la région, du moteur de base de données et de la classe d'instance pour cette fonctionnalité, consultez Prise en charge de la classe d'instances, de la région et du moteur de base de données Amazon Aurora pour les fonctionnalités d'analyse des performances.

Dans les sections suivantes, vous pouvez créer, afficher, ajouter des balises et supprimer un rapport d'analyse des performances.

Rubriques

- Création d'un rapport d'analyse des performances dans Performance Insights
- Affichage d'un rapport d'analyse des performances dans Performance Insights
- Ajout de balises à un rapport d'analyse des performances dans Performance Insights
- Suppression d'un rapport d'analyse des performances dans Performance Insights

Création d'un rapport d'analyse des performances dans Performance Insights

Vous pouvez créer un rapport d'analyse des performances pour une période spécifique dans le tableau de bord d'analyse des performances. Vous pouvez sélectionner une période et ajouter une ou plusieurs balises au rapport d'analyse.

La période d'analyse peut aller de 5 minutes à 6 jours. Il doit y avoir au moins 24 heures de données de performance avant le début de l'analyse.

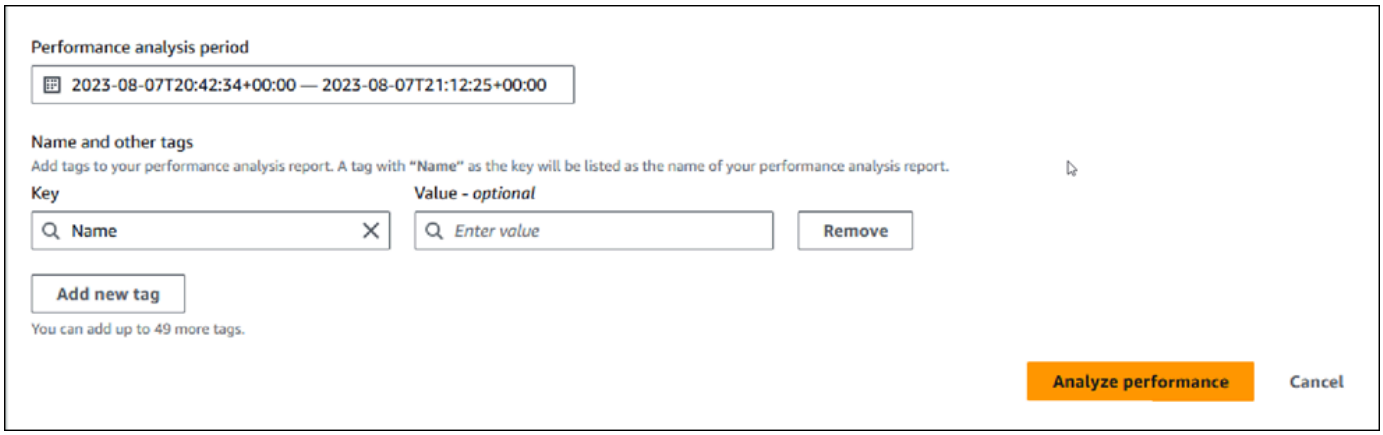
Pour obtenir les informations de prise en charge de la région, du moteur de base de données et de la classe d'instance pour cette fonctionnalité, consultez Prise en charge de la classe d'instances, de

la région et du moteur de base de données Amazon Aurora pour les fonctionnalités d'analyse des performances.

Pour créer un rapport d'analyse des performances pour une période donnée

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données.
4. Choisissez Analyser les performances dans la section Charge de base de données du tableau de bord.

   Les champs permettant de définir la période et d'ajouter une ou plusieurs balises au rapport d'analyse des performances s'affichent.

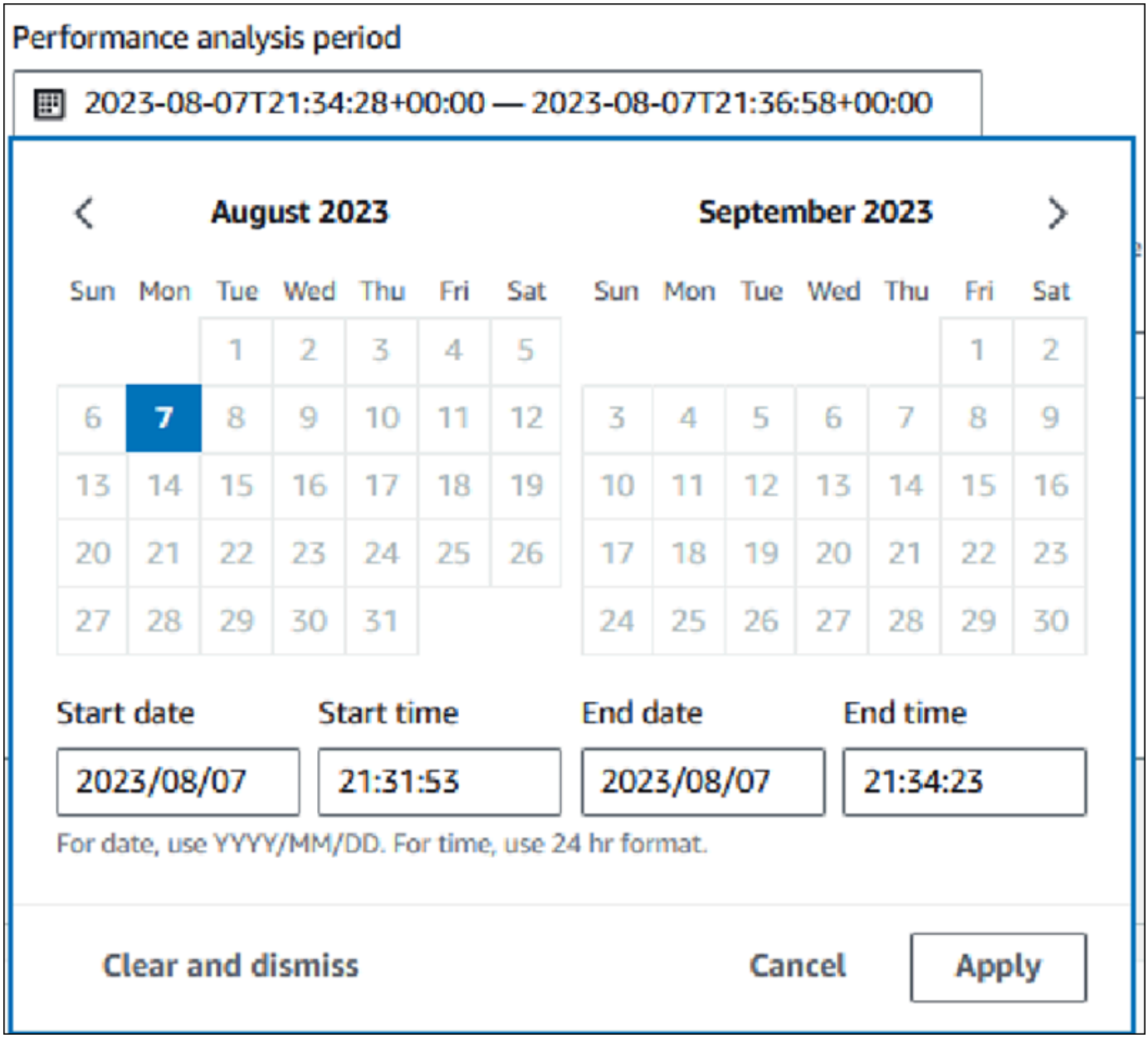


5. Choisissez une période. Si vous définissez une période dans la Plage relative ou dans la Plage absolue en haut à droite, vous pouvez uniquement saisir ou sélectionner la date et l'heure du rapport d'analyse au cours de cette période. Si vous sélectionnez la période d'analyse en dehors de cette période, un message d'erreur s'affiche.

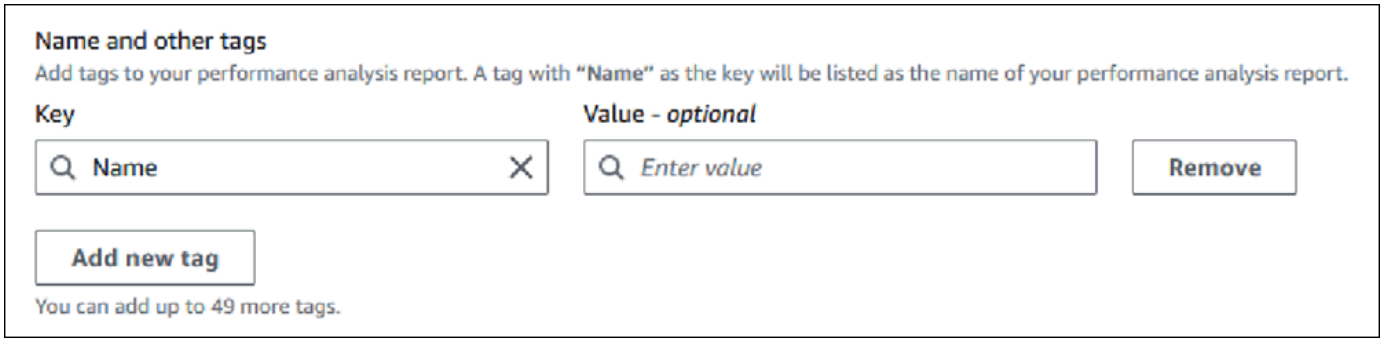
   Pour définir la période, vous pouvez effectuer l'une des opérations suivantes :

   - Appuyez sur l'un des curseurs du graphique de charge de base de données et faites-le glisser.

     La zone Période d'analyse des performances affiche la période sélectionnée et le graphique de charge de base de données met en évidence la période sélectionnée.
   - Choisissez les paramètres Date de début, Heure de début,Date de fin et Heure de fin dans la zone Période d'analyse des performances.



6. (Facultatif) Entrez Clé et Valeur-facultatif pour ajouter une balise pour le rapport.



7. Choisissez Analyser les performances.

   Une bannière affiche un message indiquant si la génération du rapport est réussie ou a échoué. Le message fournit également le lien permettant de consulter le rapport.

   L'exemple suivant montre la bannière avec le message de réussite de création du rapport.

   Le rapport peut être consulté dans l'onglet Rapports d'analyse des performances – nouveau.

Vous pouvez créer un rapport d'analyse des performances à l'aide de l'interface AWS CLI. Pour obtenir un exemple qui illustre la création d'un rapport à l'aide de l'interface AWS CLI, consultez Création d'un rapport d'analyse des performances pour une période donnée.

Affichage d'un rapport d'analyse des performances dans Performance Insights

L'onglet Rapports d'analyse des performances – nouveau répertorie tous les rapports créés pour l'instance de base de données. Pour chaque test, les résultats des tests suivants sont affichés :

- ID : identifiant unique du rapport.
- Nom : clé de balise ajoutée au rapport.
- Heure de création du rapport : heure à laquelle vous avez créé le rapport.
- Heure de début de l'analyse : heure de début de l'analyse dans le rapport.
- Heure de fin de l'analyse : heure de fin de l'analyse dans le rapport.

Pour afficher un rapport d'analyse des performances

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données pour laquelle vous souhaitez consulter le rapport d'analyse.
4. Faites défiler la page vers le bas et choisissez l'onglet Rapports d'analyse des performances – nouveau dans le tableau de bord Performance Insights.

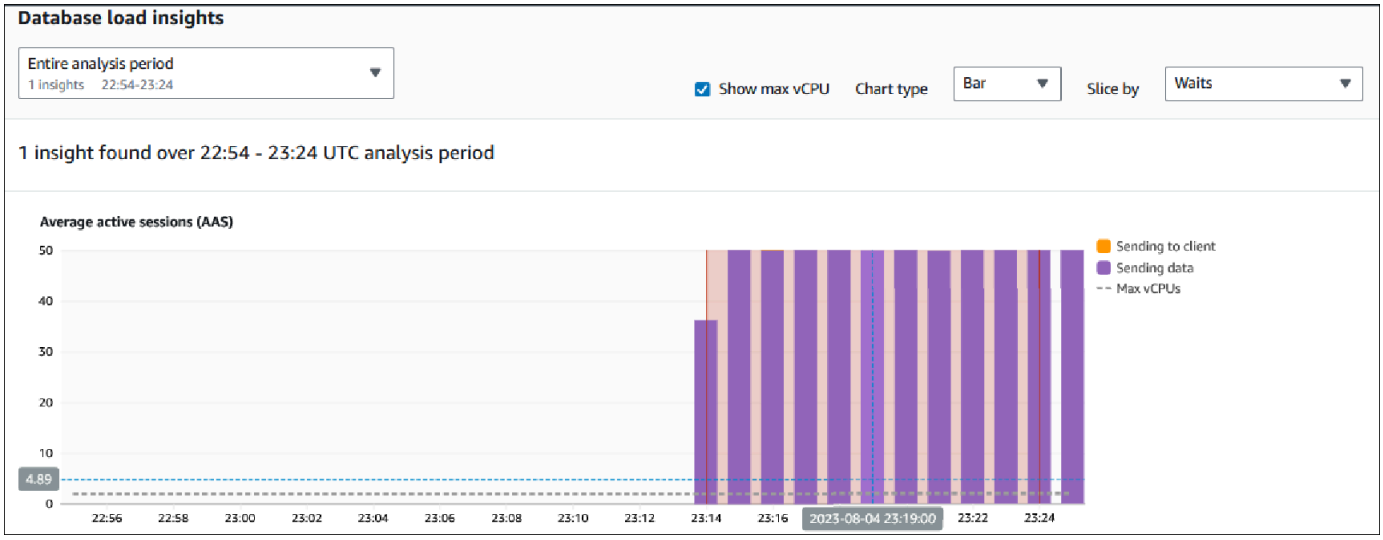
   Tous les rapports d'analyse pour les différentes périodes sont affichés.

5. Choisissez ID du rapport que vous souhaitez consulter.

   Le graphique de charge de base de données affiche la période d'analyse complète par défaut si plusieurs informations sont identifiées. Si le rapport a identifié une information, le graphique de charge de base de données affiche par défaut cette information.

   Le tableau de bord répertorie également les balises du rapport dans la section Balises.

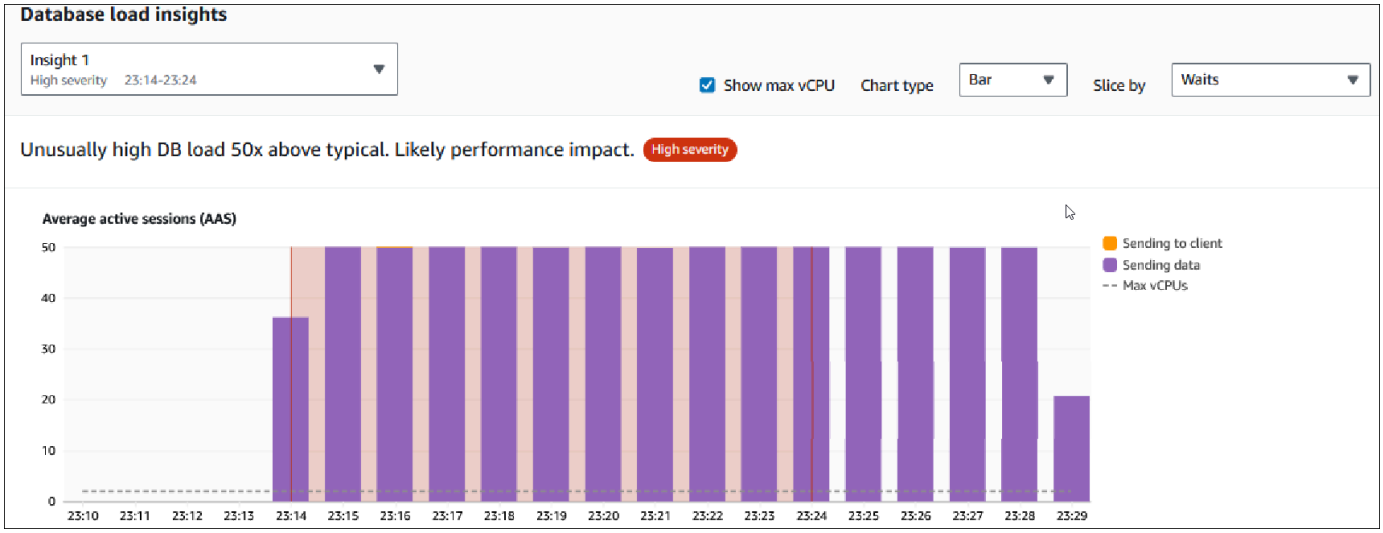
   L'exemple suivant montre l'ensemble de la période d'analyse du rapport.

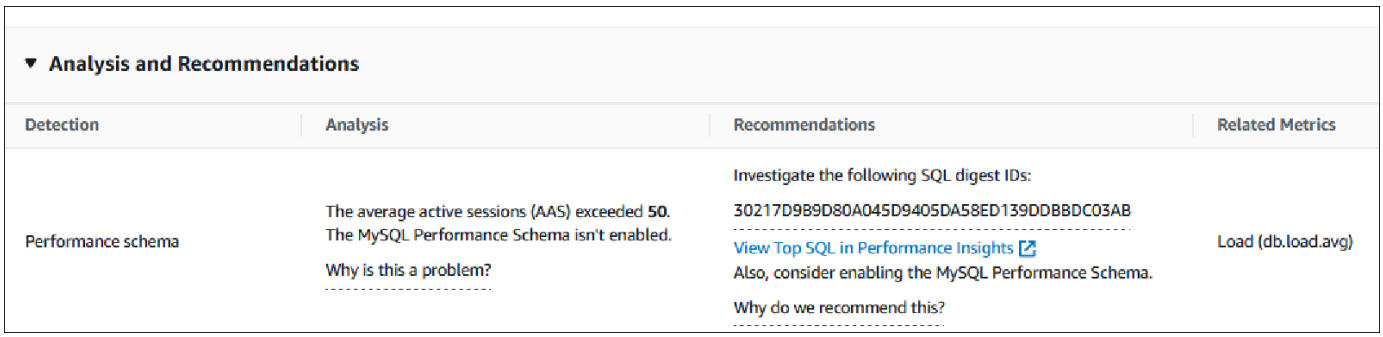


6. Choisissez l'information dans la liste Informations de charge de base de données que vous souhaitez consulter si plusieurs informations sont identifiées dans le rapport.

   Le tableau de bord affiche le message d'information, le graphique de charge de base de données mettant en évidence la période couverte par les informations, l'analyse et les recommandations, ainsi que la liste des balises de rapport.

   L'exemple suivant montre l'information de charge de base de données dans le rapport.





Ajout de balises à un rapport d'analyse des performances dans Performance Insights

Vous pouvez ajouter une balise lorsque vous créez ou consultez un rapport. Vous pouvez ajouter jusqu'à 50 balises par rapport.

Vous avez besoin d'autorisations pour ajouter les balises. Pour plus d'informations sur les stratégies d'accès pour l'analyse des performances, consultez Configuration des politiques d'accès pour Performance Insights.

Pour ajouter une ou plusieurs balises lors de la création d'un rapport, consultez l'étape 6 de la procédure Création d'un rapport d'analyse des performances dans Performance Insights.

Pour ajouter une ou plusieurs balises lors de la consultation d'un rapport

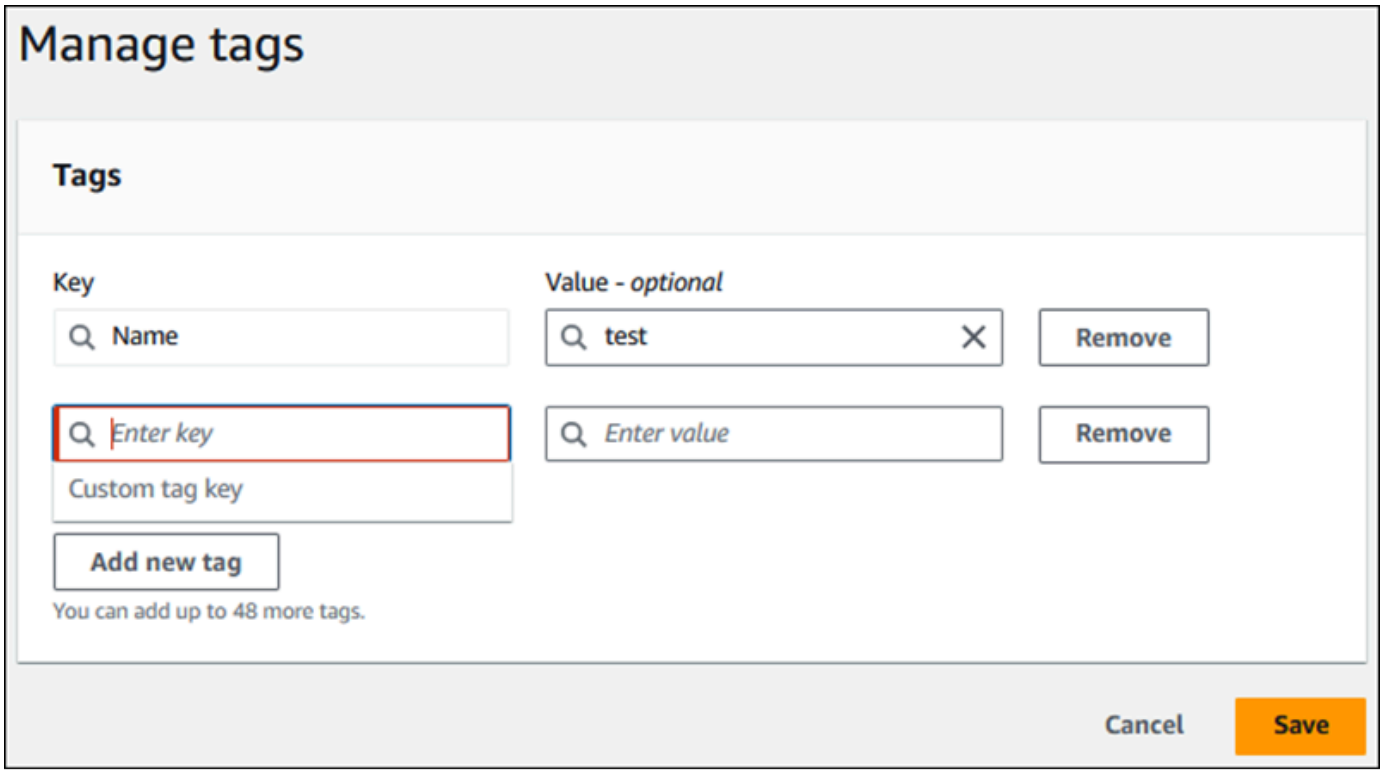
1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données.

Le tableau de bord Performance Insights correspondant à cette instance de base de données s'affiche.

4. Faites défiler la page vers le bas et choisissez Rapports d'analyse des performances – nouveau.
5. Choisissez le rapport pour lequel vous souhaitez ajouter les balises.

   Le tableau de bord affiche le rapport.
6. Faites défiler vers le bas jusqu'à Balises et choisissez Gérer les balises.
7. Sélectionnez Ajouter une nouvelle balise.
8. Entrez la Clé et la Valeur – facultatif, puis choisissez Ajouter une nouvelle balise.

   L'exemple suivant fournit la possibilité d'ajouter une nouvelle balise pour le rapport sélectionné.



Une nouvelle balise est créée pour le rapport.

La liste des balises du rapport est affichée dans la section Balises du tableau de bord. Si vous souhaitez supprimer une balise du rapport, choisissez Supprimer à côté de la balise.

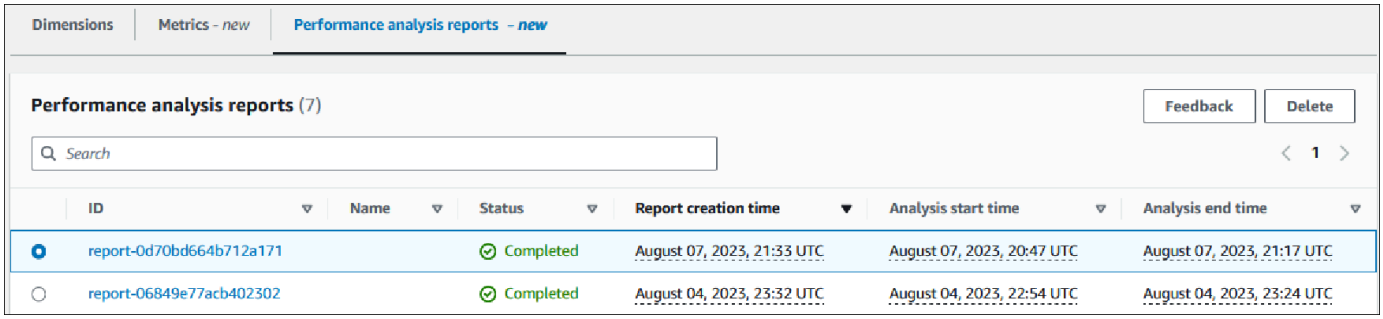
Suppression d’un rapport d’analyse des performances dans Performance Insights

Vous pouvez supprimer un rapport de la liste des rapports affichée dans l’onglet Rapports d’analyse des performances ou lors de l’affichage d’un rapport.

Pour supprimer un rapport

1. Ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. Choisissez une instance de base de données.

   Le tableau de bord Performance Insights correspondant à cette instance de base de données s’affiche.
4. Faites défiler la page vers le bas et choisissez Rapports d’analyse des performances – nouveau.
5. Sélectionnez le rapport que vous souhaitez supprimer et choisissez Supprimer en haut à droite.



   Une fenêtre de confirmation s’affiche. Le rapport est supprimé une fois que vous avez choisi de confirmer.
6. (Facultatif) Choisissez l’ID du rapport que vous souhaitez supprimer.

   Dans le coin supérieur droit de la page du rapport, choisissez Supprimer.

   Une fenêtre de confirmation s’affiche. Le rapport est supprimé une fois que vous avez choisi de confirmer.

Analyse des requêtes à l’aide de l’onglet Top SQL dans Performance Insights

Dans le tableau de bord Amazon RDS Performance Insights, vous pouvez trouver des informations sur les requêtes en cours d’exécution et récentes dans l’onglet Top SQL (Principaux éléments SQL) du tableau Top dimensions (Dimensions principales). Vous pouvez utiliser ces informations pour régler vos requêtes.

Rubriques

- Présentation de l’onglet Top SQL (Principaux éléments SQL)
- Accès à plus de texte SQL dans le tableau de bord Performance Insights
- Affichage des statistiques SQL dans le tableau de bord de Performance Insights

Présentation de l’onglet Top SQL (Principaux éléments SQL)

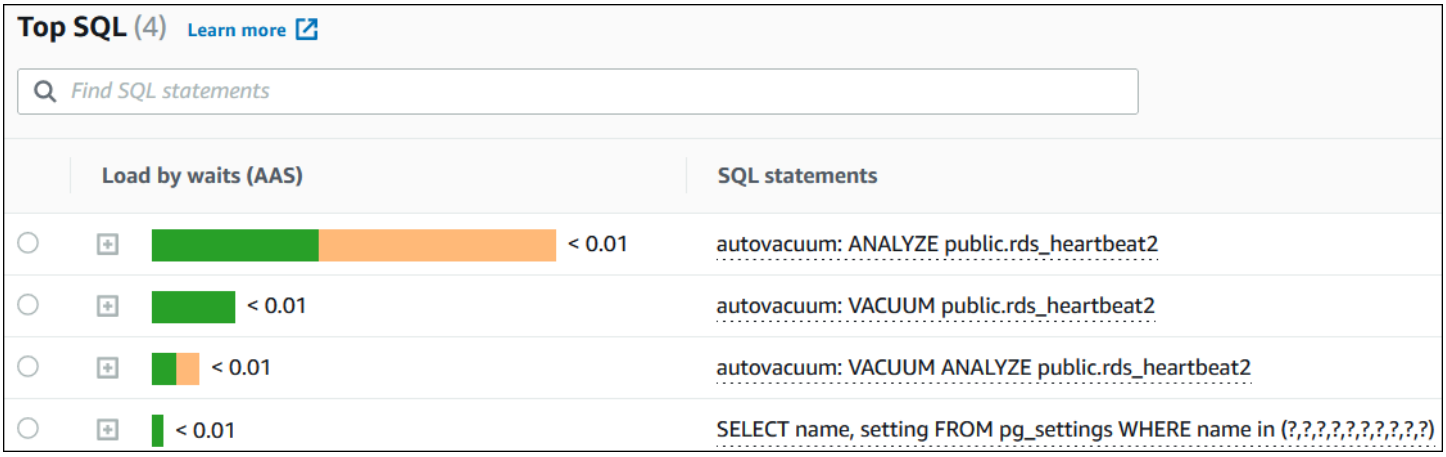
Par défaut, l’onglet Top SQL (SQL maximum) présente les 25 requêtes qui contribuent le plus à la charge de la base de données. Pour faciliter le réglage de vos requêtes, vous pouvez analyser certaines informations comme le texte de la requête et les statistiques SQL. Vous pouvez également choisir les statistiques à afficher dans l’onglet Top SQL (Principaux éléments SQL).

Rubriques

- Texte SQL
- Statistiques SQL
- Load by waits (AAS) [Charge par attentes (AAS)]
- Affichage des informations SQL
- Choix des préférences de statistiques

Texte SQL

Par défaut, chaque ligne du tableau Top SQL (SQL maximum) affiche 500 octets de texte pour chaque instruction.

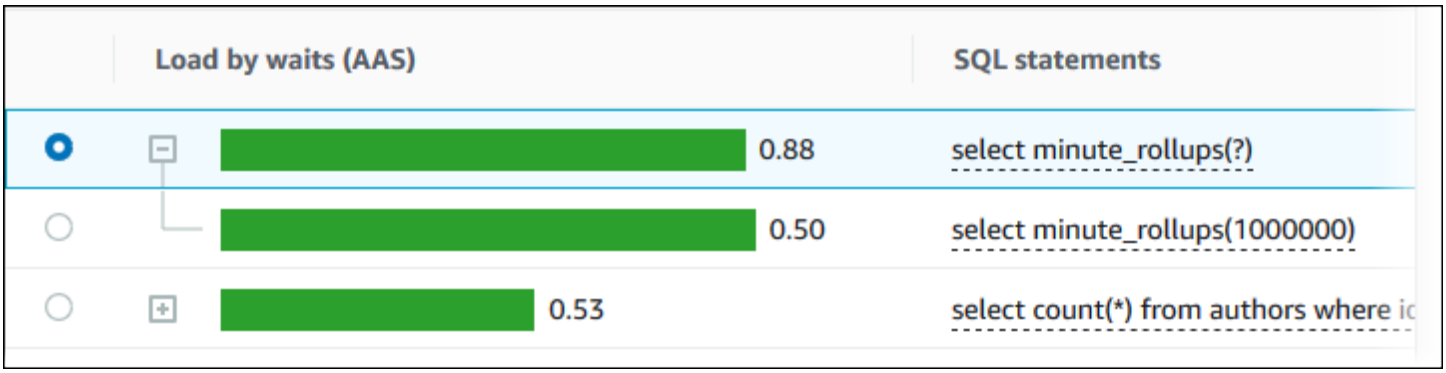


Pour savoir comment afficher plus que les 500 octets de texte SQL par défaut, consultez Accès à plus de texte SQL dans le tableau de bord Performance Insights.

Un récapitulatif SQL se compose de plusieurs requêtes réelles et structurellement similaires, mais dont les valeurs littérales peuvent être différentes. Le récapitulatif remplace les valeurs codées en dur par un point d'interrogation. Ainsi, `SELECT * FROM emp WHERE lname= ?` est un exemple de récapitulatif. Ce récapitulatif peut inclure les requêtes enfant suivantes :

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

Pour afficher les instructions SQL littérales dans un récapitulatif, sélectionnez la requête, puis choisissez le symbole plus (+). Dans l'exemple suivant, la requête sélectionnée est un récapitulatif.

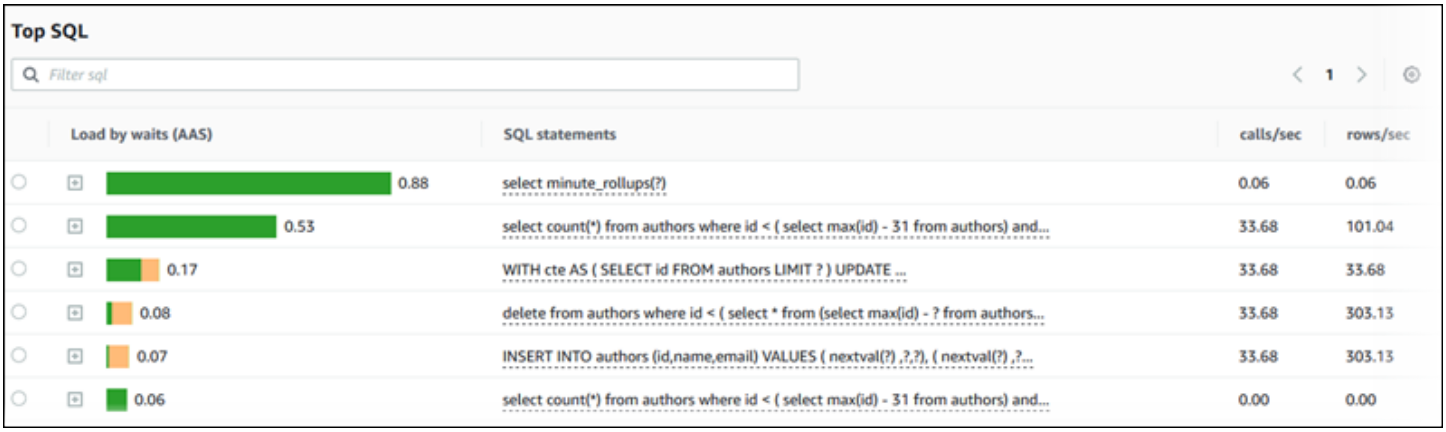


> **Note**
>
> Un récapitulatif SQL regroupe des instructions SQL similaires, mais ne censure pas les informations sensibles.

Statistiques SQL

Les statistiques SQL sont des métriques de performances qui concernent les requêtes SQL. Par exemple, Performance Insights peut montrer le nombre d'exécutions par seconde ou le nombre de lignes traitées par seconde. Performance Insights collecte des statistiques uniquement pour les requêtes les plus courantes. Généralement, celles-ci correspondent aux requêtes les plus importantes par charge affichées dans le tableau de bord Performance Insights.

Chaque ligne figurant dans le tableau Top SQL (Principaux éléments SQL) présente des statistiques pertinentes pour l'instruction ou le résumé SQL, comme le montre l'exemple suivant.



Performance Insights peut renvoyer `0.00` et `-` (inconnu) pour les statistiques SQL. Cette situation se produit dans les conditions suivantes :

- Il n'existe qu'un seul exemple. Par exemple, Performance Insights calcule les taux de modification pour les requêtes Aurora PostgreSQL sur la base de plusieurs exemples de la vue `pg_stat_statements`. Lorsqu'une charge de travail est exécutée pendant une courte période, Performance Insights peut ne collecter qu'un seul exemple, ce qui signifie qu'il ne peut pas calculer le taux de modification. La valeur inconnue est représentée par un tiret (`-`).
- Deux exemples ont les mêmes valeurs. Performance Insights ne peut pas calculer un taux de modification, car aucun changement n'a eu lieu, il rapporte donc le taux comme `0.00`.
- Il manque un identifiant valide à une déclaration Aurora PostgreSQL. PostgreSQL crée un identifiant pour une déclaration seulement après l'analyse. Ainsi, une déclaration peut exister dans les structures internes en mémoire de PostgreSQL sans identifiant. Comme Performance Insights échantillonne les structures internes en mémoire une fois par seconde, les requêtes à faible latence peuvent n'apparaître que pour un seul exemple. Si l'identifiant de la requête n'est pas disponible pour cet exemple, Performance Insights ne peut pas associer cette déclaration à ses statistiques. La valeur inconnue est représentée par un tiret (`-`).

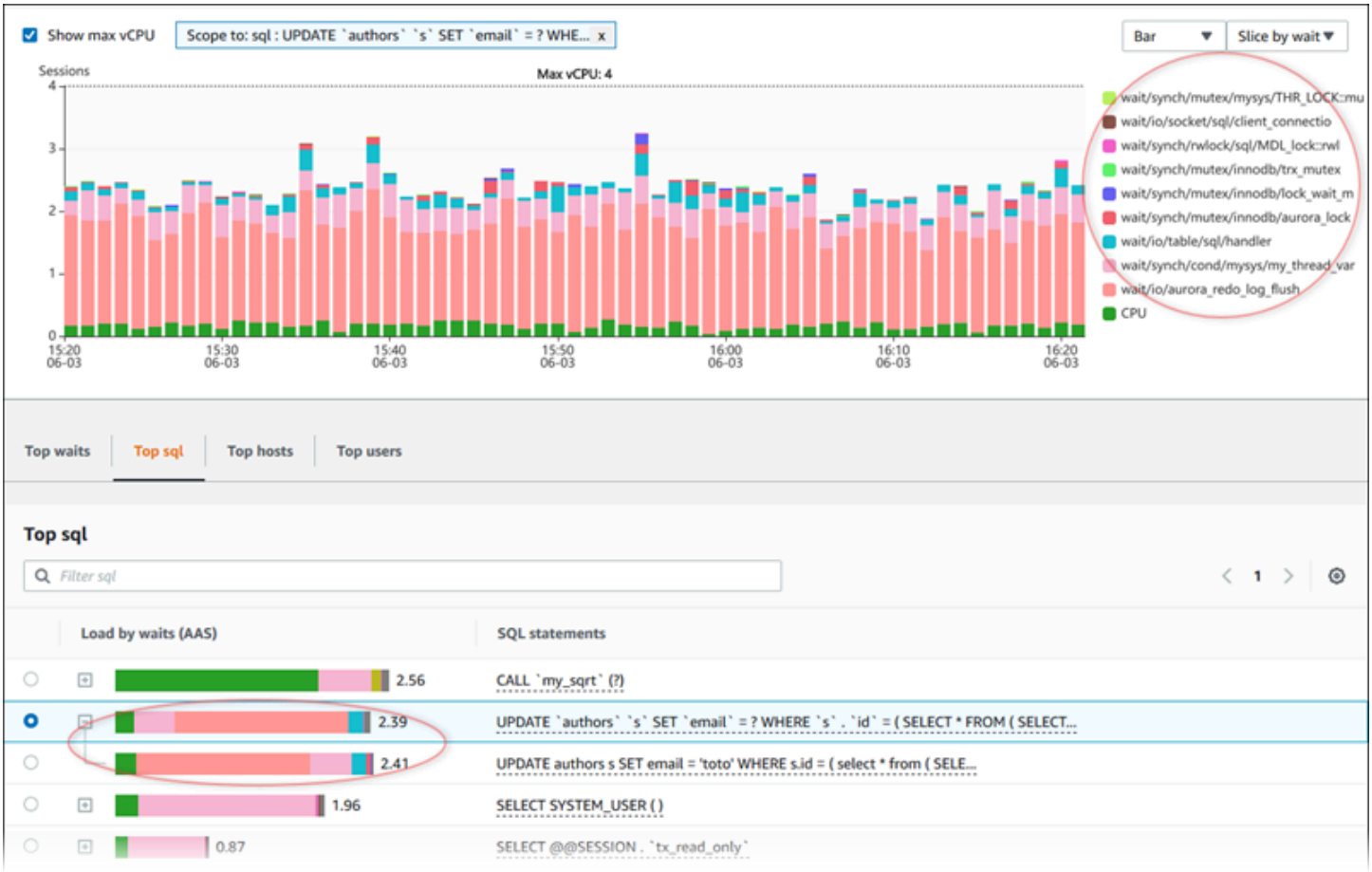
Pour obtenir une description des statistiques SQL pour les moteurs Aurora, consultez Statistiques SQL pour Performance Insights.

Load by waits (AAS) [Charge par attentes (AAS)]

Dans Top SQL (Principaux éléments SQL), la colonne Load by waits (AAS) [Charge par attentes (AAS)] illustre le pourcentage de la charge de base de données associée à chacun des principaux éléments de charge. Cette colonne reflète la charge pour cet élément selon le regroupement

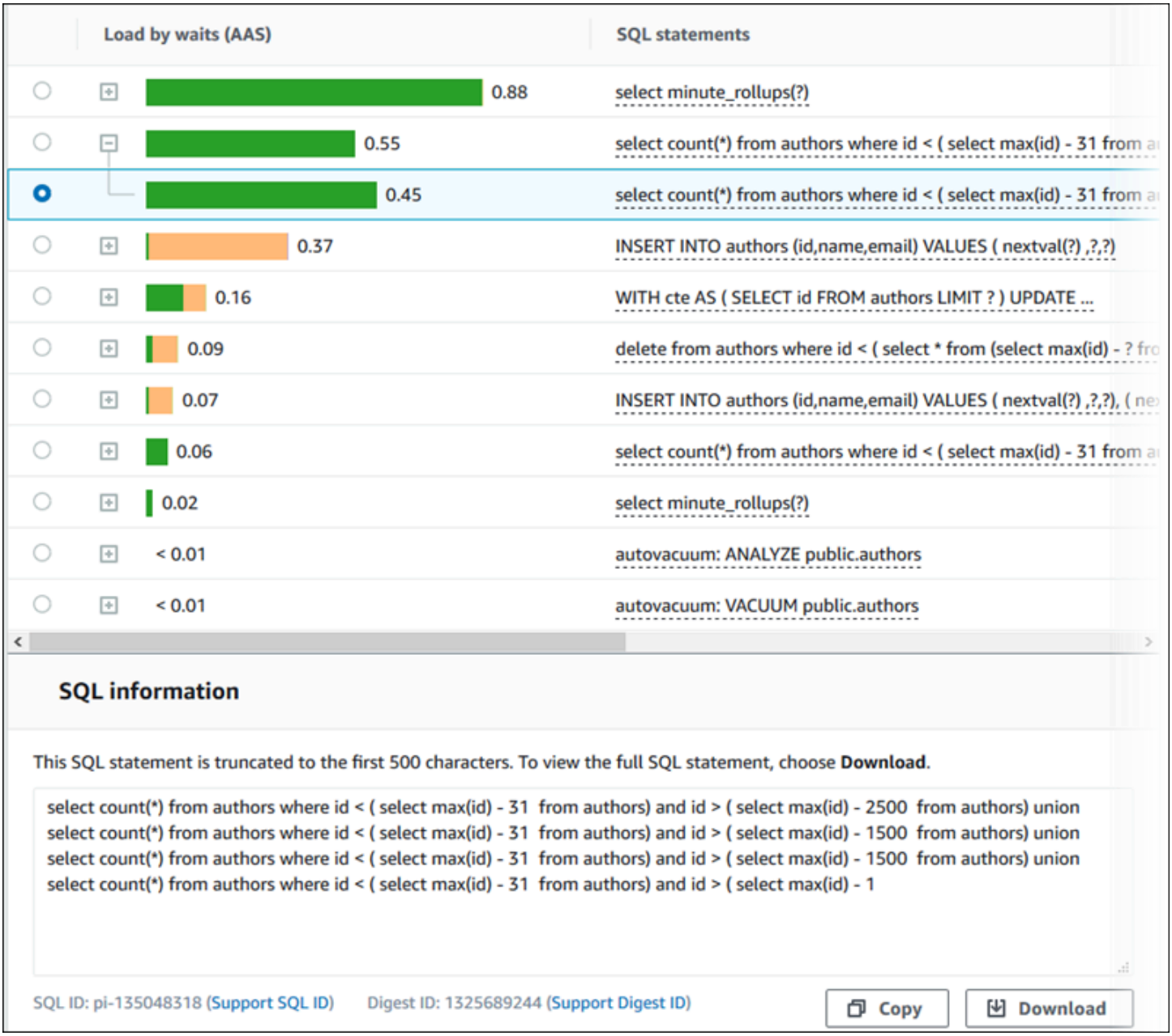
actuellement sélectionné dans DB Load Chart (Graphique de charge de base de données). Pour plus d'informations sur la moyenne des sessions actives (AAS), consultez Sessions actives en moyenne.

Par exemple, vous pouvez regrouper le graphique DB Load (Charge de la base de données) par états d'attente. Vous examinez les requêtes SQL dans le tableau des principaux éléments de charge. Dans ce cas, la dimension, la segmentation et le code de couleurs de la barre DB Load by Waits (Charge de base de données par attente) représentent la proportion du temps d'un état d'attente donné auquel cette requête contribue. Cette barre indique également les états d'attente qui affectent la requête sélectionnée.



Affichage des informations SQL

Dans le tableau Top SQL (Principaux éléments SQL), vous pouvez ouvrir une instruction pour examiner ses informations. Les informations s'affichent dans le volet inférieur.

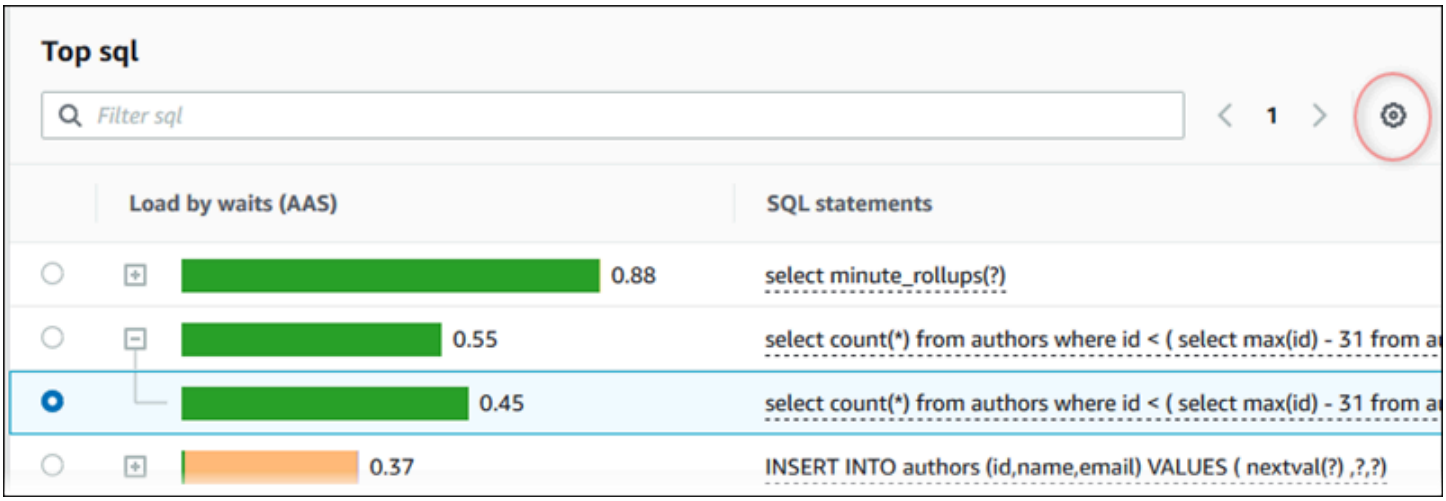


Les types d'identifiants (ID) associés à des instructions SQL sont les suivants :

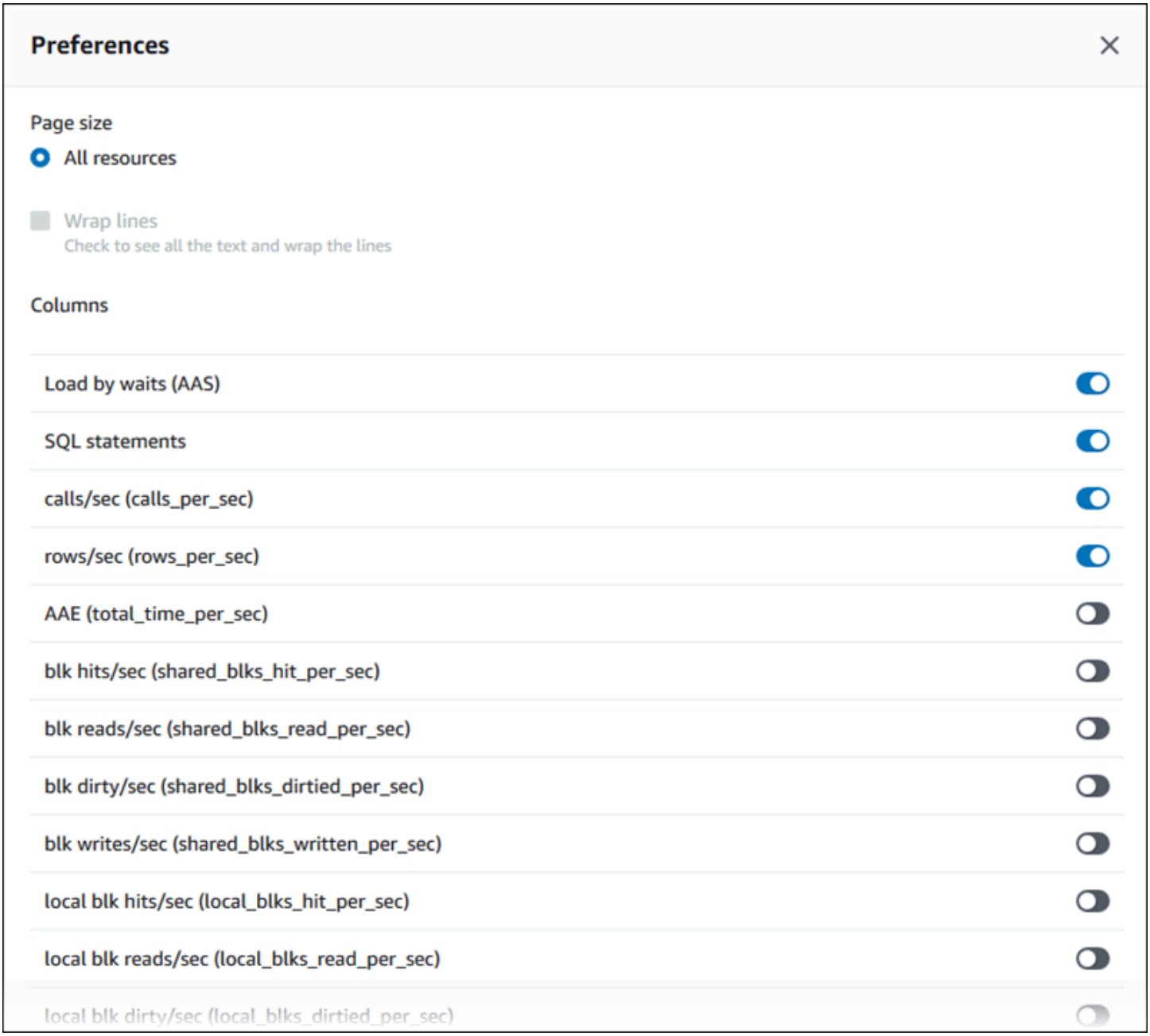
- ID SQL de support – Valeur de hachage de l'ID SQL. Cette valeur est uniquement destinée à référencer un ID SQL lorsque vous utilisez AWS Support. AWS Support n'a pas accès à vos ID SQL réels et au texte SQL.
- Support Digest ID (ID digest de support) – Valeur de hachage de l'ID digest. Cette valeur est uniquement destinée à référencer un ID digest lorsque vous utilisez AWS Support. AWS Support n'a pas accès à vos ID digest réels et au texte SQL.

Choix des préférences de statistiques

Vous pouvez contrôler les statistiques qui s'affichent dans l'onglet Top SQL (Principaux éléments SQL) en choisissant l'icône Preferences (Préférences).



Lorsque vous choisissez l'icône Préférences, la fenêtre Préférences s'ouvre. La capture d'écran suivante est un exemple de la fenêtre Preferences (Préférences).



Pour activer les statistiques afin de les faire apparaître dans l'onglet Top SQL (Principaux éléments SQL), utilisez votre souris pour faire défiler l'écran jusqu'au bas de la fenêtre, puis choisissez Continue (Continuer).

Pour plus d'informations sur les statistiques par seconde ou par appel pour les moteurs Aurora, consultez des statistiques SQL spécifiques au moteur dans Statistiques SQL pour Performance Insights

Accès à plus de texte SQL dans le tableau de bord Performance Insights

Par défaut, chaque ligne du tableau Top SQL (Principaux éléments SQL) affiche 500 octets de texte SQL pour chaque instruction SQL.

Lorsqu'une instruction SQL dépasse 500 octets, vous pouvez afficher davantage de texte dans la section SQL text (Texte SQL) située sous le tableau Top SQL (Top SQL). Dans ce cas, la longueur maximale du texte affiché dans SQL text (Texte SQL) est de 4 Ko. Cette limite est imposée par la console et est soumise aux limites fixées par le moteur de base de données. Pour enregistrer le texte affiché dans SQL text (Texte SQL), sélectionnez Download (Télécharger).

Rubriques

- Limites de taille de texte pour Aurora MySQL
- Définition de la limite de taille d'un texte SQL pour les instances de base de données Aurora PostgreSQL
- Affichage et téléchargement de texte SQL dans le tableau de bord de Performance Insights

Limites de taille de texte pour Aurora MySQL

Lorsque vous téléchargez du texte SQL, le moteur de la base de données détermine sa longueur maximale. Vous pouvez télécharger du texte SQL jusqu'aux limites suivantes par moteur.

Moteur de base de données	Longueur maximale du texte téléchargé
Aurora MySQL	La longueur est fixée à 4 096 octets.

La section SQL text (Texte SQL) de la console Performance Insights affiche jusqu'au la taille maximum renvoyée par le moteur. Par exemple, si Aurora MySQL renvoie au plus 1 Ko à Performance Insights, celui-ci ne peut collecter et afficher que 1 Ko, même si la requête d'origine est plus volumineuse. Ainsi, lorsque vous visualisez la requête en SQL text (Texte SQL) ou que vous la téléchargez, Performance Insights renvoie le même nombre d'octets.

Si vous utilisez l'AWS CLI ou l'API, Performance Insights n'a pas la limite de 4 Ko imposée par la console. `DescribeDimensionKeys` et `GetResourceMetrics` renvoient au maximum 500 octets.

> **Note**
>
> `GetDimensionKeyDetails` renvoie la requête complète, mais la taille dépend de la limite du moteur.

Définition de la limite de taille d'un texte SQL pour les instances de base de données Aurora PostgreSQL

Aurora PostgreSQL gère le texte différemment. Vous pouvez définir la limite de taille du texte avec le paramètre `track_activity_query_size` de l'instance de base de données. Ce paramètre possède les caractéristiques suivantes :

Taille de texte par défaut

Sur Aurora PostgreSQL version 9.6, la valeur par défaut du paramètre `track_activity_query_size` est 1 024 octets. Sur Aurora PostgreSQL version 10 ou versions ultérieures, la valeur par défaut est 4 096 octets.

Taille maximale du text

La limite de `track_activity_query_size` est de 102 400 octets pour Aurora PostgreSQL version 12 et versions inférieures. Le maximum est de 1 Mo pour la version 13 et versions ultérieures.

Si le moteur renvoie 1 Mo à Performance Insights, la console affiche uniquement les 4 premiers Ko. Si vous téléchargez la requête, vous obtenez la totalité des 1 Mo. Dans ce cas, l'affichage et le téléchargement renvoient des quantités différentes d'octets. Pour plus d'informations sur le paramètre `track_activity_query_size` d'instance de base de données, consultez Run-time Statistics dans la documentation PostgreSQL.

Pour augmenter la taille du texte SQL, augmentez la limite `track_activity_query_size`. Pour modifier ce paramètre, modifiez sa valeur dans le groupe de paramètres associé à l'instance de base de données Aurora PostgreSQL.

Pour modifier le paramètre lorsque l'instance utilise le groupe de paramètres par défaut

1. Créez un nouveau groupe de paramètres pour l'instance de base de données, associé au moteur de base de données et à sa version appropriés.

2. Définissez le paramètre dans le nouveau groupe de paramètres.
3. Associez le nouveau groupe de paramètres à l'instance de base de données.

Pour plus d'informations sur la définition d'un paramètre d'instance de base de données, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora.

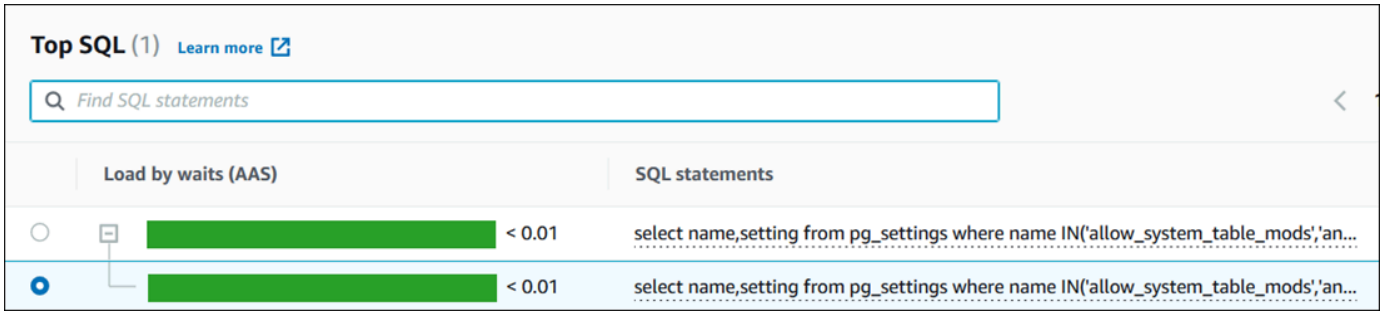
Affichage et téléchargement de texte SQL dans le tableau de bord de Performance Insights

Dans le tableau de bord de Performance Insights, vous pouvez afficher ou télécharger le texte SQL.

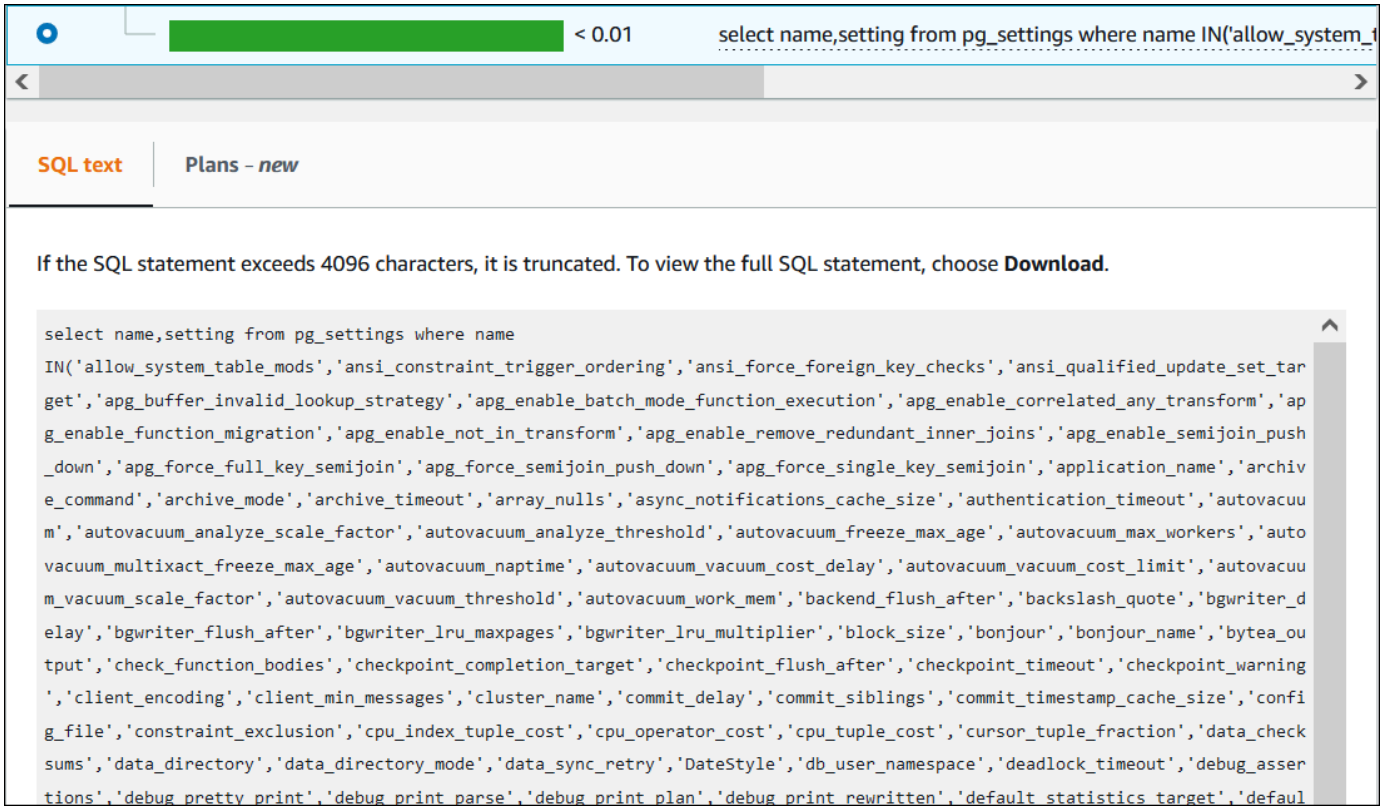
Pour afficher du texte SQL supplémentaire dans le tableau de bord de Performance Insights

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Performance Insights.
3. Choisissez une instance de base de données.
4. Faites défiler la page jusqu'à l'onglet Principaux SQL dans le tableau de bord Performance Insights.
5. Choisissez le signe plus pour développer une synthèse SQL et choisissez l'une des requêtes enfants de la synthèse.

   Les instructions SQL dont la taille du texte est supérieure à 500 octets ressemblent à l'image ci-dessous.



6. Faites défiler jusqu'à l'onglet SQL text (Texte SQL).



Le tableau de bord de Performance Insights peut afficher jusqu'à 4 096 octets par instruction SQL.

7. (Facultatif) Choisissez Copy (Copier) pour copier l'instruction SQL affichée ou Download (Télécharger) pour télécharger l'instruction SQL et en afficher le texte jusqu'à la limite du moteur de base de données.

> **Note**
>
> Pour copier ou télécharger l'instruction SQL, désactivez les bloqueurs de fenêtres contextuelles.

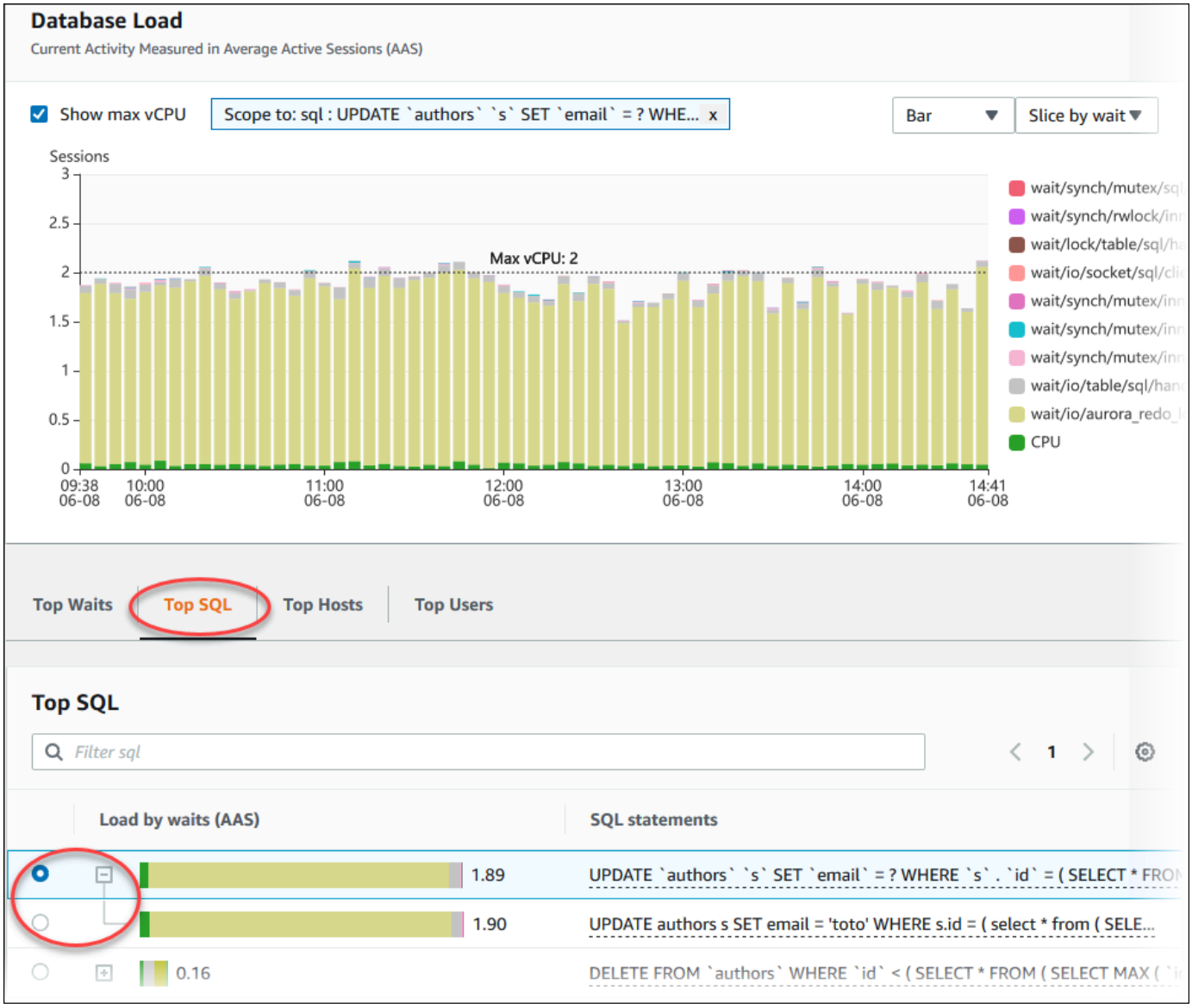
Affichage des statistiques SQL dans le tableau de bord de Performance Insights

Dans le tableau de bord de Performance Insights, les statistiques SQL sont disponibles dans l'onglet Top SQL (Principaux éléments SQL) du graphique Database load (Charge de la base de données).

Pour afficher les statistiques SQL

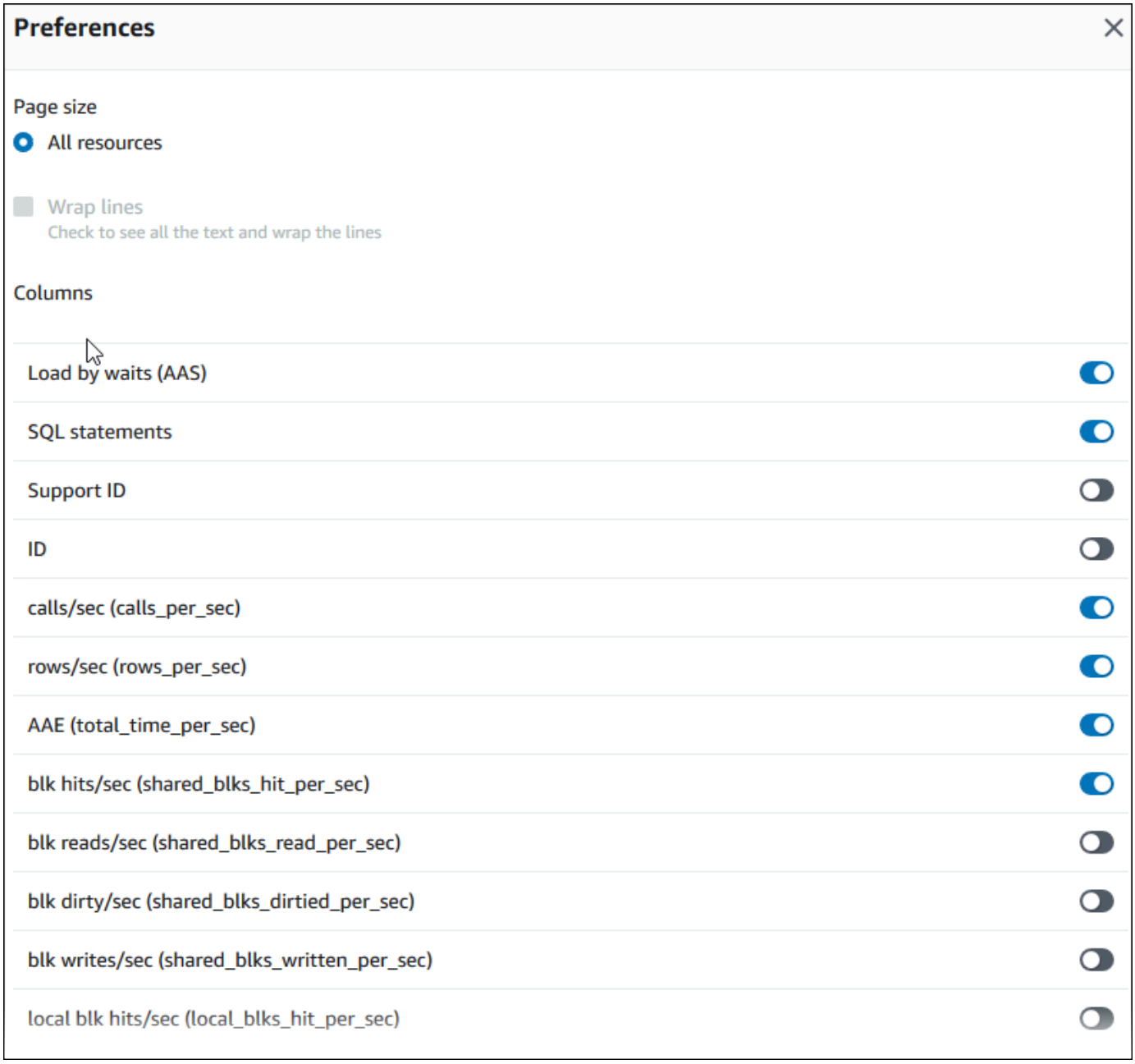
1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

2. Dans le panneau de navigation gauche, choisissez Performance Insights.
3. En haut de la page, choisissez la base de données dont vous voulez voir les statistiques SQL.
4. Faites défiler jusqu'au bas de la page et choisissez l'onglet Top SQL (Principaux éléments SQL).
5. Choisissez une déclaration individuelle(Aurora MySQL only) (Aurora MySQL uniquement) ou une requête récapitulative.

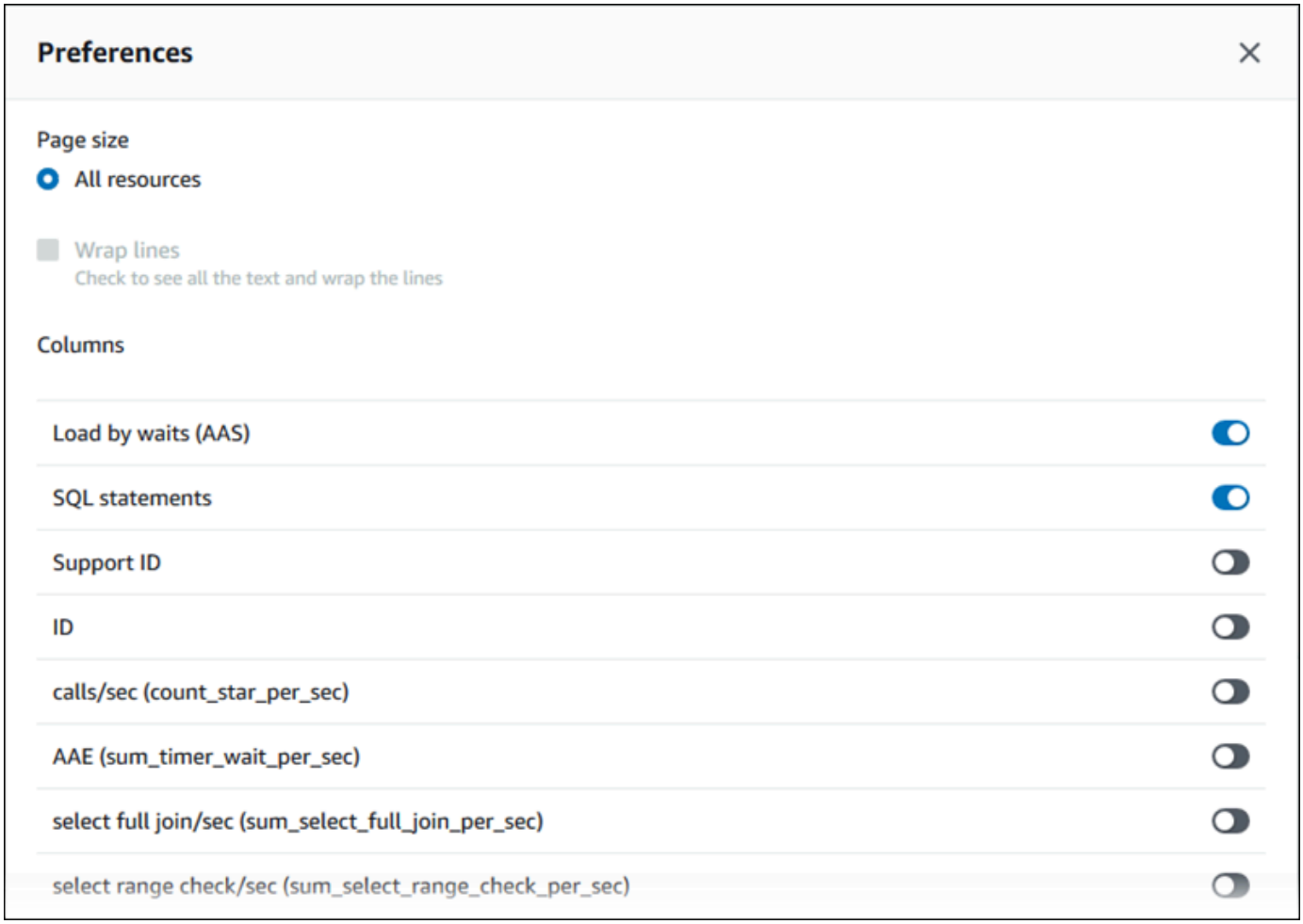


6. Choisissez les statistiques à afficher en sélectionnant l'icône en forme d'engrenage dans le coin supérieur droit du graphique. Pour obtenir des descriptions des statistiques SQL pour les moteurs Amazon RDSAurora, consultez Statistiques SQL pour Performance Insights.

   L'exemple suivant présente les préférences pour Aurora PostgreSQL.



L’exemple suivant montre les préférences pour les instances de base de données Aurora MySQL.



7. Choisissez Save (Enregistrer) pour enregistrer vos préférences.

   La table Top SQL (Principaux éléments SQL) s'actualise.

Affichage des recommandations proactives de Performance Insights

Analyse des performances d'Amazon RDS surveille des indicateurs spécifiques et crée automatiquement des seuils en analysant les niveaux susceptibles de poser problème pour une ressource spécifique. Lorsque les nouvelles valeurs métriques dépassent un seuil prédéfini sur une période donnée, Performance Insights génère une recommandation proactive. Cette recommandation permet d'éviter tout impact futur sur les performances de la base de données. Pour bénéficier de ces recommandations proactives, vous devez activer Performance Insights avec une période de conservation payante.

Pour plus d'informations sur l'activation de Performance Insights, consultez Activation ou désactivation de l'Analyse des performances pour Aurora. Pour en savoir plus sur la tarification et

la conservation des données pour Performance Insights, consultez Tarification et conservation des données pour Performance Insights.

Pour connaître les régions, les moteurs de base de données et les classes d'instance pris en charge pour les recommandations proactives, consultez Prise en charge de la classe d'instances, de la région et du moteur de base de données Amazon Aurora pour les fonctionnalités d'analyse des performances.

Vous pouvez afficher l'analyse détaillée et les investigations recommandées concernant les recommandations proactives sur la page de détails des recommandations.

Pour plus d'informations sur les recommandations, consultez Recommandations d'Amazon Aurora.

Pour afficher l'analyse détaillée d'une recommandation proactive

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, effectuez l'une des opérations suivantes :

   - Choisissez Recommandations.

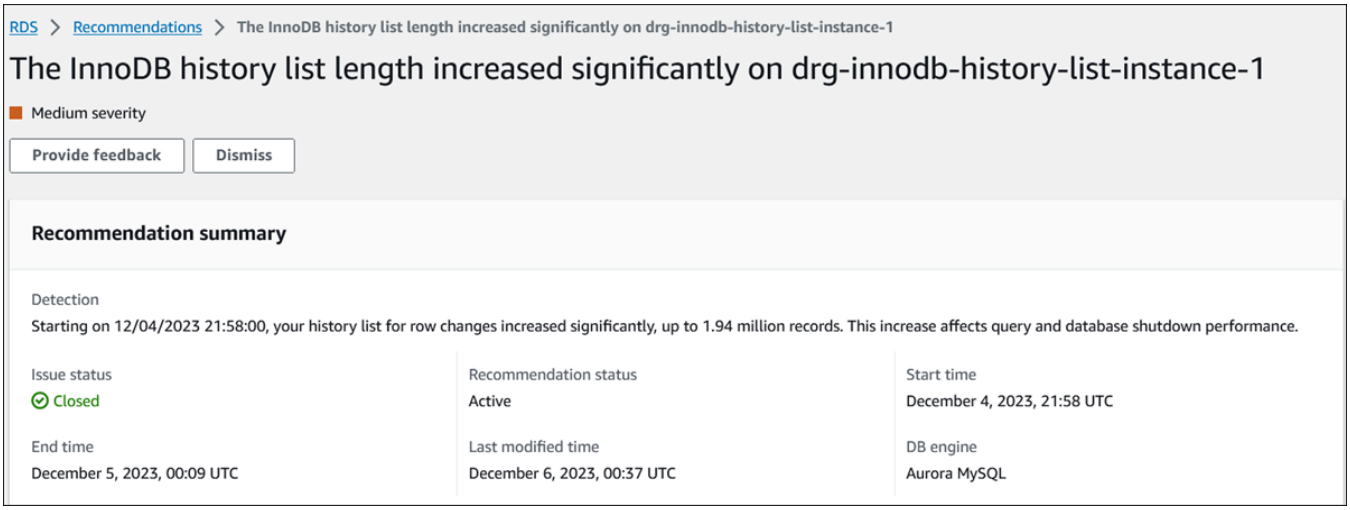
     La page Recommandations affiche une liste de recommandations triées par gravité pour toutes les ressources de votre compte.
   - Dans la page des bases de données, choisissez Bases de données, puis Recommandations pour une ressource.

     L'onglet Recommandations affiche les recommandations et leurs détails pour la ressource sélectionnée.
3. Trouvez une recommandation proactive et choisissez Afficher les détails.

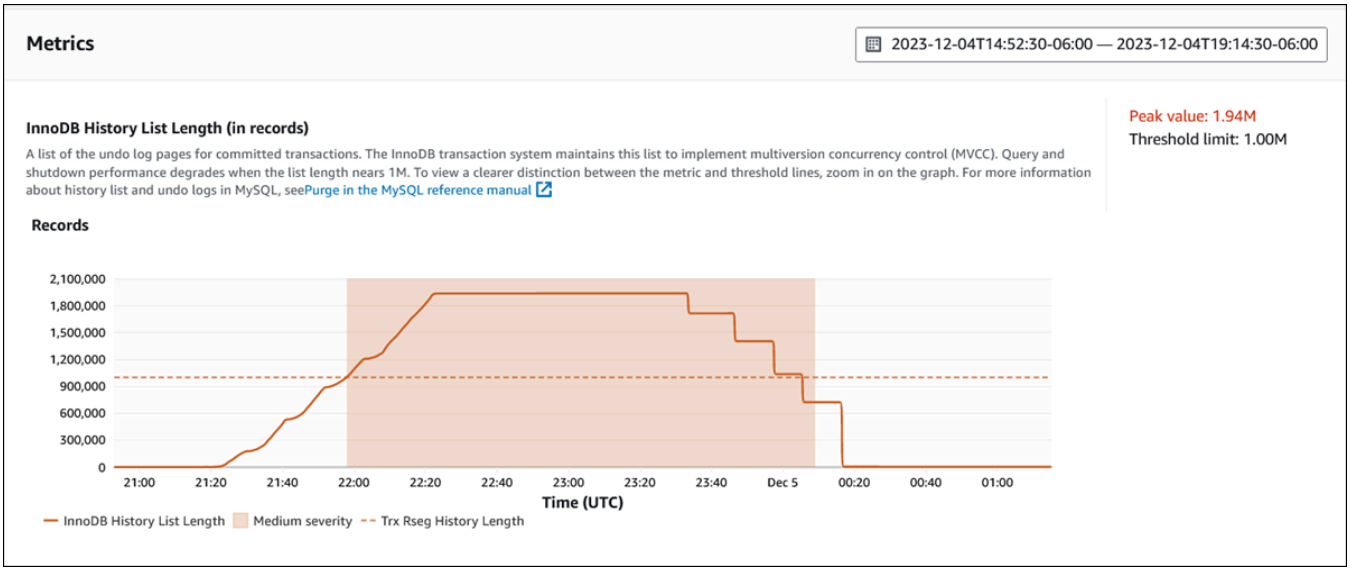
   La page des détails de la recommandation s'affiche. Le titre indique le nom de la ressource concernée ainsi que le problème détecté et sa gravité.

   Les composants de la page de détails des recommandations sont les suivants :

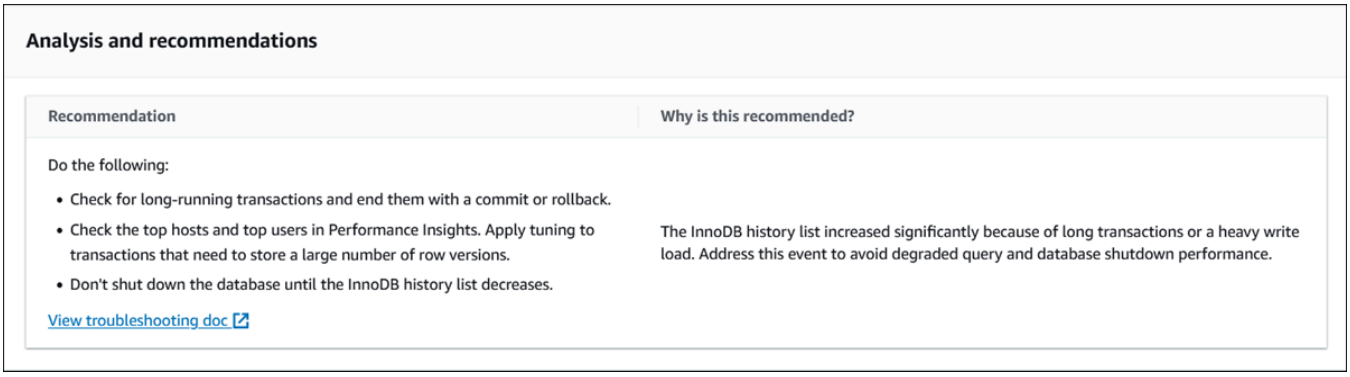
   - Résumé des recommandations : problème détecté, statut de la recommandation et du problème, heure de début et de fin du problème, heure de modification de la recommandation et type de moteur.



- Métriques : graphiques du problème détecté. Chaque graphique affiche un seuil déterminé par le comportement de base de la ressource et les données de la métrique rapportées depuis le début du problème.



- Analyse et recommandations : recommandation et motif de la recommandation suggérée.



Vous pouvez examiner la cause du problème, puis exécuter les actions recommandées pour résoudre le problème, ou choisir Ignorer dans le coin supérieur droit pour ignorer la recommandation.

Récupération de métriques avec l'API Performance Insights pour Aurora

Lorsque l'analyse des performances est activée, l'API fournit une visibilité sur les performances des instances. Amazon CloudWatch Logs fournit la source faisant autorité pour les métriques de surveillance vendues pour les services AWS.

Performance Insights offre une vue spécifique au domaine de la charge de base de données mesurée en tant que moyenne des sessions actives (AAS). Cette métrique est présentée aux consommateurs de l'API sous la forme d'un ensemble de données de série chronologique bidimensionnel. La dimension temporelle des données fournit les données de charge de la base de données pour chaque point temporel de la plage de temps interrogée. Chaque point dans te temps décompose la charge globale par rapport aux dimensions demandées, par exemple, `SQL`, `Wait-event`, `User` ou `Host`, mesurée à ce point dans le temps.

Amazon RDS Performance Insights surveille votre cluster Amazon Aurora pour vous permettre d'analyser les performances de votre base de données et de résoudre les problèmes associés. Vous pouvez consulter les données de Performance Insights dans AWS Management Console. Performance Insights fournit également une API publique qui vous permet d'interroger vos propres données. Vous pouvez utiliser l'API pour effectuer les opérations suivantes :

- Déchargement des données dans une base de données
- Ajout de données Performance Insights aux tableaux de bord de surveillance existants
- Création d'outils de surveillance

Pour utiliser l'API Performance Insights, activez Performance Insights sur l'une de vos instances de base de données Amazon RDS. Pour plus d'informations sur l'activation de Performance Insights, consultez Activation ou désactivation de l'Analyse des performances pour Aurora. Pour plus d'informations sur l'API Performance Insights, consultez la Référence d'API Amazon RDS Performance Insights.

L'API Performance Insights fournit les opérations suivantes.

Action Performance Insights	AWS CLI commande	Description
CreatePerformanceAnalysisReport	aws pi create-performance-analysis-report	Crée un rapport d'analyse des performances pour une période spécifique pour l'instance de base de données. Le résultat est `AnalysisReportId` qui est l'identifiant unique du rapport.
DeletePerformanceAnalysisReport	aws pi delete-performance-analysis-report	Supprime un rapport d'analyse des performances.
DescribeDimensionKeys	aws pi describe-dimension-keys	Récupère les N premières clés de dimension d'une mesure sur une période spécifique.
GetDimensionKeyDetails	aws pi get-dimension-key-details	Récupère les attributs du groupe de dimensions spécifié pour une instance de base de données ou une source de données. Par exemple, si vous spécifiez un ID SQL et si les détails de la dimension sont disponibles, `GetDimensionKeyDetails` récupère le texte intégral de la dimension `db.sql.statement` associée à cet ID. Cette opération est utile, car `GetResourceMetrics` et `DescribeDimensionKeys` ne prennent pas en charge la récupération de

Action Performance Insights	AWS CLI commande	Description
		texte d'instruction SQL volumineux.
GetPerformanceAnalysisReport	aws pi get-performance-analysis-report	Récupère le rapport, y compris les informations du rapport. Le résultat inclut l'état du rapport, l'ID du rapport, les détails temporels du rapport, les informations et les recommandations.
GetResourceMetadata	aws pi get-resource-metadata	Récupérez les métadonnées de différentes fonctions. Par exemple, les métadonnées peuvent indiquer qu'une fonction est activée ou désactivée sur une instance de base de données spécifique.
GetResourceMetrics	aws pi get-resource-metrics	Récupère les métriques Performance Insights d'un ensemble de sources de données, au cours d'une période. Vous pouvez fournir des groupes de dimensions et des dimensions spécifiques, ainsi que des critères d'agrégation et de filtrage, pour chaque groupe.
ListAvailableResourceDimensions	aws pi list-available-resource-dimensions	Récupérez les dimensions pouvant être interrogées pour chaque type de métrique spécifié sur une instance spécifiée.

Action Performance Insights	AWS CLI commande	Description
ListAvailableResourceMetrics	aws pi list-available-resource-metrics	Récupérez toutes les métriques disponibles des types de métriques spécifiés pouvant être interrogés pour une instance de base de données spécifiée.
ListPerformanceAnalysisReports	aws pi list-performance-analysis-reports	Récupère tous les rapports d'analyse disponibles pour l'instance de base de données. Les rapports sont répertoriés en fonction de l'heure de début de chaque rapport.
ListTagsForResource	aws pi list-tags-for-resource	Répertorie toutes les balises de métadonnées ajoutées à la ressource. La liste inclut le nom et la valeur de la balise.
TagResource	aws pi tag-resource	Ajoute des balises de métadonnées à la ressource Amazon RDS. La balise inclut un nom et une valeur.
UntagResource	aws pi untag-resource	Supprime la balise de métadonnées de la ressource.

Pour plus d'informations sur la récupération des métriques de séries chronologiques et des exemples d'AWS CLI pour Performance Insights, consultez les rubriques ci-dessous.

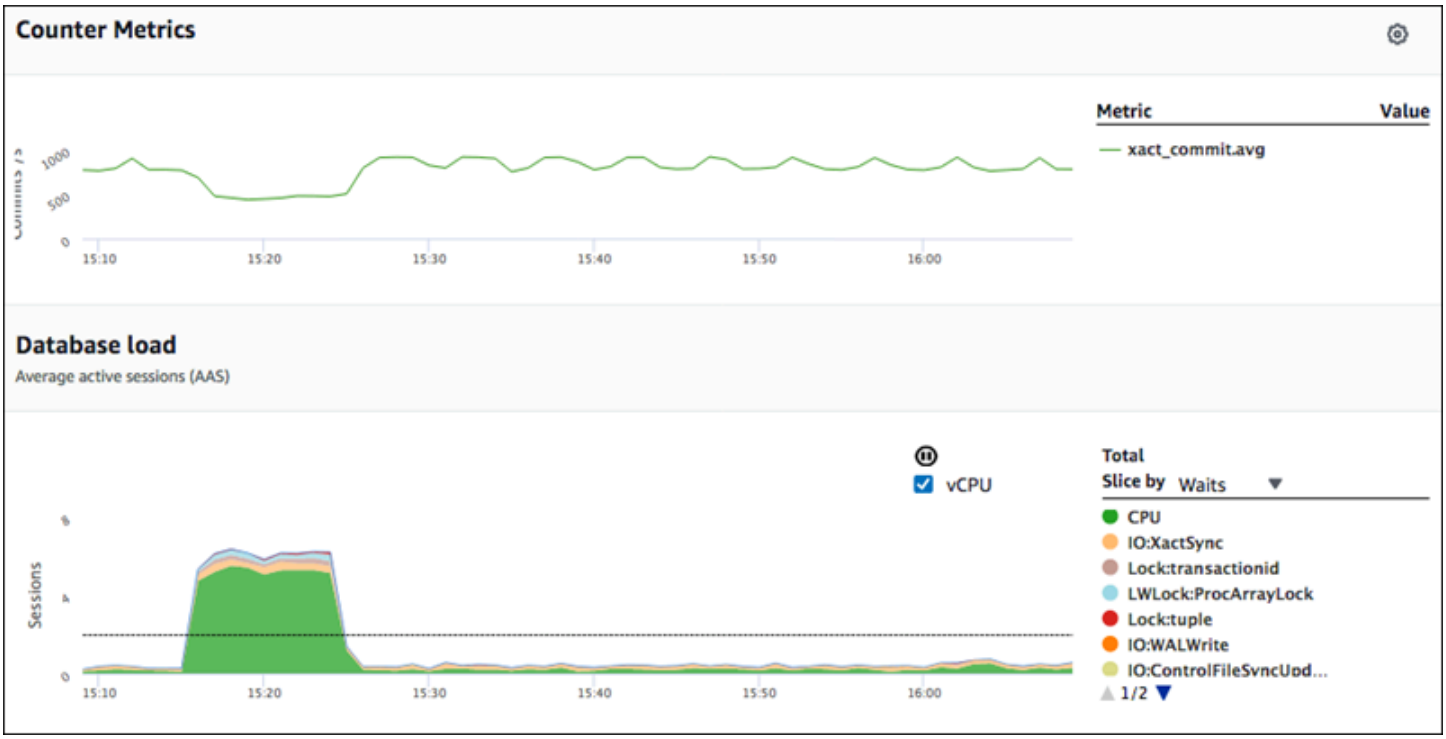
Rubriques

- Récupération de métriques de séries chronologiques pour Performance Insights
- Exemples d'AWS CLI pour Performance Insights

Récupération de métriques de séries chronologiques pour Performance Insights

L'opération `GetResourceMetrics` récupère une ou plusieurs métriques de série chronologique à partir des données de Performance Insights. `GetResourceMetrics` exige une métrique et une période, et renvoie une réponse contenant la liste des points de données.

Par exemple, AWS Management Console utilise `GetResourceMetrics` pour renseigner le graphique Counter Metrics (Métriques de compteur) et le graphique Database Load (Charge de base de données), comme illustré dans l'image ci-dessous.



Toutes les métriques renvoyées par `GetResourceMetrics` sont des métriques de série chronologique standard, à l'exception de `db.load`. Elle apparaît dans le graphique Database Load (Charge de base de données). La métrique `db.load` est différente des autres métriques de séries chronologiques, car vous pouvez la décomposer en sous-composants appelés dimensions. Dans l'image précédente, `db.load` est décomposé et regroupé en fonction des états d'attente qui constituent `db.load`.

> **Note**
>
> `GetResourceMetrics` peut également renvoyer la métrique `db.sampleload`, mais la métrique `db.load` est appropriée dans la plupart des cas.

Pour plus d'informations sur les métriques de compteur renvoyées par `GetResourceMetrics`, consultez Métrique de compteur de Performance Insights.

Les calculs suivants sont pris en charge pour les métriques :

- Moyenne – Moyenne de la métrique sur une période. Ajoutez `.avg` au nom de la métrique.
- Minimum – Valeur minimale de la métrique sur une période. Ajoutez `.min` au nom de la métrique.
- Maximum – Valeur maximale de la métrique sur une période. Ajoutez `.max` au nom de la métrique.
- Somme – Somme des valeurs de la métrique sur une période. Ajoutez `.sum` au nom de la métrique.
- Nombre échantillon – Nombre de fois où la métrique a été collectée sur une période. Ajoutez `.sample_count` au nom de la métrique.

Par exemple, supposons qu'une métrique soit collectée pendant 300 secondes (5 minutes) et qu'elle soit collectée une fois toutes les minutes. Les valeurs pour chaque minute sont 1, 2, 3, 4 et 5. Dans ce cas, les calculs suivants sont renvoyés :

- Moyenne – 3
- Minimum – 1
- Maximum – 5
- Somme – 15
- Nombre échantillon – 5

Pour plus d'informations sur l'utilisation de la commande AWS CLI `get-resource-metrics`, consultez `get-resource-metrics`.

Pour l'option `--metric-queries`, spécifiez une ou plusieurs requêtes pour lesquelles vous souhaitez obtenir les résultats. Chaque requête se compose d'un paramètre `Metric` obligatoire et des paramètres `GroupBy` et `Filter` facultatifs. Voici un exemple de spécification de l'option `--metric-queries`.

```
{
   "Metric": "string",
   "GroupBy": {
     "Group": "string",
     "Dimensions": ["string", ...],
     "Limit": integer
```

```
  },
  "Filter": {"string": "string"
    ...}
```

Exemples d'AWS CLI pour Performance Insights

Dans les sections suivantes, découvrez l'AWS Command Line Interface (AWS CLI) pour Performance Insights et des exemples d'utilisation de l'AWS CLI.

Rubriques

- Aide intégrée de l'AWS CLI pour Performance Insights
- Récupération de métriques de compteur
- Récupération de la charge de base de données moyenne pour les principaux événements d'attente
- Récupération de la charge de base de données moyenne pour les principales instructions SQL
- Récupération de la charge de base de données moyenne filtrée par instruction SQL
- Récupération du texte complet d'une instruction SQL
- Création d'un rapport d'analyse des performances pour une période donnée
- Récupération d'un rapport d'analyse des performances
- Établissement de la liste de tous les rapports d'analyse des performances pour l'instance de base de données
- Suppression d'un rapport d'analyse des performances
- Ajout d'une balise à un rapport d'analyse des performances
- Établissement de la liste de toutes les balises pour un rapport d'analyse des performances
- Suppression des balises d'un rapport d'analyse des performances

Aide intégrée de l'AWS CLI pour Performance Insights

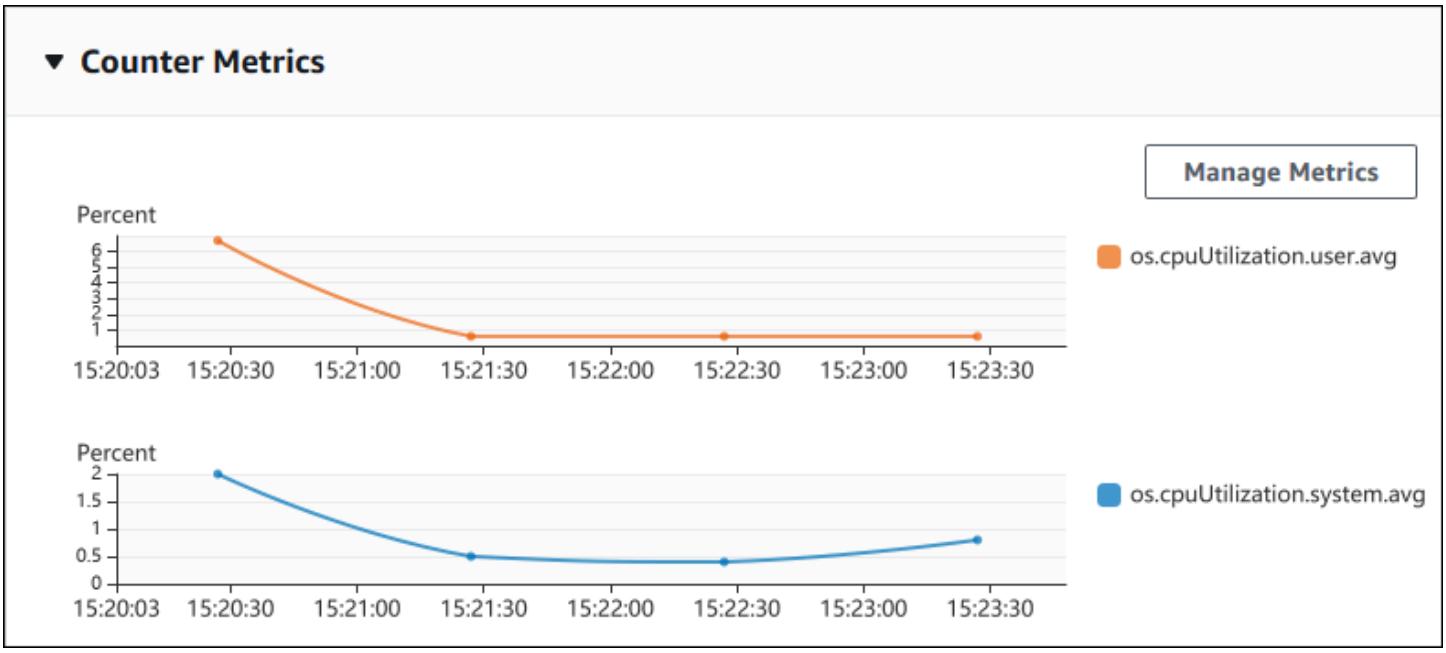
Vous pouvez consulter les données de Performance Insights à l'aide d AWS CLI. Vous pouvez obtenir de l'aide sur les commandes AWS CLI relatives à Performance Insights en saisissant le code suivant sur la ligne de commande.

```
aws pi help
```

Si l'AWS CLI n'est pas installée, consultez Installation d'AWS CLI dans le Guide de l'utilisateur d'AWS CLI pour en savoir plus sur son installation.

Récupération de métriques de compteur

L'image suivante illustre deux graphiques de métriques de compteur dans AWS Management Console.



L'exemple suivant décrit comment collecter les mêmes données utilisées par AWS Management Console pour générer les deux graphiques Counter Metrics (Métriques de compteur).

Pour Linux, macOS ou Unix :

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Pour Windows :

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
```

```
   --period-in-seconds 60 ^
   --metric-queries '[{"Metric": "os.cpuUtilization.user.avg"  },
                      {"Metric": "os.cpuUtilization.idle.avg"}]'
```

Vous pouvez également simplifier la lecture d'une commande en spécifiant un fichier pour l'option `--metrics-query`. L'exemple suivant utilise un fichier nommé query.json pour l'option. Le contenu du fichier est le suivant.

```
[
    {
        "Metric": "os.cpuUtilization.user.avg"
    },
    {
        "Metric": "os.cpuUtilization.idle.avg"
    }
]
```

Exécutez la commande suivante pour utiliser le fichier.

Pour Linux, macOS ou Unix :

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Pour Windows :

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

L'exemple précédent spécifie les valeurs suivantes pour les options :

- `--service-type` – `RDS` pour Amazon RDS
- `--identifier` – ID de ressource de l'instance de base de données
- `--start-time` et `--end-time` – Valeurs `DateTime` conformes à l'ISO 8601 pour la période à interroger, avec plusieurs formats pris en charge

L'interrogation se déroule pendant un intervalle d'une heure :

- `--period-in-seconds` – `60` pour une requête toutes les minutes
- `--metric-queries` – Tableau de deux requêtes s'appliquant chacune à une métrique.

  Le nom de la métrique utilise des points pour classifier la métrique dans une catégorie utile, l'élément final étant une fonction. Dans l'exemple, la fonction est `avg` pour chaque requête. Comme pour Amazon CloudWatch, les fonctions prises en charge sont `min`, `max`, `total` et `avg`.

La réponse ressemble à ce qui suit.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                "Metric": "os.cpuUtilization.user.avg" //Metric1
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 4.0
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 10.0
                }
```

```
                //... 60 datapoints for the os.cpuUtilization.user.avg metric
            ]
        },
        {
            "Key": {
                "Metric": "os.cpuUtilization.idle.avg" //Metric2
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 12.0
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 13.5
                },
                //... 60 datapoints for the os.cpuUtilization.idle.avg metric
            ]
        }
    ] //end of MetricList
} //end of response
```

La réponse contient les éléments `Identifier`, `AlignedStartTime` et `AlignedEndTime`. Étant donné que la valeur de `--period-in-seconds` était définie sur `60`, les heures de début et de fin ont été arrondies à la minute près. Si `--period-in-seconds` était défini sur `3600`, les heures de début et de fin auraient été arrondies à l'heure près.

L'élément `MetricList` dans la réponse comporte un certain nombre d'entrées, chacune associée à une entrée `Key` et `DataPoints`. Chaque élément `DataPoint` comporte une entrée `Timestamp` et `Value`. Chaque liste `Datapoints` répertorie 60 points de données, car les requêtes sont exécutées toutes les minutes pendant une heure, avec `Timestamp1/Minute1`, `Timestamp2/Minute2`, etc. jusqu'à `Timestamp60/Minute60`.

Étant donné que la requête s'applique à deux métriques de compteur différentes, contient deux élément `MetricList`.

Récupération de la charge de base de données moyenne pour les principaux événements d'attente

L'exemple suivant illustre la même requête utilisée par AWS Management Console pour générer un graphique en aires empilées. Il récupère la valeur de `db.load.avg` sur la dernière heure en divisant la charge conformément aux sept principaux événements d'attente. La commande est identique à la

commande de la rubrique Récupération de métriques de compteur. Le contenu du fichier query.json est cependant différent :

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 7 }
    }
]
```

Exécutez la commande suivante.

Pour Linux, macOS ou Unix :

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Pour Windows :

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

L'exemple spécifie la métrique de `db.load.avg` et exécute une action `GroupBy` pour les sept principaux événements d'attente. Pour plus de détails sur les valeurs valides pour cet exemple, consultez DimensionGroup dans la Référence d'API Performance Insights.

La réponse ressemble à ce qui suit.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1540857600.0,
    "AlignedEndTime": 1540861200.0,
```

```
    "MetricList": [
        { //A list of key/datapoints
            "Key": {
                //A Metric with no dimensions. This is the total db.load.avg
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                //Each list of datapoints has the same timestamps and same number of
 items
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.5166666666666667
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.38333333333333336
                },
                {
                    "Timestamp": 1540857780.0, //Minute 3
                    "Value": 0.26666666666666666
                }
                //... 60 datapoints for the total db.load.avg key
            ]
        },
        {
            "Key": {
                //Another key. This is db.load.avg broken down by CPU
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.name": "CPU",
                    "db.wait_event.type": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1540857660.0, //Minute1
                    "Value": 0.35
                },
                {
                    "Timestamp": 1540857720.0, //Minute2
                    "Value": 0.15
                },
                //... 60 datapoints for the CPU key
            ]
```

```
        },
        //... In total we have 8 key/datapoints entries, 1) total, 2-8) Top Wait Events
    ] //end of MetricList
} //end of response
```

Dans cette réponse, comporte huit entrée `MetricList`. Une entrée s'applique à la valeur totale de `db.load.avg` et les sept autres entrées s'appliquent à chacune des valeurs de `db.load.avg` divisées conformément à l'un des sept principaux événements d'attente. Contrairement au premier exemple qui comportait une dimension de regroupement, cet exemple doit définir un élément Key pour chaque regroupement de la métrique. Un seul élément Key peut être associé à chaque métrique, comme dans le cas d'utilisation de la métrique de compteur de base.

Récupération de la charge de base de données moyenne pour les principales instructions SQL

L'exemple suivant regroupe `db.wait_events` par les 10 principales instructions SQL. Il existe deux groupes différents pour les instructions SQL :

- `db.sql` – Instruction SQL complète, telle que `select * from customers where customer_id = 123`
- `db.sql_tokenized` – Instruction SQL tokenisée, telle que `select * from customers where customer_id = ?`

Lors de l'analyse des performances de base de données, il peut s'avérer utile de considérer les instructions SQL dont les paramètres sont différents comme un seul élément logique. Vous pouvez donc utiliser `db.sql_tokenized` lors de l'interrogation. Toutefois, en particulier si vous êtes intéressé par les plans d'explication, il est parfois plus utile d'examiner les instructions SQL complètes avec leurs paramètres, et le regroupement des requêtes par `db.sql`. Il existe une relation parent-enfant entre une instruction SQL tokenisée et une instruction SQL complète, où plusieurs instructions SQL complètes (enfants) sont regroupées sous la même instruction SQL tokenisée (parent).

La commande illustrée dans cet exemple est identique à la commande de la rubrique Récupération de la charge de base de données moyenne pour les principaux événements d'attente. Le contenu du fichier query.json est cependant différent :

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.sql_tokenized", "Limit": 10 }
```

```
    }
]
```

L'exemple suivant utilise `db.sql_tokenized`.

Pour Linux, macOS ou Unix :

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-29T00:00:00Z \
   --end-time   2018-10-30T00:00:00Z \
   --period-in-seconds 3600 \
   --metric-queries file://query.json
```

Pour Windows :

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-29T00:00:00Z ^
   --end-time   2018-10-30T00:00:00Z  ^
   --period-in-seconds 3600 ^
   --metric-queries file://query.json
```

Cet exemple procède à une interrogation pendant 24 heures, avec une période en secondes (period-in-seconds) d'une heure.

L'exemple spécifie la métrique de `db.load.avg` et exécute une action `GroupBy` pour les sept principaux événements d'attente. Pour plus de détails sur les valeurs valides pour cet exemple, consultez DimensionGroup dans la Référence d'API Performance Insights.

La réponse ressemble à ce qui suit.

```
{
    "AlignedStartTime": 1540771200.0,
    "AlignedEndTime": 1540857600.0,
    "Identifier": "db-XXX",

    "MetricList": [ //11 entries in the MetricList
        {
            "Key": { //First key is total
```

```
                "Metric": "db.load.avg"
            }
            "DataPoints": [ //Each DataPoints list has 24 per-hour Timestamps and a
 value
                {
                    "Value": 1.6964980544747081,
                    "Timestamp": 1540774800.0
                },
                //... 24 datapoints
            ]
        },
        {
            "Key": { //Next key is the top tokenized SQL
                "Dimensions": {
                    "db.sql_tokenized.statement": "INSERT INTO authors (id,name,email)
 VALUES\n( nextval(?)  ,?,?)",
                    "db.sql_tokenized.db_id": "pi-2372568224",
                    "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE"
                },
                "Metric": "db.load.avg"
            },
            "DataPoints": [ //... 24 datapoints
            ]
        },
        // In total 11 entries, 10 Keys of top tokenized SQL, 1 total key
    ] //End of MetricList
} //End of response
```

Cette réponse comporte 11 entrées dans `MetricList` (1 correspondant au total, les 10 autres correspondant aux principales instructions SQL tokenisées), chaque entrée étant associée à 24 `DataPoints` par heure.

Pour les instructions SQL tokenisées, chaque liste de dimensions répertorie trois entrées :

- `db.sql_tokenized.statement` – Instruction SQL tokenisée.
- `db.sql_tokenized.db_id` – ID de base de données native utilisé pour faire référence à l'instruction SQL, ou ID synthétique généré par Performance Insights si l'ID de base de données native n'est pas disponible. Cet exemple renvoie l'ID synthétique `pi-2372568224`.
- `db.sql_tokenized.id` – ID de la requête dans Performance Insights.

  Dans AWS Management Console, cet ID se nomme ID de support. Il porte ce nom, car l'ID représente les données qu'AWS Support peut examiner pour vous aider à résoudre un problème

lié à votre base de données. AWS prend la sécurité et la confidentialité de vos données très au sérieux, et presque toutes les données sont stockées en mode chiffré avec votre clé AWS KMS. Personne au sein d'AWS ne peut ainsi consulter ces données. Dans l'exemple précédent, `tokenized.statement` et `tokenized.db_id` sont tous les deux stockés sous forme chiffrée. Si vous rencontrez un problème avec votre base de données, AWS Support peut vous aider en fournissant l'ID de support.

Lors de l'interrogation, il peut s'avérer utile de spécifier une entrée `Group` dans `GroupBy`. Toutefois, pour contrôler les données renvoyées de manière plus précise, spécifier la liste des dimensions. Par exemple, si `db.sql_tokenized.statement` est le seul élément nécessaire, un attribut `Dimensions` peut être ajouté au fichier query.json.

```
[
    {
        "Metric": "db.load.avg",
        "GroupBy": {
            "Group": "db.sql_tokenized",
            "Dimensions":["db.sql_tokenized.statement"],
            "Limit": 10
        }
    }
]
```

Récupération de la charge de base de données moyenne filtrée par instruction SQL

L'image précédente indique qu'une requête particulière est sélectionnée et que le graphique en aires empilées Average active sessions (Sessions actives en moyenne) qui apparaît dans la section supérieure s'y applique. Bien que la requête concerne toujours les sept principaux événements d'attente globaux, la valeur de la réponse est filtrée. Le filtre permet à la requête de prendre uniquement en compte les sessions qui correspondent à un filtre en particulier.

La requête d'API correspondante illustrée dans cet exemple est identique à la commande de la rubrique Récupération de la charge de base de données moyenne pour les principales instructions SQL. Le contenu du fichier query.json est cependant différent :

```
[
 {
        "Metric": "db.load.avg",
        "GroupBy": { "Group": "db.wait_event", "Limit": 5  },
        "Filter": { "db.sql_tokenized.id": "AKIAIOSFODNN7EXAMPLE" }
    }
]
```

Pour Linux, macOS ou Unix :

```
aws pi get-resource-metrics \
   --service-type RDS \
   --identifier db-ID \
   --start-time 2018-10-30T00:00:00Z \
   --end-time   2018-10-30T01:00:00Z \
   --period-in-seconds 60 \
   --metric-queries file://query.json
```

Pour Windows :

```
aws pi get-resource-metrics ^
   --service-type RDS ^
   --identifier db-ID ^
   --start-time 2018-10-30T00:00:00Z ^
   --end-time   2018-10-30T01:00:00Z ^
   --period-in-seconds 60 ^
   --metric-queries file://query.json
```

La réponse ressemble à ce qui suit.

```
{
    "Identifier": "db-XXX",
    "AlignedStartTime": 1556215200.0,
    "MetricList": [
        {
            "Key": {
                "Metric": "db.load.avg"
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.4878117913832196
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.192823803967328
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
```

```
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/aurora_redo_log_flush"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 1.1360544217687074
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 1.058051341890315
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "io",
                    "db.wait_event.name": "wait/io/table/sql/handler"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.16241496598639457
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05163360560093349
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
                    "db.wait_event.name": "wait/synch/mutex/innodb/
aurora_lock_thread_slot_futex"
                }
            },
```

```
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.11479591836734694
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.013127187864644107
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "CPU",
                    "db.wait_event.name": "CPU"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.05215419501133787
                },
                {
                    "Timestamp": 1556222400.0,
                    "Value": 0.05805134189031505
                }
            ]
        },
        {
            "Key": {
                "Metric": "db.load.avg",
                "Dimensions": {
                    "db.wait_event.type": "synch",
                    "db.wait_event.name": "wait/synch/mutex/innodb/lock_wait_mutex"
                }
            },
            "DataPoints": [
                {
                    "Timestamp": 1556218800.0,
                    "Value": 0.017573696145124718
                },
                {
```

```
                    "Timestamp": 1556222400.0,
                    "Value": 0.002333722287047841
                }
            ]
        }
    ],
    "AlignedEndTime": 1556222400.0
} //end of response
```

Dans cette réponse, toutes les valeurs sont filtrées selon la contribution de l'instruction SQL tokenisée AKIAIOSFODNN7EXAMPLE spécifiée dans le fichier query.json. Les éléments Key peuvent également suivre un ordre différent d'une requête sans filtre, car ils correspondent aux cinq principaux événements d'attente qui ont affecté l'instruction SQL filtrée.

Récupération du texte complet d'une instruction SQL

L'exemple suivant montre comment récupérer le texte intégral d'une instruction SQL pour une instance de base de données `db-10BCD2EFGHIJ3KL4M5NO6PQRS5`. Le `--group` est `db.sql` et l'`--group-identifier` est `db.sql.id`. Dans cet exemple, *`my-sql-id`* représente un ID SQL récupéré en invoquant `pi get-resource-metrics` ou `pi describe-dimension-keys`.

Exécutez la commande suivante.

Pour Linux, macOS ou Unix :

```
aws pi get-dimension-key-details \
   --service-type RDS \
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 \
   --group db.sql \
   --group-identifier my-sql-id \
   --requested-dimensions statement
```

Pour Windows :

```
aws pi get-dimension-key-details ^
   --service-type RDS ^
   --identifier db-10BCD2EFGHIJ3KL4M5NO6PQRS5 ^
   --group db.sql ^
   --group-identifier my-sql-id ^
   --requested-dimensions statement
```

Dans cet exemple, les détails des dimensions sont disponibles. Ainsi, Performance Insights récupère le texte intégral de l'instruction SQL, sans le tronquer.

```
{
    "Dimensions":[
    {
        "Value": "SELECT e.last_name, d.department_name FROM employees e, departments d
 WHERE e.department_id=d.department_id",
        "Dimension": "db.sql.statement",
        "Status": "AVAILABLE"
    },
    ...
    ]
}
```

Création d'un rapport d'analyse des performances pour une période donnée

L'exemple suivant crée un rapport d'analyse des performances avec l'heure de début `1682969503` et l'heure de fin `1682979503` pour la base de données `db-loadtest-0`.

```
aws pi create-performance-analysis-report \
        --service-type RDS \
        --identifier db-loadtest-0 \
        --start-time 1682969503 \
        --end-time 1682979503 \
        --region us-west-2
```

La réponse est l'identifiant unique `report-0234d3ed98e28fb17` du rapport.

```
{
   "AnalysisReportId": "report-0234d3ed98e28fb17"
}
```

Récupération d'un rapport d'analyse des performances

L'exemple suivant extrait les détails du rapport d'analyse pour le rapport `report-0d99cc91c4422ee61`.

```
aws pi get-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
```

```
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

La réponse fournit l'état du rapport, son identifiant, les détails temporels et des informations.

```
        {
    "AnalysisReport": {
        "Status": "Succeeded",
        "ServiceType": "RDS",
        "Identifier": "db-loadtest-0",
        "StartTime": 1680583486.584,
        "AnalysisReportId": "report-0d99cc91c4422ee61",
        "EndTime": 1680587086.584,
        "CreateTime": 1680587087.139,
        "Insights": [
           ... (Condensed for space)
        ]
    }
}
```

Établissement de la liste de tous les rapports d'analyse des performances pour l'instance de base de données

L'exemple suivant répertorie tous les rapports d'analyse des performances disponibles pour la base de données `db-loadtest-0`.

```
aws pi list-performance-analysis-reports \
--service-type RDS \
--identifier db-loadtest-0 \
--region us-west-2
```

La réponse répertorie tous les rapports avec l'ID du rapport, le statut et les détails temporels de la période.

```
{
    "AnalysisReports": [
        {
            "Status": "Succeeded",
            "EndTime": 1680587086.584,
            "CreationTime": 1680587087.139,
```

```
            "StartTime": 1680583486.584,
            "AnalysisReportId": "report-0d99cc91c4422ee61"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681491137.914,
            "CreationTime": 1681491145.973,
            "StartTime": 1681487537.914,
            "AnalysisReportId": "report-002633115cc002233"
        },
        {
            "Status": "Succeeded",
            "EndTime": 1681493499.849,
            "CreationTime": 1681493507.762,
            "StartTime": 1681489899.849,
            "AnalysisReportId": "report-043b1e006b47246f9"
        },
        {
            "Status": "InProgress",
            "EndTime": 1682979503.0,
            "CreationTime": 1682979618.994,
            "StartTime": 1682969503.0,
            "AnalysisReportId": "report-01ad15f9b88bcbd56"
        }
    ]
}
```

Suppression d'un rapport d'analyse des performances

L'exemple suivant supprime le rapport d'analyse pour la base de données `db-loadtest-0`.

```
aws pi delete-performance-analysis-report \
--service-type RDS \
--identifier db-loadtest-0 \
--analysis-report-id report-0d99cc91c4422ee61 \
--region us-west-2
```

Ajout d'une balise à un rapport d'analyse des performances

L'exemple suivant ajoute une balise avec une clé `name` et une valeur `test-tag` au rapport `report-01ad15f9b88bcbd56`.

```
aws pi tag-resource \
```

```
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tags Key=name,Value=test-tag \
--region us-west-2
```

Établissement de la liste de toutes les balises pour un rapport d'analyse des performances

L'exemple suivant répertorie toutes les balises pour le rapport `report-01ad15f9b88bcbd56`.

```
aws pi list-tags-for-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--region us-west-2
```

La réponse répertorie la valeur et la clé de toutes les balises ajoutées au rapport :

```
{
    "Tags": [
        {
            "Value": "test-tag",
            "Key": "name"
        }
    ]
}
```

Suppression des balises d'un rapport d'analyse des performances

L'exemple suivant montre comment supprimer la balise `name` du rapport `report-01ad15f9b88bcbd56`.

```
aws pi untag-resource \
--service-type RDS \
--resource-arn arn:aws:pi:us-west-2:356798100956:perf-reports/RDS/db-loadtest-0/
report-01ad15f9b88bcbd56 \
--tag-keys name \
--region us-west-2
```

Une fois la balise supprimée, l'appel de l'API `list-tags-for-resource` ne répertorie pas cette balise.

Journalisation des appels Performance Insights avec AWS CloudTrail

Performance Insights s'exécute avec AWS CloudTrail, un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un service AWS dans Performance Insights. CloudTrail capture tous les appels d'API pour Performance Insights en tant qu'événements. Cette capture inclut les appels de la console Amazon RDS et les appels de code aux opérations de l'API Performance Insights.

Si vous créez un journal de suivi, vous pouvez activer la diffusion en continu des événements CloudTrail dans un compartiment Amazon S3, y compris les événements concernant Performance Insights. Si vous ne configurez pas de journal de suivi, vous pouvez toujours afficher les événements les plus récents dans la console CloudTrail dans Event history (Historique des événements). Grâce aux données collectées par CloudTrail, vous pouvez déterminer certaines informations. Ces informations incluent la demande qui a été envoyée à Performance Insights, l'adresse IP à partir de laquelle la demande a été effectuée, l'auteur de la demande et sa date. Elles comprennent également des détails supplémentaires.

Pour en savoir plus sur CloudTrail, consultez le Guide de l'utilisateur AWS CloudTrail.

Utilisation des informations Performance Insights dans CloudTrail

CloudTrail est activé dans votre compte AWS lors de la création de ce dernier. Quand une activité se produit dans Performance Insights, cette activité est enregistrée dans un événement CloudTrail avec d'autres événements de service AWS dans la console CloudTrail, dans Event history (Historique des événements). Vous pouvez afficher, rechercher et télécharger les événements récents dans votre compte AWS. Pour plus d'informations, consultez Affichage des événements avec l'historique des événements CloudTrail dans le Guide de l'utilisateur AWS CloudTrail.

Pour un enregistrement continu des événements dans votre compte AWS, y compris des événements concernant Performance Insights, créez un journal de suivi. Un journal de suivi permet à CloudTrail de livrer des fichiers journaux dans un compartiment Amazon S3. Par défaut, lorsque vous créez un journal de suivi dans la console, il s'applique à toutes les régions AWS. Le journal d'activité consigne les événements de toutes les Régions dans la partition AWSAWS et transfère les fichiers journaux dans le compartiment Amazon S3 de votre choix. En outre, vous pouvez configurer d'autres services AWS pour analyser et agir sur les données d'événements collectées dans les journaux CloudTrail. Pour plus d'informations, consultez les rubriques suivantes dans le Guide de l'utilisateur AWS CloudTrail :

- Vue d'ensemble de la création d'un journal d'activité

- Intégrations et services pris en charge par CloudTrail
- Configuration des notifications d'Amazon SNS pour CloudTrail
- Réception des fichiers journaux CloudTrail de plusieurs régions et Réception des fichiers journaux CloudTrail de plusieurs comptes

Toutes les opérations de Performance Insights sont consignées par CloudTrail et documentées dans la Référence d'API Performance Insights. Par exemple, les appels aux opérations `DescribeDimensionKeys` et `GetResourceMetrics` génèrent des entrées dans les fichiers journaux CloudTrail.

Chaque événement ou entrée du journal contient des informations sur la personne qui a généré la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été effectuée avec des informations d'identification d'utilisateur root ou IAM.
- Si la demande a été effectuée avec des informations d'identification de sécurité temporaires pour un rôle ou un utilisateur fédéré.
- Si la demande a été effectuée par un autre service AWS.

Pour plus d'informations, consultez l'élément userIdentity CloudTrail.

Entrées du fichier journal Performance Insights

Un journal de suivi est une configuration qui permet la livraison d'événements sous forme de fichiers journaux vers un compartiment Amazon S3 que vous spécifiez. Les fichiers journaux CloudTrail peuvent contenir une ou plusieurs entrées de journal. Un événement représente une demande individuelle d'une source quelconque. Chaque événement comprend des informations sur l'opération demandée, la date et l'heure de l'opération, les paramètres de la demande, etc. Les fichiers journaux CloudTrail ne constituent pas une série ordonnée retraçant les appels d'API publics. Ils ne suivent aucun ordre précis.

L'exemple suivant montre une entrée de journal CloudTrail qui illustre l'opération `GetResourceMetrics`.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
```

```
         "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T19:28:46Z",
    "eventSource": "pi.amazonaws.com",
    "eventName": "GetResourceMetrics",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.198.67",
    "userAgent": "aws-cli/1.16.240 Python/3.7.4 Darwin/18.7.0 botocore/1.12.230",
    "requestParameters": {
        "identifier": "db-YTDU5J5V66X7CXSCVDFD2V3SZM",
        "metricQueries": [
            {
                "metric": "os.cpuUtilization.user.avg"
            },
            {
                "metric": "os.cpuUtilization.idle.avg"
            }
        ],
        "startTime": "Dec 18, 2019 5:28:46 PM",
        "periodInSeconds": 60,
        "endTime": "Dec 18, 2019 7:28:46 PM",
        "serviceType": "RDS"
    },
    "responseElements": null,
    "requestID": "9ffbe15c-96b5-4fe6-bed9-9fccff1a0525",
    "eventID": "08908de0-2431-4e2e-ba7b-f5424f908433",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


API Performance Insights et points de terminaison de VPC d’interface (AWS PrivateLink)

Vous pouvez utiliser AWS PrivateLink pour créer une connexion privée entre votre VPC et l’Analyse des performances d’Amazon RDS. Vous pouvez accéder à Performance Insights comme si le service se trouvait dans votre VPC, sans passerelle Internet, périphérique NAT, connexion VPN ou connexion Direct Connect. Les instances de votre VPC ne nécessitent pas d’adresses IP publiques pour accéder à Performance Insights.

Vous établissez cette connexion privée en créant un point de terminaison d'interface optimisé par AWS PrivateLink. Nous créons une interface réseau de point de terminaison dans chaque sous-réseau que vous activez pour le point de terminaison d'interface. Il s'agit d'interfaces réseau gérées par le demandeur qui servent de point d'entrée pour le trafic destiné à Performance Insights.

Pour plus d'informations, consultez Accès aux Services AWS via AWS PrivateLink dans le Guide AWS PrivateLink.

Considérations relatives à Performance Insights

Avant de configurer un point de terminaison d'interface pour Performance Insights, consultez Considérations dans le Guide AWS PrivateLink.

Performance Insights prend en charge les appels vers toutes ses actions d'API via le point de terminaison d'interface.

Par défaut, l'accès complet à Performance Insights est autorisé via le point de terminaison d'interface. Pour contrôler le trafic vers Performance Insights via le point de terminaison d'interface, associez un groupe de sécurité aux interfaces réseau de ce point de terminaison.

Disponibilité

L'API Performance Insights prend actuellement en charge les points de terminaison de VPC dans Régions AWS compatibles avec Performance Insights. Pour en savoir plus sur la disponibilité de Performance Insights, consulter Régions et moteurs de base de données Aurora pris en charge pour Performance Insights.

Créez un point de terminaison d'interface pour Performance Insights

Vous pouvez créer un point de terminaison d'interface pour Performance Insights à l'aide de la console Amazon VPC ou de l'AWS Command Line Interface (AWS CLI). Pour plus d'informations, consultez Création d'un point de terminaison d'interface dans le Guide AWS PrivateLink.

Créez un point de terminaison d'interface pour Performance Insights à l'aide du nom de service suivant :

Si vous activez le DNS privé pour le point de terminaison d'interface, vous pouvez adresser des demandes d'API à Performance Insights en utilisant son nom DNS par défaut pour la région. Par exemple, `pi.us-east-1.amazonaws.com`.

Création d’une stratégie de point de terminaison de VPC pour l’API Performance Insights

Une politique de point de terminaison est une ressource IAM que vous pouvez attacher à votre point de terminaison d’interface. La politique de point de terminaison par défaut autorise un accès complet aux API Performance Insights via le point de terminaison d’interface. Pour contrôler l’accès autorisé à Performance Insights depuis votre VPC, attachez une politique de point de terminaison personnalisée au point de terminaison de l’interface.

Une politique de point de terminaison spécifie les informations suivantes :

- Les principaux qui peuvent effectuer des actions (Comptes AWS, utilisateurs IAM et rôles IAM).
- Les actions qui peuvent être effectuées.
- La ressource sur laquelle les actions peuvent être effectuées.

Pour plus d’informations, consultez Contrôle de l’accès aux services à l’aide de politiques de point de terminaison dans le Guide AWS PrivateLink.

Exemple : politique de point de terminaison d’un VPC pour les actions Performance Insights

Voici un exemple de politique de point de terminaison personnalisée. Lorsque vous attachez cette politique à votre point de terminaison d’interface, elle accorde l’accès aux actions Performance Insights répertoriées pour tous les principaux sur toutes les ressources.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreatePerformanceAnalysisReport",
            "rds:DeletePerformanceAnalysisReport",
            "rds:GetPerformanceAnalysisReport"
         ],
         "Resource":"*"
      }
   ]
}
```

Exemple : stratégie de point de terminaison d'un VPC qui refuse tout accès à partir d'un compte AWS spécifié

La stratégie de point de terminaison d'un VPC suivante refuse au compte AWS `123456789012` tout accès aux ressources utilisant le point de terminaison. La politique autorise toutes les actions provenant d'autres comptes.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Adressage IP pour Performance Insights

Les adresses IP permettent aux ressources de votre VPC de communiquer entre elles et avec les ressources sur Internet. Performance Insights prend en charge les protocoles d'adressage IPv4 et IPv6. Par défaut, IP Performance Insights et Amazon VPC utilisent le protocole d'adressage IPv4. Vous ne pouvez pas désactiver ce comportement. Lorsque vous créez un VPC, veillez à spécifier un bloc CIDR IPv4 (une plage d'adresses IPv4 privées).

Vous pouvez également attribuer un bloc CIDR IPv6 à votre VPC et vos sous-réseaux, et attribuer des adresses IPv6 de ce bloc aux ressources RDS de votre sous-réseau. La prise en charge du protocole IPv6 augmente le nombre d'adresses IP prises en charge. En utilisant le protocole IPv6, vous vous assurez d'avoir suffisamment d'adresses disponibles pour la croissance future d'Internet. Les ressources RDS nouvelles et existantes peuvent utiliser des adresses IPv4 et IPv6 dans votre VPC. La configuration, la sécurisation et la traduction du trafic réseau entre les deux protocoles utilisés dans les différentes parties d'une application peuvent entraîner une surcharge opérationnelle. Vous pouvez standardiser le protocole IPv6 pour les ressources Amazon RDS afin de simplifier la

configuration de votre réseau. Pour plus d'informations sur les points de terminaison de service et les quotas, consultez Points de terminaison de service et quotas Amazon Relational Database Service.

Pour plus d'informations sur l'adressage IP Aurora, consultez Adressage IP Aurora.

Analyse des anomalies de performance d'Aurora avec Amazon DevOps Guru pour Amazon RDS

Amazon DevOps Guru est un service d'exploitation entièrement géré qui aide les développeurs et les opérateurs à améliorer les performances et la disponibilité de leurs applications. DevOpsGuru décharge les tâches associées à l'identification des problèmes opérationnels afin que vous puissiez rapidement mettre en œuvre les recommandations visant à améliorer votre application. Pour plus d'informations, consultez Qu'est-ce qu'Amazon DevOps Guru ? dans le guide de l'utilisateur Amazon DevOps Guru.

DevOpsGuru détecte, analyse et formule des recommandations pour les problèmes opérationnels existants pour tous les moteurs de base de données Amazon RDS. DevOpsGuru for RDS étend cette fonctionnalité en appliquant l'apprentissage automatique aux métriques Performance Insights pour les bases de données Amazon Aurora . Ces fonctionnalités de surveillance permettent à DevOps Guru for RDS de détecter et de diagnostiquer les problèmes de performance et de recommander des mesures correctives spécifiques. DevOpsGuru for RDS peut également détecter les situations problématiques dans vos bases de données Aurora (base de données .

Vous pouvez désormais consulter ces recommandations dans la console RDS. Pour de plus amples informations, veuillez consulter Recommandations d'Amazon Aurora.

> **Important**
>
> Amazon DevOps Guru n'est pas disponible pour les clusters de Aurora Serverless bases de données.

La vidéo suivante présente un aperçu de DevOps Guru for RDS.

Pour en savoir plus sur ce sujet, consultez Amazon DevOps Guru for RDS under the hood.

Rubriques

- Avantages de DevOps Guru for RDS
- Comment fonctionne DevOps Guru for RDS
- Configuration de DevOps Guru pour RDS

Avantages de DevOps Guru for RDS

En tant que responsable d’une base de données Amazon Aurora, vous ne savez pas systématiquement lorsqu’un événement ou une régression affectant cette base de données se produit. Lorsque vous prenez connaissance de ce problème, vous ne savez pas toujours pourquoi il se produit ni comment y remédier. Plutôt que de vous adresser à un administrateur de base de données (DBA) pour obtenir de l'aide ou de vous fier à des outils tiers, vous pouvez suivre les recommandations de DevOps Guru for RDS.

L'analyse détaillée de DevOps Guru for RDS vous apporte les avantages suivants :

Diagnostic rapide

DevOpsGuru for RDS surveille et analyse en permanence la télémétrie des bases de données. DevOpsGuru for RDS utilise des techniques statistiques et d'apprentissage automatique pour exploiter ces données et détecter les anomalies. Pour en savoir plus sur les données de télémétrie, consultez Surveillance de la charge de la base de données avec Performance Insights sur Amazon Aurora et Surveillance des métriques du système d’exploitation à l’aide de la Surveillance améliorée dans le Guide de l’utilisateur Amazon Aurora .

Résolution rapide

Chaque anomalie identifie le problème de performances et suggère des pistes d’enquête ou des actions correctives. Par exemple, DevOps Guru for RDS peut vous recommander d'étudier des événements d'attente spécifiques. Il peut également vous recommander de régler les paramètres de votre groupe d’applications afin de limiter le nombre de connexions à la base de données. Sur la base de ces recommandations, vous pouvez résoudre les problèmes de performances plus rapidement qu’en effectuant un dépannage manuel.

Insights proactifs

DevOpsGuru for RDS utilise les indicateurs de vos ressources pour détecter les comportements potentiellement problématiques avant qu'ils ne s'aggravent. Par exemple, il peut détecter lorsque votre base de données utilise un nombre croissant de tables temporaires sur disque, ce qui peut avoir un impact sur les performances. DevOps Guru fournit ensuite des recommandations pour vous aider à résoudre les problèmes avant qu'ils ne s'aggravent.

Connaissance approfondie des ingénieurs Amazon et du machine learning

Pour détecter les problèmes de performance et vous aider à résoudre les goulots d'étranglement, DevOps Guru for RDS s'appuie sur l'apprentissage automatique (ML) et des formules

mathématiques avancées. Les ingénieurs de base de données Amazon ont contribué au développement des résultats de DevOps Guru for RDS, qui résument de nombreuses années de gestion de centaines de milliers de bases de données. En s'appuyant sur ces connaissances collectives, DevOps Guru for RDS peut vous enseigner les meilleures pratiques.

Comment fonctionne DevOps Guru for RDS

DevOpsGuru for RDS collecte des données sur vos bases de données Aurora Insights. La métrique la plus importante est`DBLoad`. DevOpsGuru for RDS utilise les indicateurs Performance Insights, les analyse à l'aide de l'apprentissage automatique et publie des informations sur le tableau de bord.

Un aperçu est un ensemble d'anomalies connexes détectées par DevOps Guru.

Dans DevOps Guru for RDS, une anomalie est un schéma qui s'écarte des performances considérées comme normales pour votre base de données Amazon Aurora PostgreSQL.

Insights proactifs

Un insight proactif vous permet de connaître un comportement problématique avant qu'il se produise. Il contient des anomalies accompagnées de recommandations et de métriques associées pour vous aider à résoudre les problèmes dans vos bases de données Amazon Aurora avant qu'ils ne s'aggravent. Ces informations sont publiées dans le tableau de bord DevOps Guru.

Par exemple, DevOps Guru peut détecter que votre base de données Aurora PostgreSQL crée de nombreuses tables temporaires sur disque. Si elle n'est pas corrigée, cette tendance peut entraîner des problèmes de performances. Chaque insight proactif inclut des recommandations de comportement correctif et des liens vers des rubriques pertinentes dans Réglage d'Aurora MySQL avec les insights proactifs Amazon DevOps Guru ou Optimisation de grâce aux informations proactives d'Amazon Guru DevOps. Pour plus d'informations, consultez la section Travailler avec des informations dans DevOps Guru dans le guide de l'utilisateur Amazon DevOps Guru.

Insights réactifs

Un insight réactif identifie un comportement anormal lorsqu'il se produit. Si DevOps Guru for RDS détecte des problèmes de performances dans vos instances de base de données Amazon Aurora , il publie un aperçu réactif dans le tableau de bord Guru. DevOps Pour plus d'informations, consultez la section Travailler avec des informations dans DevOps Guru dans le guide de l'utilisateur Amazon DevOps Guru.

Anomalies causales

Une anomalie causale est une anomalie de premier niveau au sein d'un insight réactif. Le chargement de la base de données (charge de base de données) est l'anomalie causale de DevOps Guru for RDS.

Une anomalie mesure l'impact sur les performances en attribuant un niveau de gravité High (Élevé), Medium (Moyen) ou Low (Faible). Pour en savoir plus, consultez la section Concepts clés de DevOps Guru for RDS dans le guide de l'utilisateur Amazon DevOps Guru.

Si DevOps Guru détecte une anomalie actuelle sur votre instance de base de données, vous êtes alerté sur la page Bases de données de la console RDS. La console vous alerte également des anomalies survenues au cours des dernières 24 heures. Pour accéder à la page des anomalies à partir de la console RDS, cliquez sur le lien présent dans le message d'alerte. La console RDS vous alerte également dans la page de votre cluster de bases de données Amazon Aurora .

Anomalies contextuelles

Une anomalie contextuelle est un résultat dans la charge de base de données qui est associé à un insight réactif. Chaque anomalie contextuelle décrit un problème de performances Amazon Aurora spécifique qui requiert un examen. Par exemple, DevOps Guru for RDS peut vous recommander d'augmenter la capacité du processeur ou d'étudier les événements d'attente qui contribuent à la charge de la base de données.

> **Important**
>
> Nous vous recommandons de tester toutes les modifications apportées à une instance test avant de modifier une instance de production. Vous pouvez ainsi mieux appréhender l'impact d'une modification.

Pour en savoir plus, consultez la section Analyse des anomalies dans Amazon RDS dans le guide de l'utilisateur Amazon DevOps Guru.

Configuration de DevOps Guru pour RDS

Pour permettre à DevOps Guru for Amazon RDS de publier des informations pour une base de données Amazon Aurora suivantes.

Rubriques

- Configuration des politiques d'accès IAM pour DevOps Guru for RDS
- Activation de Performance Insights pour vos instances de base de données Aurora
- Activer DevOps Guru et spécifier la couverture des ressources

Configuration des politiques d'accès IAM pour DevOps Guru for RDS

Pour afficher les alertes de DevOps Guru dans la console RDS, votre utilisateur ou rôle Gestion des identités et des accès AWS (IAM) doit respecter l'une des politiques suivantes :

- La politique AWS gérée `AmazonDevOpsGuruConsoleFullAccess`
- La stratégie AWS gérée `AmazonDevOpsGuruConsoleReadOnlyAccess` et l'une des politiques suivantes :
  - La politique AWS gérée `AmazonRDSFullAccess`
  - Politique gérée par le client qui inclut `pi:GetResourceMetrics` et `pi:DescribeDimensionKeys`

Pour plus d’informations, consultez Configuration des politiques d’accès pour Performance Insights.

Activation de Performance Insights pour vos instances de base de données Aurora

DevOpsGuru for RDS s'appuie sur Performance Insights pour ses données. Sans Performance Insights, DevOps Guru publie des anomalies, mais n'inclut pas l'analyse détaillée ni les recommandations.

Lorsque vous créez un cluster de bases de données Aurora ou modifiez une instance de cluster, vous pouvez activer Performance Insights. Pour plus d’informations, consultez Activation ou désactivation de l’Analyse des performances pour Aurora.

Activer DevOps Guru et spécifier la couverture des ressources

Vous pouvez activer DevOps Guru pour qu'il surveille vos bases de données Amazon Aurora de l'une des manières suivantes.

Rubriques

- Activer DevOps Guru dans la console RDS
- Ajout de ressources Aurora dans la console Guru DevOps

- Ajout de ressources Aurora l'aide de CloudFormation

Activer DevOps Guru dans la console RDS

Vous pouvez emprunter plusieurs chemins dans la console Amazon RDS pour activer DevOps Guru.

Rubriques

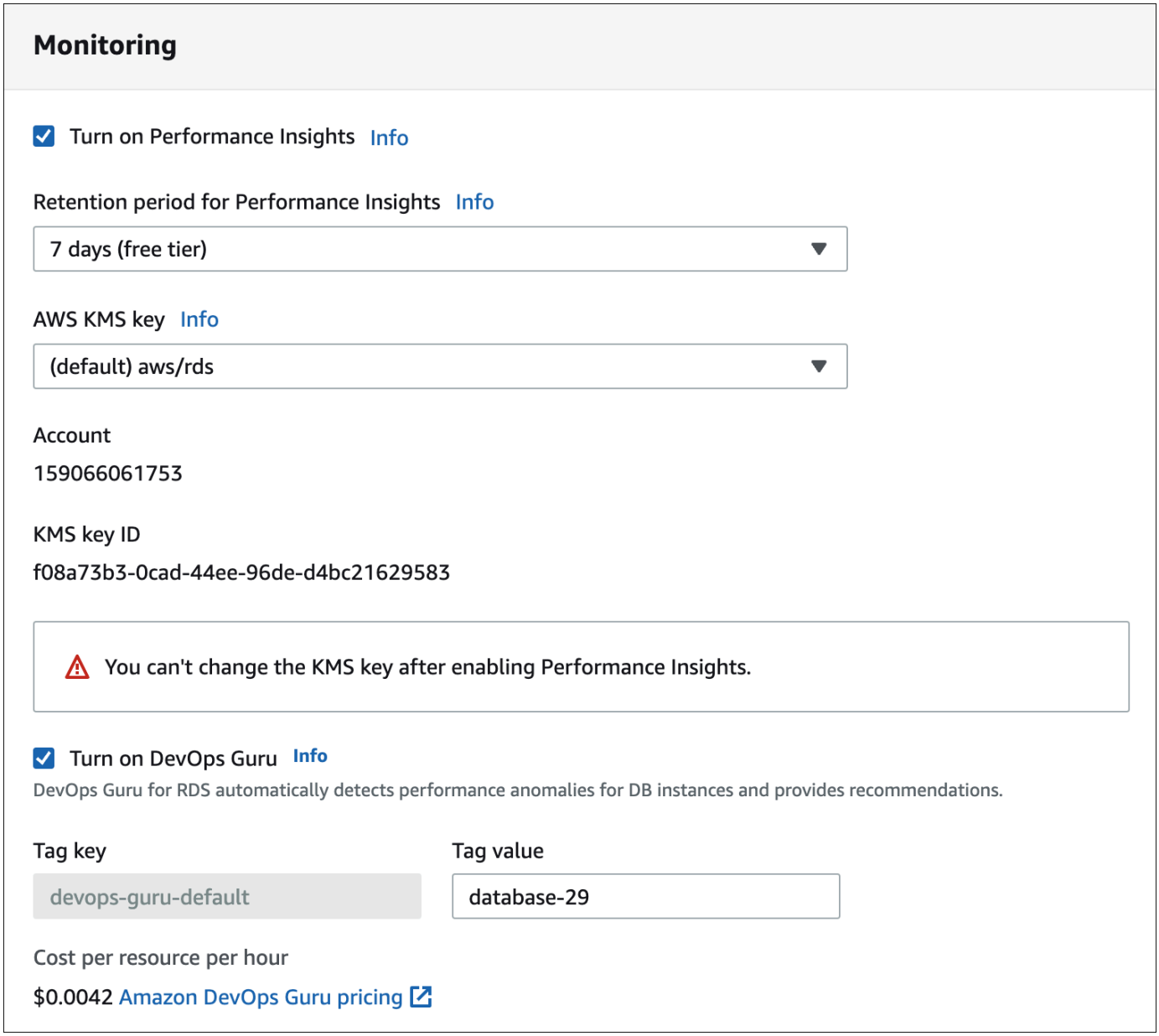
- Activer DevOps Guru lorsque vous créez une base de données Aurora
- Activer DevOps Guru depuis la bannière de notification
- Répondre à une erreur d'autorisation lorsque vous activez DevOps Guru

Activer DevOps Guru lorsque vous créez une base de données Aurora

Le flux de création inclut un paramètre qui active la couverture DevOps Guru pour votre base de données. Ce paramètre est activé par défaut lorsque vous choisissez le modèle Production.

Pour activer DevOps Guru lorsque vous créez une base de données Aurora

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Suivez les étapes de la section Création d’un cluster de bases de données, jusqu’à l’étape où vous choisissez les paramètres de surveillance, sans la réaliser.
3. Dans Monitoring (Surveillance), choisissez Turn on Performance Insights (Activer Performance Insights). Pour que DevOps Guru for RDS puisse fournir une analyse détaillée des anomalies de performance, Performance Insights doit être activé.
4. Choisissez Turn on DevOps Guru.



5. Créez une balise pour votre base de données afin que DevOps Guru puisse la surveiller. Procédez comme suit :

   - Dans le champ de texte Tag key (Clé de balise), saisissez un nom commençant par **`Devops-Guru-`**.
   - Dans le champ de texte Tag value (Valeur de balise), saisissez n'importe quelle valeur. Par exemple, si vous saisissez **`rds-database-1`** comme nom de votre base de données Aurora, vous pouvez également saisir **`rds-database-1`** comme valeur de balise.

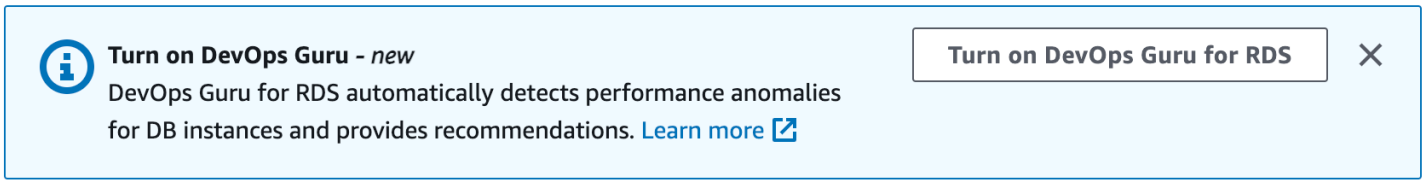
Pour plus d'informations sur les balises, consultez la section « Utiliser des balises pour identifier les ressources dans vos applications DevOps Guru » dans le guide de l'utilisateur Amazon DevOps Guru.

6. Suivez les étapes restantes fournies dans Création d'un cluster de bases de données.

Activer DevOps Guru depuis la bannière de notification

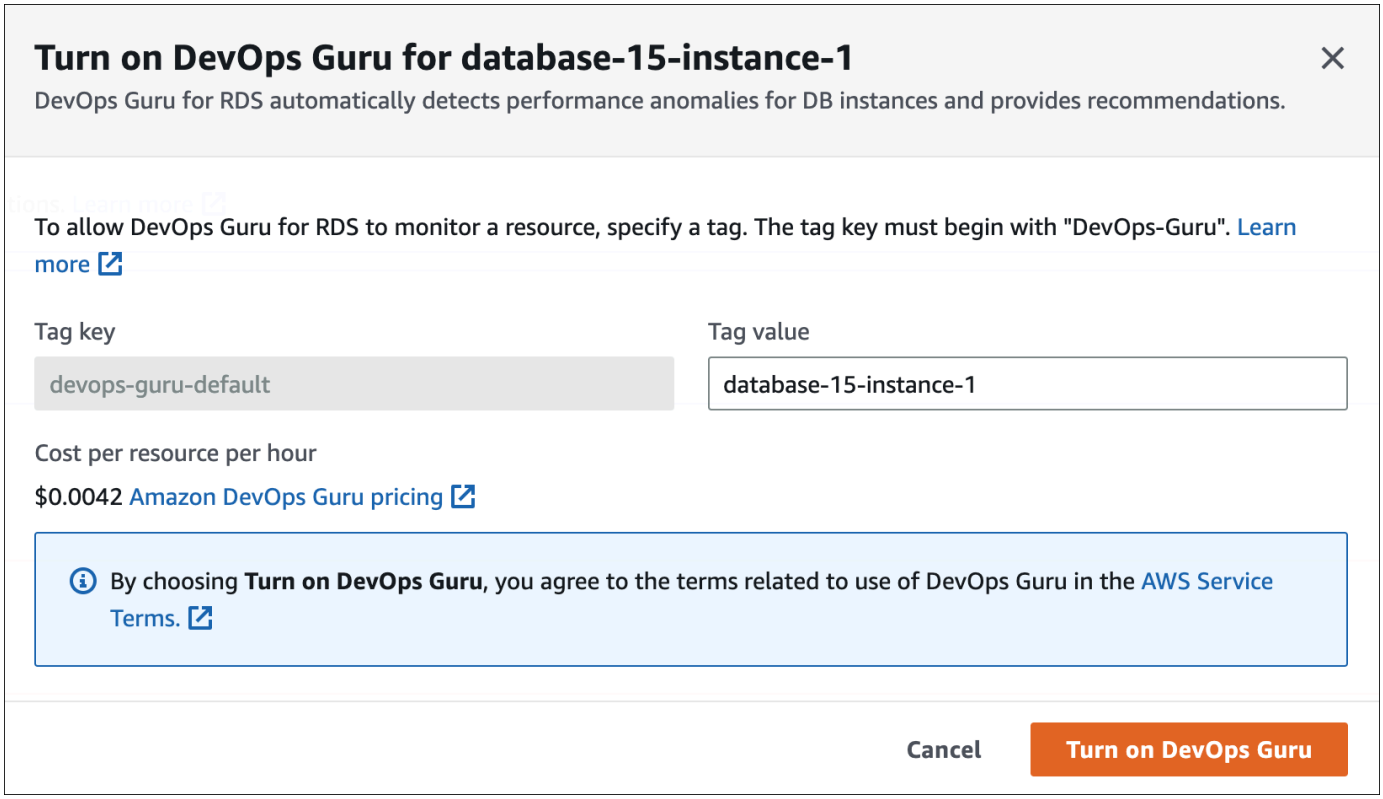
Si vos ressources ne sont pas couvertes par DevOps Guru, Amazon RDS vous en informe par le biais d'une bannière aux emplacements suivants :

- L'onglet Monitoring (Surveillance) d'une instance de cluster de bases de données
- Tableau de bord Performance Insights



Pour activer DevOps Guru pour votre base de données Aurora

1. Dans la bannière, choisissez Turn on DevOps Guru for RDS.
2. Saisissez un nom de clé et une valeur pour la balise. Pour plus d'informations sur les balises, consultez la section « Utiliser des balises pour identifier les ressources dans vos applications DevOps Guru » dans le guide de l'utilisateur Amazon DevOps Guru.



3. Choisissez Turn on DevOps Guru.

Répondre à une erreur d'autorisation lorsque vous activez DevOps Guru

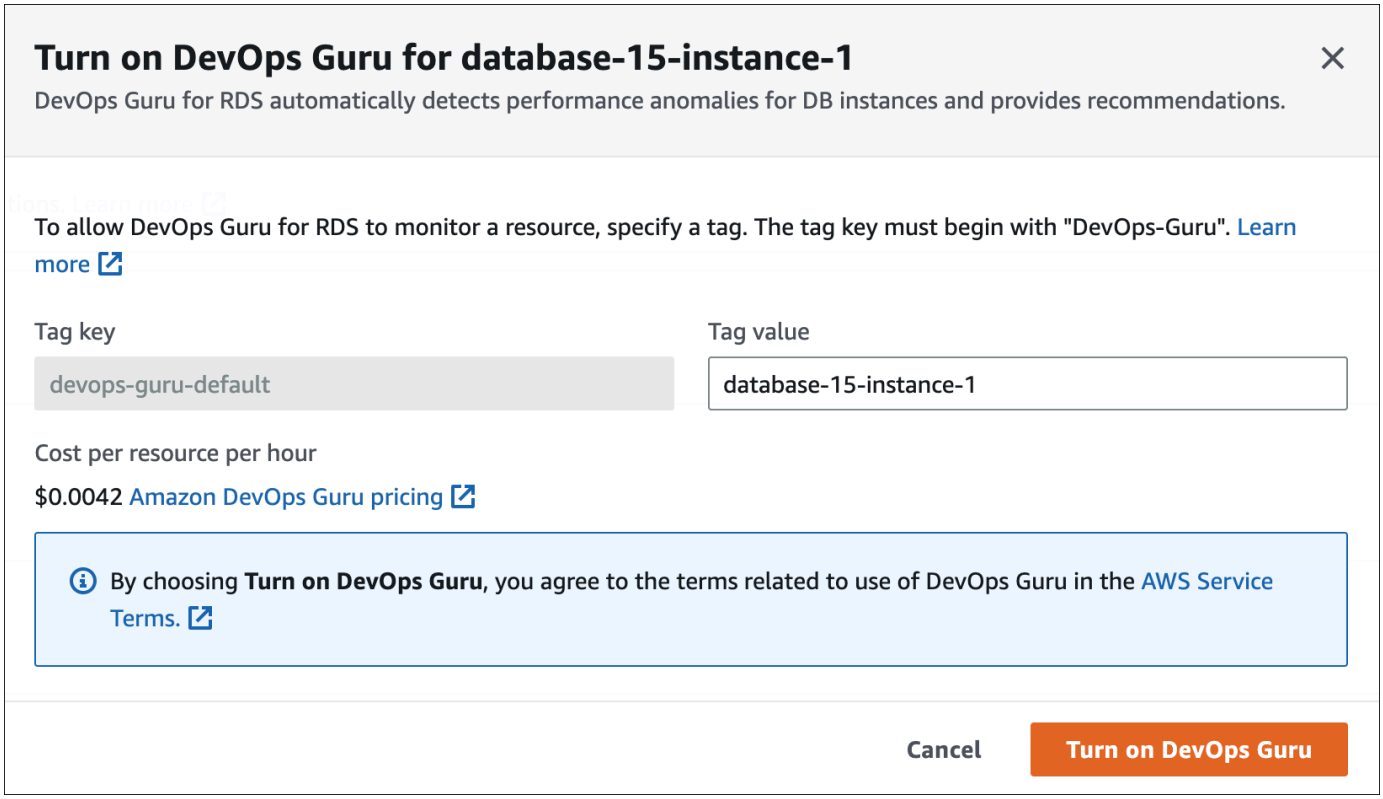
Si vous activez DevOps Guru depuis la console RDS lorsque vous créez une base de données, RDS peut afficher la bannière suivante concernant les autorisations manquantes.

Pour résoudre une erreur d'autorisations

1. Accordez à votre utilisateur ou rôle IAM le rôle géré par l'utilisateur `AmazonDevOpsGuruConsoleFullAccess`. Pour plus d'informations, consultez Configuration des politiques d'accès IAM pour DevOps Guru for RDS.
2. Ouvrez la console RDS.
3. Dans le panneau de navigation, choisissez Performance Insights.
4. Choisissez une instance de base de données dans le cluster que vous venez de créer.
5. Choisissez le commutateur pour activer DevOpsGuru for RDS.

DevOps Guru for RDS

6. Choisissez une valeur de balise. Pour plus d'informations, consultez la section « Utiliser des balises pour identifier les ressources dans vos applications DevOps Guru » dans le guide de l'utilisateur Amazon DevOps Guru.



7. Choisissez Turn on DevOps Guru.

Ajout de ressources Aurora dans la console Guru DevOps

Vous pouvez spécifier la couverture de vos ressources DevOps Guru sur la console DevOps Guru. Suivez l'étape décrite dans Spécifiez la couverture de vos ressources DevOps Guru dans le guide de l'utilisateur Amazon DevOps Guru. Lorsque vous modifiez vos ressources analysées, choisissez l'une des options suivantes :

- Choisissez Toutes les ressources du compte pour analyser toutes les ressources prises en charge, y compris les bases de données Aurora , dans votre région. Compte AWS
- Choisissez CloudFormation des piles pour analyser les bases de données Aurora qui se trouvent dans les piles de votre choix. Pour plus d'informations, consultez la section Utiliser des AWS CloudFormation piles pour identifier les ressources de vos applications DevOps Guru dans le guide de l'utilisateur Amazon DevOps Guru.

- Choisissez Balises pour analyser les bases de données Aurora que vous avez balisées. Pour plus d'informations, consultez la section Utiliser des balises pour identifier les ressources dans vos applications DevOps Guru dans le guide de l'utilisateur Amazon DevOps Guru.

Pour plus d'informations, consultez Enable DevOps Guru dans le guide de l'utilisateur Amazon DevOps Guru.

Ajout de ressources Aurora l'aide de CloudFormation

Vous pouvez utiliser des balises pour étendre la couverture de vos ressources Aurora à vos modèles. CloudFormation La procédure suivante suppose que vous disposez d'un CloudFormation modèle à la fois pour votre instance de base de données Aurora et pour votre stack Guru. DevOps

Pour spécifier une instance de base de données Aurora à l'aide d'une balise CloudFormation

1. Dans le CloudFormation modèle de votre instance de base de données, définissez une balise à l'aide d'une paire clé/valeur.

   L'exemple suivant attribue la valeur `my-aurora-db-instance1` à `Devops-guru-cfn-default` pour une instance de la base de données Aurora.

   ```
   MyAuroraDBInstance1:
     Type: "AWS::RDS::DBInstance"
     Properties:
       DBClusterIdentifier: my-aurora-db-cluster
       DBInstanceIdentifier: my-aurora-db-instance1
       Tags:
         - Key: Devops-guru-cfn-default
           Value: devopsguru-my-aurora-db-instance1
   ```

2. Dans le CloudFormation modèle de votre pile DevOps Guru, spécifiez le même tag dans votre filtre de collecte de ressources.

   L'exemple suivant configure DevOps Guru pour fournir une couverture à la ressource avec la valeur `my-aurora-db-instance1` de balise.

   ```
   DevOpsGuruResourceCollection:
     Type: AWS::DevOpsGuru::ResourceCollection
     Properties:
       ResourceCollectionFilter:
         Tags:
   ```

```
        - AppBoundaryKey: "Devops-guru-cfn-default"
          TagValues:
          - "devopsguru-my-aurora-db-instance1"
```

L'exemple suivant fournit une prise en charge pour toutes les ressources à l'intérieur du périmètre de l'application `Devops-guru-cfn-default`.

```
DevOpsGuruResourceCollection:
  Type: AWS::DevOpsGuru::ResourceCollection
  Properties:
    ResourceCollectionFilter:
      Tags:
        - AppBoundaryKey: "Devops-guru-cfn-default"
          TagValues:
          - "*"
```

Pour plus d'informations, consultez AWS::DevOpsGuru::ResourceCollectionAWS : :RDS : : DBInstance dans le guide de l'CloudFormation utilisateur.

Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée

La surveillance améliorée vous permet de surveiller le système d'exploitation de votre instance de base de données en temps réel. Lorsque vous voulez voir comment différents processus ou threads utilisent l'UC, les métriques de la surveillance améliorée sont utiles.

Rubriques

Vue d'ensemble de la surveillance améliorée

Amazon RDS fournit des métriques en temps réel pour le système d'exploitation sur lequel votre instance de base de données s'exécute. Vous pouvez afficher toutes les métriques système et les informations de processus pour vos instances de base de données RDS sur la console. Vous pouvez gérer les métriques que vous souhaitez surveiller pour chaque instance et personnaliser le tableau de bord en fonction de vos besoins. Pour obtenir une description des métriques de la surveillance améliorée, consultez Métriques du système d'exploitation dans la surveillance améliorée.

RDS fournit les métriques de la surveillance améliorée à votre compte Amazon CloudWatch Logs. Vous pouvez créer des filtres de métrique dans CloudWatch à partir de CloudWatch Logs et afficher les graphiques sur le tableau de bord CloudWatch. Vous pouvez utiliser la sortie JSON de surveillance améliorée depuis CloudWatch Logs dans un système de surveillance de votre choix. Pour plus d'informations, consultez Surveillance améliorée dans la FAQ Amazon RDS.

Rubriques

Différences entre les métriques CloudWatch et de surveillance améliorée

Un hyperviseur crée et exécute des machines virtuelles (VM). À l'aide d'un hyperviseur, une instance peut prendre en charge plusieurs machines virtuelles invitées en partageant virtuellement la mémoire et l'UC. CloudWatch recueille les métriques relatives à l'utilisation de l'UC à partir de l'hyperviseur pour une instance de base de données. En revanche, la surveillance améliorée collecte ses métriques à partir d'un agent sur l'instance de base de données.

Vous pouvez rencontrer des différences de mesures entre le CloudWatch et la surveillance améliorée, car une petite quantité de travail est effectuée par la couche de l'hyperviseur. Les différences peuvent être plus importantes si vos instances de base de données utilisent des classes d'instance plus petites. Dans ce scénario, davantage de machines virtuelles (VM) sont probablement gérées par la couche de l'hyperviseur sur une seule instance physique.

Pour obtenir une description des métriques de la surveillance améliorée, consultez Métriques du système d'exploitation dans la surveillance améliorée. Pour plus d'informations sur les métriques Amazon CloudWatch, consultez le guide de l'utilisateur Amazon CloudWatch.

Conservation des métriques de surveillance améliorée

Par défaut, les métriques de surveillance améliorée sont stockées dans CloudWatch Logs pendant 30 jours. Cette période de rétention est différente des métriques CloudWatch classiques.

Pour modifier la durée de stockage des métriques dans CloudWatch Logs, modifiez la période de rétention pour le groupe de journaux `RDSOSMetrics` dans la console CloudWatch. Pour plus d'informations, consultez Modification de la rétention des données de journaux dans CloudWatch Logs dans le Guide de l'utilisateur Amazon CloudWatch Logs.

Coût de la surveillance améliorée

Les métriques de surveillance améliorée sont stockées dans les journaux CloudWatch plutôt que dans les métriques Cloudwatch. Le coût de la surveillance améliorée dépend des facteurs suivants :

- Vous n'êtes facturé pour l'utilisation de la surveillance améliorée que si vous dépassez la quantité de stockage et de transfert de données fournie par Amazon CloudWatch Logs. Les frais sont basés sur les taux de transfert et de stockage des données CloudWatch Logs.
- La quantité d'informations transférées pour une instance RDS est directement proportionnelle à la granularité définie pour la fonction de surveillance améliorée. Plus l'intervalle de surveillance est court, plus la fréquence des rapports sur les métriques du système d'exploitation est élevée, ce qui

augmente les coûts de surveillance. Pour gérer les coûts, définissez différentes granularités pour différentes instances de vos comptes.

- Les coûts d'utilisation de la surveillance améliorée sont appliqués pour chaque instance de base de données pour laquelle l'option est activée. Surveiller un grand nombre d'instances de bases de données est plus onéreux que de n'en surveiller que quelques unes.
- Les instances de bases de données qui prennent en charge une charge de travail nécessitant des calculs intensifs doivent signaler une activité de traitement du système d'exploitation plus intense et des coûts plus élevés pour la surveillance améliorée.

Pour plus d'informations sur la tarification, consultez Tarification Amazon CloudWatch.

Configuration et activation de la surveillance améliorée

Pour utiliser la surveillance améliorée, vous devez créer un rôle IAM, puis activer la surveillance améliorée.

Rubriques

- Création d'un rôle IAM pour la surveillance améliorée
- Activer et désactiver la surveillance améliorée
- Lutter contre le problème de l'adjoint confus

Création d'un rôle IAM pour la surveillance améliorée

La surveillance améliorée nécessite l'autorisation d'agir en votre nom pour envoyer des informations métriques du système d'exploitation à CloudWatch Logs. Vous accordez des autorisations de surveillance améliorée à l'aide d'un rôle Gestion des identités et des accès AWS (IAM). Vous pouvez soit créer ce rôle lorsque vous activez la surveillance améliorée, soit le créer au préalable.

Rubriques

- Création du rôle IAM lorsque vous activez la surveillance améliorée
- Création du rôle IAM avant d'activer la surveillance améliorée

Création du rôle IAM lorsque vous activez la surveillance améliorée

Lorsque vous activez la surveillance améliorée dans la console RDS, Amazon RDS peut créer le rôle IAM requis pour vous. Le rôle est nommé `rds-monitoring-role`. RDS utilise ce rôle pour

l’instance de base de données, le réplica en lecture spécifié ou le cluster de bases de données Multi-AZ.

Pour créer le rôle IAM lors de l’activation de la surveillance améliorée

1. Suivez les étapes de Activer et désactiver la surveillance améliorée.
2. Définissez Rôle de surveillance sur Par défaut à l’étape où vous choisissez un rôle.

Création du rôle IAM avant d’activer la surveillance améliorée

Vous pouvez créer le rôle requis avant d’activer la surveillance améliorée. Lorsque vous activez la surveillance améliorée, spécifiez le nom de votre nouveau rôle. Vous devez créer ce rôle requis si vous activez la surveillance améliorée à l’aide de l’AWS CLI ou de l’API RDS.

L’utilisateur qui active la surveillance améliorée doit se voir accorder l’autorisation `PassRole`. Pour plus d'informations, consultez l'exemple 2 de la section Octroi à un utilisateur des autorisations pour transmettre un rôle à un AWS service dans le guide de l'utilisateur IAM.

Pour créer un rôle IAM pour la surveillance améliorée Amazon RDS

1. Ouvrez la console IAM à https://console.aws.amazon.coml'adresse.
2. Dans le panneau de navigation, choisissez Rôles.
3. Sélectionnez Create role (Créer un rôle).
4. Choisissez l’onglet Service AWS, puis choisissez RDS dans la liste de services.
5. Choisissez RDS - Enhanced Monitoring (RDS - Surveillance améliorée), puis Next (Suivant).
6. Assurez-vous que les politiques d'autorisations indiquent Amazon RDSEnhanced MonitoringRole, puis choisissez Next.
7. Dans le champ Role name (Nom de rôle), saisissez un nom pour votre rôle. Par exemple, entrez **`emaccess`**.

   L'entité de confiance correspondant à votre rôle est le AWS service monitoring.rds.amazonaws.com.
8. Choisissez Créer un rôle.

Activer et désactiver la surveillance améliorée

Vous pouvez gérer la surveillance améliorée à l'aide de l'API AWS Management ConsoleAWS CLI, ou RDS. Vous pouvez définir différents niveaux de détails pour la collecte de métriques sur chaque cluster de bases de données. Vous pouvez également activer la surveillance améliorée pour un cluster de bases de données existant depuis la console.

Console

Vous pouvez activer la surveillance améliorée lorsque vous créez un cluster de bases de données ou un réplica en lecture, ou lorsque vous modifiez un cluster de bases de données. Si vous modifiez une instance ou un cluster de bases de données afin d'activer la surveillance améliorée, vous n'avez pas besoin de redémarrer votre instance de base de données pour que la modification prenne effet.

Vous pouvez activer la surveillance améliorée dans la console RDS lorsque vous effectuez l'une des opérations suivantes sur la page Bases de données :

- Création d'un cluster de bases de données : choisissez Create database (Créer une base de données).
- Créer un réplica en lecture : choisissez Actions, puis Create read replica (Créer un réplica en lecture).
- Modifier une instance de base de données ou un cluster de bases de données : choisissez Modifier.

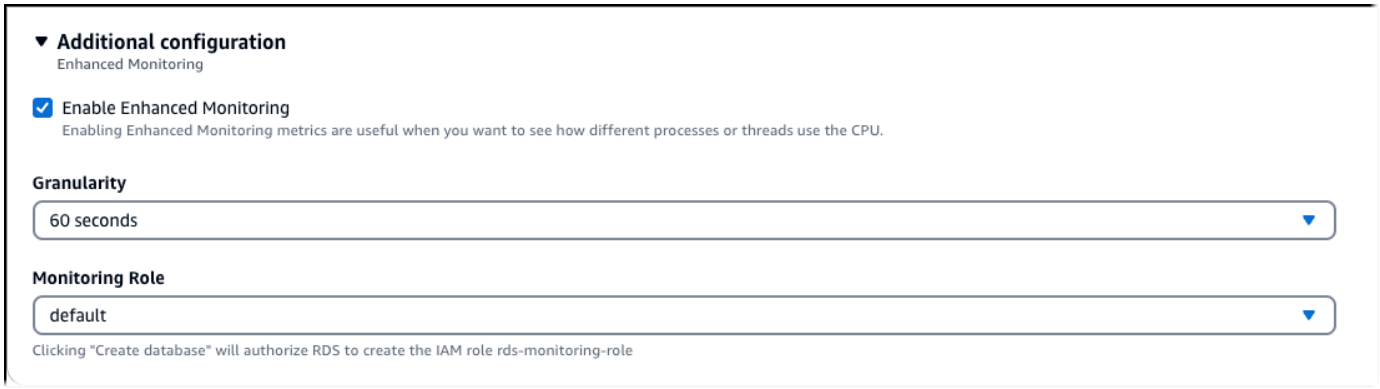
> **Note**
>
> Lorsque vous activez la surveillance améliorée pour un cluster de bases de données, vous ne pouvez pas gérer la surveillance améliorée pour des instances de base de données individuelles au sein du cluster.

Si vous gérez la surveillance améliorée pour des instances de base de données individuelles dans un cluster de bases de données et que la granularité est définie sur des valeurs différentes pour les différentes instances, votre cluster de bases de données est hétérogène en ce qui concerne la surveillance améliorée. Dans de tels cas, vous ne pouvez pas modifier le cluster de bases de données pour gérer la surveillance améliorée au niveau du cluster.

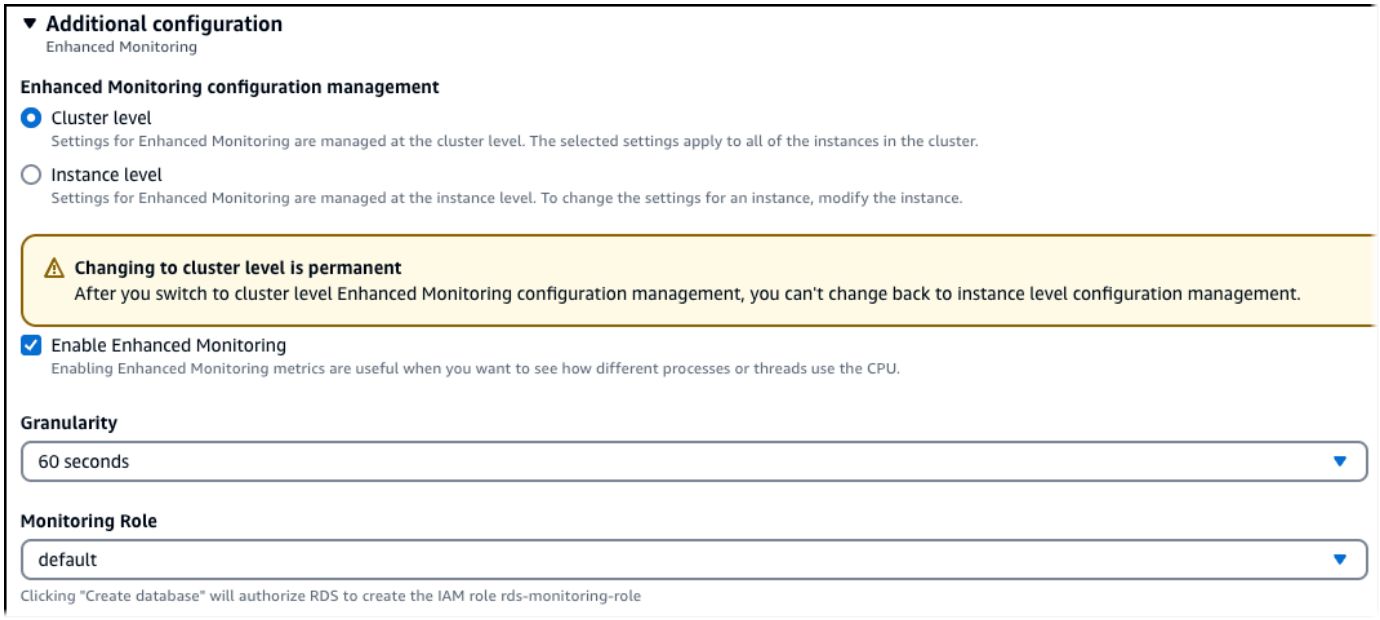
Pour activer ou désactiver la surveillance améliorée dans la console RDS

1. Descendez jusqu'à Additional configuration (Configuration supplémentaire).
2. Dans Surveillance, choisissez Activer la surveillance améliorée pour votre , cluster de bases de données ou réplica en lecture. L'activation de la surveillance améliorée au niveau du cluster vous permet de gérer les paramètres et les options de surveillance améliorée au niveau du cluster. Les paramètres au niveau du cluster s'appliquent à toutes les instances de base de données du cluster.Désélectionnez l'option pour désactiver la surveillance améliorée au niveau du cluster. Vous pouvez modifier ultérieurement les paramètres de surveillance améliorée pour les instances de base de données individuelles du cluster.

   Sur la page Créer une base de données, vous pouvez choisir d'activer la surveillance améliorée au niveau du cluster.



   Si vous n'activez pas la surveillance améliorée lors de la création d'un cluster, vous pouvez modifier le cluster sur la page Modifier le cluster de bases de données.



> **Note**
>
> Vous ne pouvez pas gérer la surveillance améliorée pour une instance de base de données individuelle dans un cluster de bases de données pour lequel la surveillance améliorée est déjà gérée au niveau du cluster.

3. Vous ne pouvez pas choisir de gérer la surveillance améliorée au niveau du cluster si celui-ci est hétérogène par rapport à la surveillance améliorée. Pour gérer la surveillance améliorée au niveau du cluster, modifiez les paramètres de surveillance améliorée pour chaque instance afin qu'ils correspondent. Vous pouvez désormais choisir de gérer la surveillance améliorée au niveau du cluster lorsque vous modifiez celui-ci.
4. Définissez la propriété Monitoring Role sur le rôle IAM que vous avez créé pour permettre à Amazon RDS de communiquer avec Amazon CloudWatch Logs à votre place, ou choisissez Default pour que RDS crée un rôle nommé pour vous. `rds-monitoring-role`
5. Définissez la propriété Granularité sur l'intervalle, en secondes, entre les points lorsque les métriques sont collectées pour votre instance de base de données, cluster de bases de données ou réplica en lecture. La propriété Granularité peut être définie sur l'une des valeurs suivantes : `1`, `5`, `10`, `15`, `30` ou `60`.

   La console RDS est actualisée toutes les 5 secondes. Si la granularité est définie sur 1 seconde dans la console RDS, les métriques mises à jour s'affichent toutes les 5 secondes uniquement. Vous pouvez récupérer les mises à jour des métriques en une seconde à l'aide CloudWatch des journaux.

AWS CLI

Pour activer la surveillance améliorée à l'aide des commandes suivantesAWS CLI, définissez l'`--monitoring-interval`option sur une valeur autre que le rôle dans lequel vous l'avez créé `0` et définissez l'`--monitoring-role-arn`option sur le rôle dans lequel vous l'avez crééCréation d'un rôle IAM pour la surveillance améliorée.

- create-db-cluster
- modify-db-cluster
- create-db-instance
- create-db-instance-read-réplique
- modify-db-instance

L'option `--monitoring-interval` spécifie l'intervalle, en secondes, entre les points lorsque des métriques de surveillance améliorée sont collectées. Les valeurs valides pour l'option sont `0`, `1`, `5`, `10`, `15`, `30` et `60`.

Pour désactiver la surveillance améliorée à l'aide deAWS CLI, définissez l'`--monitoring-interval`option sur `0` dans ces commandes.

Example

L'exemple suivant active la surveillance améliorée pour une instance de base de données :

Pour Linux, macOS ou Unix :

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Pour Windows :

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

L'exemple suivant active la surveillance améliorée pour un cluster de bases de données :

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbinstance \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbinstance ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Example

L'exemple suivant active la surveillance améliorée pour une cluster de bases de données Multi-AZ :

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --monitoring-interval 30 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --monitoring-interval 30 ^
    --monitoring-role-arn arn:aws:iam::123456789012:role/emaccess
```

API RDS

Pour activer la surveillance améliorée à l'aide de l'AP RDS, définissez le paramètre `MonitoringInterval` sur une valeur autre que `0` et définissez le paramètre `MonitoringRoleArn` sur le rôle que vous avez créé dans Création d'un rôle IAM pour la surveillance améliorée. Définissez ces paramètres dans les actions suivantes :

- CréerDBCluster
- ModifyDBCluster
- CréerDBInstance
- CréerDBInstanceReadReplica
- ModifyDBInstance

Le paramètre `MonitoringInterval` spécifie l'intervalle, en secondes, entre les points lorsque des métriques de surveillance améliorée sont collectées. Les valeurs valides sont `0`, `1`, `5`, `10`, `15`, `30` et `60`.

Pour désactiver la surveillance améliorée à l'aide de l'API RDS, définissez `MonitoringInterval` sur `0`.

Lutter contre le problème de l'adjoint confus

Le problème de député confus est un problème de sécurité dans lequel une entité qui n'est pas autorisée à effectuer une action peut contraindre une entité plus privilégiée à le faire. EnAWS, l'usurpation d'identité interservices peut entraîner la confusion des adjoints. L'usurpation d'identité entre services peut se produire lorsqu'un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé et ses autorisations utilisées pour agir sur les ressources d'un autre client auxquelles on ne serait pas autorisé à accéder autrement. Pour éviter cela, AWS fournit des outils qui vous aident à protéger vos données pour tous les services avec des principaux de service qui ont eu accès aux ressources de votre compte. Pour plus d'informations, consultez Le problème de l'adjoint confus.

Afin de limiter les autorisations octroyées par Amazon RDS à un autre service pour la ressource, nous vous recommandons d'utiliser les clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount` dans une politique d'approbation pour votre rôle de surveillance améliorée. Si vous utilisez les deux clés de contexte de condition globale, elles doivent utiliser le même ID de compte.

Le moyen le plus efficace de se protéger contre le problème de député confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource. Pour Amazon RDS, définissez `aws:SourceArn` sur `arn:aws:rds:`*`Region`*`:`*`my-account-id`*`:db:`*`dbname`*.

L'exemple suivant utilise les clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount` dans une politique d'approbation afin d'empêcher le problème d'adjoint confus.

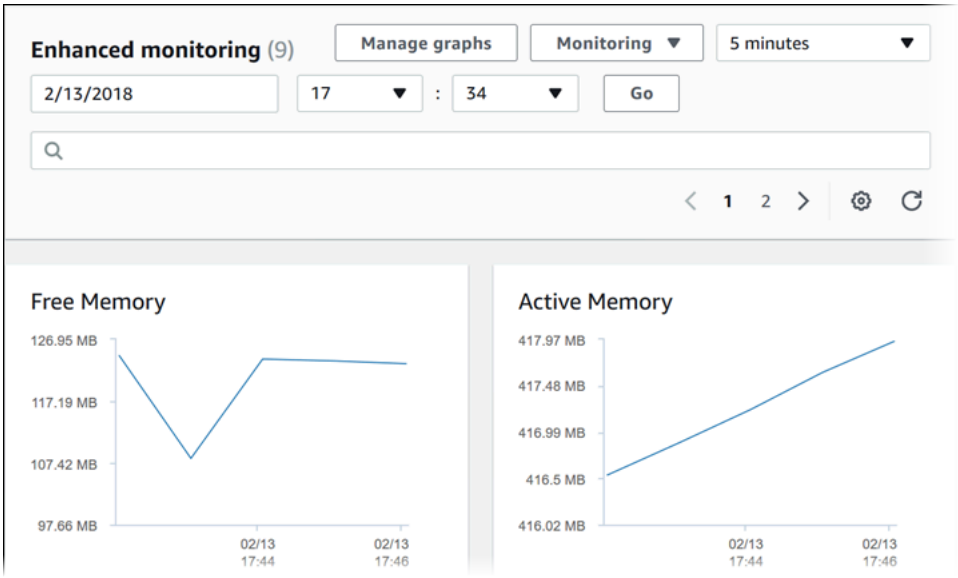
JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "monitoring.rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:rds:Region:my-account-id:db:dbname"
        },
        "StringEquals": {
          "aws:SourceAccount": "my-account-id"
        }
      }
    }
  ]
}
```

Affichage des métriques du système d'exploitation dans la console RDS

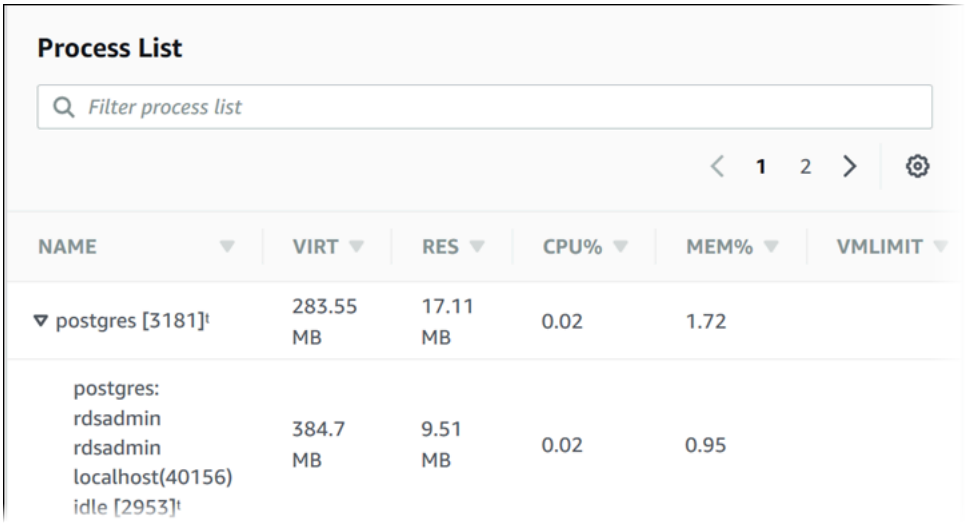
Vous pouvez afficher les métriques du système d'exploitation relevées par la surveillance améliorée dans la console RDS en choisissant Enhanced monitoring (Surveillance améliorée) pour Monitoring (Surveillance).

L'exemple suivant montre la page de surveillance améliorée. Pour obtenir une description des métriques de la surveillance améliorée, consultez Métriques du système d'exploitation dans la surveillance améliorée.



Si vous voulez afficher des détails sur les processus qui s'exécutent sur votre instance de base de données, choisissez Liste de processus de système d'exploitation pour Surveillance.

La vue Liste des processus est affichée ci-dessous.



Les métriques de surveillance améliorée affichées dans la vue Liste des processus sont organisées de la manière suivante :

- Processus enfants RDS : affiche un résumé des processus RDS prenant en charge l'instance de base de données, par exemple `aurora` pour les clusters de bases de données Amazon Aurora. Les threads du processus sont imbriqués sous le processus parent. Les threads du processus affichent uniquement l'utilisation de l'UC, car les autres métriques sont identiques pour tous les threads du processus. La console affiche 100 processus et threads maximum. Les résultats

représentent une combinaison des principaux processus et threads de la consommation de l'UC et de la mémoire. S'il existe plus de 50 processus et 50 threads, la console affiche les 50 premiers éléments de chaque catégorie. Cette présentation vous aide à identifier les processus ayant le plus d'impact sur les performances.

- Processus RDS : affiche un résumé des ressources utilisées par l'agent de gestion RDS, des processus de surveillance des diagnostics et des autres AWS processus requis pour prendre en charge les instances de base de données RDS.
- OS processes (Processus SE) – Affiche un récapitulatif des processus du noyau et du système, qui ont généralement un faible impact sur les performances.

Les éléments répertoriés pour chaque processus sont les suivants :

- VIRT – Affiche la taille virtuelle du processus.
- RES : affiche la mémoire physique réelle en cours d'utilisation par le processus.
- UC% : affiche le pourcentage de la bande passante totale de l'UC utilisé par le processus.
- MEM% – Affiche le pourcentage de mémoire totale utilisé par le processus.

Les données de surveillance affichées dans la console RDS sont extraites d'Amazon CloudWatch Logs. Vous pouvez également récupérer les métriques d'une instance de base de données sous forme de flux de journal à partir de CloudWatch Logs. Pour de plus amples informations, veuillez consulter Affichage des mesures du système d'exploitation à l'aide de CloudWatch Logs.

Les métriques de surveillance améliorée ne sont pas renvoyées dans les situations suivantes :

- Basculement de l'instance de base de données.
- Modification de la classe d'instance de l'instance de base de données (dimensionnement du calcul).

Les métriques de surveillance améliorée sont renvoyées pendant le redémarrage d'une instance de base de données, car seul le moteur de base de données est redémarré. Les métriques concernant le système d'exploitation continuent d'être relevées.

Affichage des mesures du système d’exploitation à l’aide de CloudWatch Logs

Après avoir activé la surveillance améliorée pour votre un cluster de bases de données, vous pouvez afficher les métriques qui s’y rapportent à l’aide de CloudWatch Logs, chaque flux de journal représentant une seule instance de base de données surveillée. L’identifiant du flux de journal est l’identifiant de la ressource (`DbiResourceId`) de l’instance de base de données ou du cluster de bases de données.

Pour afficher les données du journal de la surveillance améliorée

1. Ouvrez la console CloudWatch à l’adresse https://console.aws.amazon.com/cloudwatch/.
2. Si nécessaire, choisissez l’Région AWS dans laquelle se trouve votre cluster de bases de données. Pour plus d’informations, consultez Régions et points de terminaison dans la Référence générale Amazon Web Services.
3. Choisissez Logs (Journaux) dans le panneau de navigation.
4. Choisissez RDSOSMetrics dans la liste de groupes de journaux.
5. Choisissez le flux de journal à afficher dans la liste des flux de journaux.

Référence des métriques pour Amazon Aurora

Dans cette référence, vous trouverez les descriptions des métriques Amazon Aurora pour Amazon CloudWatch, Performance Insights et Enhanced Monitoring (Surveillance améliorée).

Rubriques

CloudWatch Métriques Amazon pour Amazon Aurora

L'espace de noms `AWS/RDS` inclut les métriques suivantes qui s'appliquent aux entités de base de données qui s'exécutent sur Amazon Aurora. Certaines métriques s'appliquent à Aurora MySQL, à Aurora PostgreSQL ou aux deux. En outre, certaines métriques sont spécifiques à un cluster de bases de données, à une instance de base de données principale, à une instance de base de données de réplica ou à toutes les instances de base de données.

Pour les métriques de base de données globale Aurora, consultez Métriques Amazon CloudWatch pour le transfert d'écriture dans Aurora MySQL et Métriques Amazon CloudWatch pour le transfert d'écriture dans Aurora PostgreSQL. Pour les métriques de requête parallèle Aurora, consultez Surveillance des requêtes parallèles pour Aurora MySQL.

Rubriques

Métriques de niveau cluster pour Amazon Aurora

Le tableau suivant décrit les métriques spécifiques aux clusters Aurora.

Métrique	Description	S'applique à	Uunités
`AuroraGlobalDBDataTransferBytes`	Dans une base de données globale Aurora, quantité de données de journalisation transférées de la AWS région source vers une AWS région secondaire. **Note** Cette métrique est disponible uniquement dans les Régions AWS secondaires.	Aurora MySQL et Aurora PostgreSQL	Octets
`AuroraGlobalDBProgressLag`	Dans une base de données Aurora Global Database, la mesure du retard du cluster secondaire par rapport au cluster principal pour les transactions utilisateur et système. **Note** Cette métrique est disponible uniquement dans les Régions AWS secondaires.	Aurora MySQL et Aurora PostgreSQL	Millisecondes
`AuroraGlobalDBReplicatedWriteIO`	Dans une base de données globale Aurora, le nombre d' I/O opérations d'écriture	Aurora MySQL	Nombre

Métrique	Description	S'applique à	Uunités
	répliquées depuis la AWS région principale vers le volume de cluster d'une AWS région secondaire. Les calculs de facturation pour les AWS régions secondaires d'une base de données globale sont utilisés pour `VolumeWriteIOPs` prendre en compte les écritures effectuées au sein du cluster. Les calculs de facturation pour la AWS région principale d'une base de données globale sont utilisés pour `VolumeWriteIOPs` prendre en compte l'activité d'écriture au sein de ce cluster et `AuroraGlobalDBReplicatedWriteIO` pour tenir compte de la réplication entre régions au sein de la base de données globale. **Note** Cette métrique est disponible uniquement dans les Régions AWS secondaires.	et Aurora PostgreSQL	

Métrique	Description	S'applique à	Uunités
`AuroraGlobalDBReplicationLag`	Pour une base de données Aurora Global Database, importance du retard de réplication des mises à jour à partir de la région AWS principale. **Note** Cette métrique est disponible uniquement dans les Régions AWS secondaires.	Aurora MySQL et Aurora PostgreSQL	Millisecondes
`AuroraGlobalDBRPOLag`	Dans une base de données Aurora Global Database, le délai de l'objectif de point de reprise (RPO). Cette métrique mesure le retard du cluster secondaire par rapport au cluster principal pour les transactions utilisateur. **Note** Cette métrique est disponible uniquement dans les Régions AWS secondaires.	Aurora MySQL et Aurora PostgreSQL	Millisecondes

Métrique	Description	S’applique à	Uunités
`AuroraVolumeBytesLeftTotal`	Espace disponible restant pour le volume du cluster. À mesure que le volume du cluster augmente, cette valeur diminue. S'il atteint zéro, le cluster signale une out-of-space erreur. Si vous voulez détecter si votre cluster Aurora MySQL approche de la limite de taille de 128 tébioctets (Tio), cette valeur est plus simple et plus fiable à surveiller que `VolumeBytesUsed` . `AuroraVolumeBytesLeftTotal` tient compte du stockage utilisé pour la gestion interne et d’autres attributions qui n’affecte nt pas la facturation du stockage.	Aurora MySQL	Octets
`BacktrackChangeRecordsCreationRate`	Nombre d’enregistrements de modification de retour en arrière créés en 5 minutes pour votre cluster de bases de données.	Aurora MySQL	Compte par 5 minutes
`BacktrackChangeRecordsStored`	Nombre d’enregistrements de modification de retour en arrière utilisés par votre cluster de bases de données.	Aurora MySQL	Nombre

Métrique	Description	S’applique à	Uunités
`BackupRetentionPeriodStorageUsed`	Quantité totale de stockage de sauvegarde utilisée pour prendre en charge la fonction de point-in-time restauration dans la fenêtre de conservation des sauvegardes du cluster de base de données Aurora. Ce montant est inclus dans le total déclaré par la métrique `TotalBackupStorageBilled` . Calculée séparément pour chaque cluster Aurora. Pour obtenir des instructions, consultez Comprendre l’utilisation du stockage de sauvegarde Amazon Aurora.	Aurora MySQL et Aurora PostgreSQL	Octets
`ServerlessDatabaseCapacity`	Capacité actuelle d'un cluster de bases de données Aurora Serverless.	Aurora MySQL et Aurora PostgreSQL	Nombre

Métrique	Description	S’applique à	Uunités
`SnapshotStorageUsed`	Quantité totale de stockage de sauvegarde consommée par tous les instantanés Aurora pour un cluster de bases de données Aurora en dehors de sa période de rétention des sauvegard es. Ce montant est inclus dans le total déclaré par la métrique `TotalBack upStorageBilled` . Calculée séparément pour chaque cluster Aurora. Pour obtenir des instructi ons, consultez Comprendr e l’utilisation du stockage de sauvegarde Amazon Aurora.	Aurora MySQL et Aurora PostgreSQL	Octets

Métrique	Description	S’applique à	Uunités
`TotalBackupStorageBilled`	Quantité totale de stockage de sauvegarde en octets qui vous est facturée pour un cluster de bases de données Aurora spécifique. La métrique Inclut le stockage de sauvegarde mesuré par les métriques `BackupRetentionPeriodStorageUsed` et `SnapshotStorageUsed` . Cette métrique est calculée séparément pour chaque cluster Aurora. Pour obtenir des instructions, consultez Comprendre l’utilisation du stockage de sauvegarde Amazon Aurora.	Aurora MySQL et Aurora PostgreSQL	Octets

Métrique	Description	S’applique à	Uunités
`VolumeBytesUsed`	Volume de stockage utilisé par votre cluster de bases de données Aurora. Cette valeur affecte le coût du cluster de bases de données Aurora (pour les informations de tarification, consultez la page de tarification Amazon RDS). Cette valeur ne reflète pas certaines allocations de stockage interne qui n’affectent pas la facturation du stockage. Pour Aurora MySQL, vous pouvez anticiper les out-of-space problèmes avec plus de précision en testant si la valeur `AuroraVolumeBytesLeftTotal` est proche de zéro au lieu de `VolumeBytesUsed` les comparer à la limite de stockage de 128 TiB. Pour les clusters qui sont des clones, la valeur de cette métrique dépend de la quantité de données ajoutées ou modifiées sur le clone. La métrique peut également augmenter ou diminuer lorsque le cluster	Aurora MySQL et Aurora PostgreSQL	Octets

Métrique	Description	S’applique à	Uunités
	d’origine est supprimé, ou lorsque de nouveaux clones sont ajoutés ou supprimés . Pour plus d’informations, consultez Suppression d’un volume de cluster source. Choisir une valeur `--period` faible n’a aucun sens, car Amazon RDS collecte ces métriques à intervalles réguliers et non en continu.		

Métrique	Description	S’applique à	Uunités
`VolumeReadIOPs`	Nombre d' I/O opérations de lecture facturées à partir d'un volume de cluster dans un intervalle de 5 minutes. Les opérations lues facturées sont calculées au niveau du volume de cluster, regroupées à partir de toutes les instances du cluster de bases de données Aurora, puis rapportées par intervalles de 5 minutes. La valeur est calculée en prenant la valeur de la métrique Read operations (Opérations en lecture) sur une période de 5 minutes. Vous pouvez déterminer la quantité d’opérations lues facturées par seconde en prenant la valeur de la métrique Billed read operations (Opérations en lecture facturées) et en la divisant par 300 secondes. Par exemple, si la métrique Billed read operations (Opérations en lecture facturée) renvoie 13 686, les opérations en lecture facturées par seconde s’élèvent à 45 (13 686 / 300 = 45,62).	Aurora MySQL et Aurora PostgreSQL	Compte par 5 minutes

Métrique	Description	S’applique à	Uunités
	Vous cumulez les opérations de lecture facturées pour les requêtes qui demandent des pages de base de données non présentes dans le cache des tampons et qui doivent être chargées depuis le stockage. Il se peut que vous constatiez des pics dans les opérations de lecture facturées, car les résultats des requêtes sont lus à partir du stockage, puis chargés dans le cache des tampons. Choisir une valeur `--period` faible n’a aucun sens, car Amazon RDS collecte ces métriques à intervalles réguliers et non en continu. **Tip** Si votre cluster Aurora MySQL utilise une requête parallèle, vous pouvez voir une augmentation des valeurs d’`VolumeReadIOPS` . Les requêtes parallèle		

Métrique	Description	S’applique à	Uunités
	s n’utilisent pas le pool de mémoires tampons. Ainsi, bien que les requêtes soient rapides, ce traitement optimisé peut entraîner une augmentation des opérations de lecture et des frais associés.		
`VolumeWriteIOPs`	Nombre d' I/O opération s d'écriture du disque sur le volume du cluster, indiqué à intervalles de 5 minutes. Pour une descripti on détaillée de la façon dont les opérations d’écriture facturées sont calculées , consultez `VolumeRea dIOPs` . Choisir une valeur `--period` faible n’a aucun sens, car Amazon RDS collecte ces métriques à intervalles réguliers et non en continu.	Aurora MySQL et Aurora PostgreSQL	Compte par 5 minutes

Métriques de niveau instance pour Amazon Aurora

Les CloudWatch métriques Amazon spécifiques aux instances suivantes s'appliquent à toutes les instances Aurora MySQL et Aurora PostgreSQL, sauf indication contraire.

Métrique	Description	S’applique à	Unités
`AbortedClients`	Nombre de connexions client qui n’ont pas été fermées correctement.	Aurora MySQL	Nombre
`ActiveTransactions`	Nombre moyen de transactions actives s’exécutant sur une instance de base de données Aurora par seconde. Par défaut, Aurora n’active pas cette métrique. Pour commencer à mesurer cette valeur, définissez `innodb_monitor_enable='all'` dans le groupe de paramètres de base de données pour une instance de base de données spécifique.	Aurora MySQL	Nombre par seconde
`ACUUtilization`	Valeur de la métrique `ServerlessDatabaseCapacity` divisée par le nombre maximal d’ACU du cluster de bases de données. Cette métrique n’est applicable que pour Aurora sans serveur v2.	Aurora MySQL et Aurora PostgreSQL	Pourcentage
`AuroraBinlogReplicaLag`	Durée pendant laquelle un cluster de bases de données de réplica de journal binaire exécuté sur Aurora MySQL est en retard par rapport à la source de réplication du	Principale pour Aurora MySQL	Secondes

Métrique	Description	S’applique à	Unités
	journal binaire. Un décalage signifie que la source génère des enregistrements plus rapidement que le réplica ne peut les appliquer. Cette métrique signale différentes valeurs en fonction de la version du moteur : Aurora MySQL version 2 Le `Seconds_Behind_Master` champ MySQL `SHOW SLAVE STATUS` Aurora MySQL version 3 `SHOW REPLICA STATUS` Vous pouvez utiliser cette métrique pour surveiller les erreurs et le décalage de réplica dans un cluster qui agit comme un réplica de journaux binaires. La valeur de la métrique indique ce qui suit : Une valeur élevée Le réplica est en retard par rapport à la source de réplication.		

Métrique	Description	S’applique à	Unités
	0 ou une valeur proche de **0** Le processus de réplica est actif et actuel. **-1** Aurora ne peut pas déterminer le décalage, ce qui peut se produire lors de la configuration du réplica ou lorsque le réplica est à l’état d’erreur. Étant donné que la réplication de journaux binaires ne se produit que sur l’instance de rédacteur du cluster, nous vous recommandons d’utiliser la version de cette métrique associée au rôle WRITER. Pour plus d’informations sur l’administration de la réplication, consultez Réplication de clusters de bases de données Amazon Aurora MySQL dans différentes Régions AWS. Pour plus d’informations sur la résolution des problèmes, consultez Problèmes de réplication Amazon Aurora MySQL.		

Métrique	Description	S’applique à	Unités
`AuroraDMLRejectedMasterFull`	Nombre de requêtes transférées qui sont rejetées, car la session est pleine sur l’instance de base de données d’enregistreur.	Principal pour Aurora MySQL version 2	Nombre
`AuroraDMLRejectedWriterFull`	Nombre de requêtes transférées qui sont rejetées, car la session est pleine sur l’instance de base de données d’enregistreur.	Principal pour Aurora MySQL version 3	Nombre
`AuroraEstimatedSharedMemoryBytes`	Estimation de la quantité de mémoire tampon partagée ou de mémoire de pool de tampons qui a été activement utilisée au cours du dernier intervalle d’interrogation configuré.	Aurora PostgreSQL	Octets
`AuroraMemoryHealthState`	Indique l’état de la mémoire. La valeur `0` équivaut à `NORMAL`. La valeur `10` équivaut à `RESERVED`, ce qui signifie que le serveur approche d’un niveau critique d’utilisation de la mémoire. Pour plus d’informations, consultez Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL.	Aurora MySQL versions 3.06.1 et ultérieures	Jauge

Métrique	Description	S’applique à	Unités
`AuroraMemoryNumDeclinedSqlTotal`	Le nombre incrémentiel de requêtes a diminué dans le cadre de l' out-of-memoryévitement (OOM). Pour de plus amples informations, veuillez consulter Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL.	Aurora MySQL versions 3.06.1 et ultérieures	Nombre
`AuroraMemoryNumKillConnTotal`	Nombre incrémentiel de connexions fermées afin d’éviter le manque de mémoire (OOM). Pour plus d’informations, consultez Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL.	Aurora MySQL versions 3.06.1 et ultérieures	Nombre
`AuroraMemoryNumKillQueryTotal`	Le nombre incrémentiel de requêtes a diminué afin d’éviter le manque de mémoire (OOM). Pour plus d’informations, consultez Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL.	Aurora MySQL versions 3.06.1 et ultérieures	Nombre

Métrique	Description	S'applique à	Unités
`AuroraMilliseconds SpentInOomRecovery`	Temps écoulé depuis que l'état de la mémoire est tombé en dessous de l'état normal. Pour plus d'informations, consultez Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL.	Aurora MySQL versions 3.08.0 et ultérieures	Milliseco ndes
`AuroraNumOomRecove rySuccessful`	Nombre de fois où l'état normal de la mémoire a été rétabli. Pour plus d'informations, consultez Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL.	Aurora MySQL versions 3.08.0 et ultérieures	Nombre
`AuroraNumOomRecove ryTriggered`	Nombre de fois où l'état de la mémoire est tombé en dessous de l'état normal. Pour plus d'informations, consultez Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL.	Aurora MySQL versions 3.08.0 et ultérieures	Nombre

Métrique	Description	S’applique à	Unités
`AuroraOptimizedReadsCacheHitRatio`	Pourcentage de demandes traitées par le cache Optimized Reads. La valeur est calculée à l’aide de la formule suivante : `orcache_blks_hit/ (orcache_blks_hit + storage_blks_read)` Lorsque `AuroraOptimizedReadsCacheHitRatio` atteint 100 %, cela signifie que toutes les pages ont été lues à partir du cache Optimized Reads. Lorsque `AuroraOptimizedReadsCacheHitRatio` est égal à `0`, cela signifie que toutes les pages ont été lues à partir du cache Optimized Reads.	Principale pour Aurora PostgreSQL	Pourcentage
`AuroraReplicaLag`	Pour un réplica Aurora, durée du retard lors de la réplication des mises à jour à partir de l’instance principale.	Réplica pour Aurora MySQL et Aurora PostgreSQL	Millisecondes

Métrique	Description	S'applique à	Unités
`AuroraReplicaLagMaximum`	La durée maximale du retard entre l'instance principale et chaque instance de base de données Aurora dans le cluster de bases de données. Lorsque des réplicas en lecture sont supprimés ou renommés, il peut y avoir un pic temporaire de retard de réplication lorsque l'ancienne ressource est soumise à un processus de recyclage. Pour obtenir une représentation précise du retard de réplication pendant cette période, nous vous recommandons de surveiller la métrique `AuroraReplicaLag` sur chaque instance de réplica en lecture.	Principale Aurora MySQL et Aurora PostgreSQL	Millisecondes
`AuroraReplicaLagMinimum`	La durée minimale du retard entre l'instance principale et chaque instance de base de données Aurora dans le cluster de bases de données.	Principale Aurora MySQL et Aurora PostgreSQL	Millisecondes

Métrique	Description	S’applique à	Unités
`AuroraSlowConnectionHandleCount`	Le nombre de connexions qui ont attendu au-delà deux secondes ou plus pour initier la négociation. Cette métrique s’applique uniquement à Aurora MySQL version 3.	Aurora MySQL	Nombre
`AuroraSlowHandshakeCount`	Le nombre de connexions qui ont mis 50 millisecondes ou plus pour terminer la négociation. Cette métrique s’applique uniquement à Aurora MySQL version 3.	Aurora MySQL	Nombre
`BacktrackWindowActual`	Différence entre la fenêtre de retour sur trace cible et la fenêtre de retour sur trace réelle.	Principale pour Aurora MySQL	Minutes
`BacktrackWindowAlert`	Nombre de fois où la fenêtre de retour sur trace réelle est plus petite que la fenêtre de retour sur trace cible pendant une période donnée.	Principale pour Aurora MySQL	Nombre
`BlockedTransactions`	Nombre moyen de transactions de la base de données bloquées par seconde.	Aurora MySQL	Nombre par seconde

Métrique	Description	S’applique à	Unités
`BufferCacheHitRatio`	Pourcentage de demandes traitées par le cache de tampons.	Aurora MySQL et Aurora PostgreSQL	Pourcentage
`CommitLatency`	Durée moyenne nécessaire au moteur et au stockage pour effectuer les opérations de validation.	Aurora MySQL et Aurora PostgreSQL	Millisecondes
`CommitThroughput`	Nombre moyen d’opérations de validation par seconde.	Aurora MySQL et Aurora PostgreSQL	Nombre par seconde
`ConnectionAttempts`	Le nombre de tentatives de connexion à une instance, qu’elles soient réussies ou non.	Aurora MySQL	Nombre

Métrique	Description	S’applique à	Unités
`CPUCreditBalance`	Nombre de crédits CPU accumulés par une instance, rapporté, rapporté par intervalles de 5 minutes. Cette métrique vous permet de déterminer combien de temps une instance de base de données peut fonctionn er en rafale au-delà de son niveau de performance de départ à un débit donné. Cette métrique s’appliqu e uniquement aux classes d’instance suivantes : • Aurora MySQL : `db.t2.small` , `db.t2.medium` , `db.t3` et `db.t4g` • Aurora PostgreSQL : `db.t3` et `db.t4g` **Note** Nous recommand ons d’utiliser les classes d’instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d’autres serveurs non dédiés à la production. Pour	Aurora MySQL et Aurora PostgreSQL	Nombre

Métrique	Description	S’applique à	Unités
	plus de détails sur les classes d’instance T, consultez Types de classes d’instance de base de données. Les crédits de lancement fonctionnent de la même manière sur Amazon RDS et sur Amazon EC2. Pour plus d’informations, consultez Launch credits (Crédits de lancement) dans le Guide de l’utilisateur d’Amazon Elastic Compute Cloud pour les instances Linux.		

Métrique	Description	S’applique à	Unités
`CPUCreditUsage`	Nombre de crédits CPU consommés au cours de la période spécifiée, rapporté par intervalles de 5 minutes. Cette métrique mesure le temps pendant lequel le matériel physique a CPUs été utilisé pour traiter les instructions par le virtuel CPUs alloué à l'instance de base de données. Cette métrique s’appliqu e uniquement aux classes d’instance suivantes : • Aurora MySQL : `db.t2.small` , `db.t2.medium` , `db.t3` et `db.t4g` • Aurora PostgreSQL : `db.t3` et `db.t4g` **Note** Nous recommand ons d’utiliser les classes d’instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d’autres serveurs non dédiés à la production. Pour	Aurora MySQL et Aurora PostgreSQL	Nombre

Métrique	Description	S’applique à	Unités
	plus de détails sur les classes d’instance T, consultez Types de classes d’instance de base de données.		
`CPUSurplusCreditBalance`	Nombre de crédits excédentaires dépensés par une instance illimitée lorsque la valeur `CPUCreditBalance` est nulle. La valeur de `CPUSurplusCreditBalance` est remboursée progressivement par les crédits UC gagnés. Si le nombre de crédits excédentaires dépasse le nombre maximum de crédits que l’instance peut gagner en 24 heures, les crédits excédentaires dépensés au-dessus du maximum génèrent des frais supplémentaires. Les métriques de crédits CPU sont disponibles toutes les 5 minutes uniquement.	Aurora MySQL et Aurora PostgreSQL	Crédits (minutes vCPU)

Métrique	Description	S’applique à	Unités
`CPUSurplusCreditsCharged`	Nombre de crédits excédentaires dépensés qui ne sont pas remboursés progressivement par les crédits UC gagnés et qui génèrent donc des frais supplémentaires. Les crédits excédentaires dépensés sont facturés lorsque l’une des situations suivantes se produit : • Les crédits excédentaires dépensés dépassent le nombre maximum de crédits que l’instance peut gagner sur une période de 24 heures. Les crédits excédentaires dépensés au-dessus de ce maximum sont facturés à la fin de l’heure. • L’instance est arrêtée ou résiliée. • L’instance bascule du mode `unlimited` au mode `standard`. Les métriques de crédits CPU sont disponibles toutes les 5 minutes uniquement.	Aurora MySQL et Aurora PostgreSQL	Crédits (minutes vCPU)

Métrique	Description	S’applique à	Unités
`CPUUtilization`	Pourcentage de l’UC utilisé par une instance de base de données Aurora.	Aurora MySQL et Aurora PostgreSQL	Pourcentage
`DatabaseConnections`	Nombre de connexions réseau client à l’instance de base de données. Le nombre de sessions de base de données peut être supérieur à la valeur de la métrique, car celle-ci n’inclut pas les éléments suivants : • Sessions qui n’ont plus de connexion réseau mais dont la base de données n’a pas été nettoyée • Sessions créées par le moteur de base de données à ses propres fins • Sessions créées par les capacités d’exécution parallèle du moteur de base de données • Sessions créées par le planificateur de tâches du moteur de base de données • Connexions Amazon Aurora	Aurora MySQL et Aurora PostgreSQL	Nombre

Métrique	Description	S’applique à	Unités
DDLLatency	Durée moyenne des requêtes, telles que les requêtes d’exemple, de création, alter et drop.	Aurora MySQL	Millisecondes
DDLThroughput	Nombre moyen de demandes DDL par seconde.	Aurora MySQL	Nombre par seconde
Deadlocks	Nombre moyen de blocages de la base de données par seconde.	Aurora MySQL et Aurora PostgreSQL	Nombre par seconde
DeleteLatency	Durée moyenne des opérations de suppression.	Aurora MySQL	Millisecondes
DeleteThroughput	Nombre moyen de requêtes de suppression par seconde.	Aurora MySQL	Nombre par seconde
DiskQueueDepth	Le nombre de read/write demandes en attente d'accès au disque.	Aurora MySQL et Aurora PostgreSQL	Nombre
DMLLatency	Durée moyenne des insertions, mises à jour et suppressions.	Aurora MySQL	Millisecondes
DMLThroughput	Nombre moyen d’insertions, de mises à jour et de suppressions par seconde.	Aurora MySQL	Nombre par seconde
EngineUptime	Temps d’exécution de l’instance.	Aurora MySQL et Aurora PostgreSQL	Secondes

Métrique	Description	S’applique à	Unités
`FreeableMemory`	Quantité de mémoire vive disponible. Pour les bases de données Aurora MySQL et Aurora PostgreSQL, cette métrique contient la valeur du champ `MemAvailable` de `/proc/meminfo` .	Aurora MySQL et Aurora PostgreSQL	Octets
`FreeEphemeralStorage`	La quantité de stockage éphémère NVMe disponible.	Aurora PostgreSQL	Octets

Métrique	Description	S'applique à	Unités
`FreeLocalStorage`	Quantité de stockage local disponible. Contrairement aux autres moteurs de base de données, pour les instances de base de données Aurora, cette métrique indique la quantité de stockage accessible à chaque instance de base de données. Cette valeur dépend de la classe d'instance de base de données (pour les informations de tarification, consultez la page de tarification Amazon RDS). Vous pouvez augmenter la quantité d'espace de stockage libre pour une instance en choisissant une classe d'instance de base de données plus grande pour votre instance. (Cela ne s'applique pas à Aurora Serverless v2).	Aurora MySQL et Aurora PostgreSQL	Octets
`InsertLatency`	Durée moyenne des opérations d'insertion.	Aurora MySQL	Millisecondes
`InsertThroughput`	Nombre moyen d'opérations d'insertion par seconde.	Aurora MySQL	Nombre par seconde
`LoginFailures`	Nombre moyen de tentatives de connexion ayant échoué par seconde.	Aurora MySQL	Nombre par seconde

Métrique	Description	S’applique à	Unités
`MaximumUsedTransactionIDs`	Âge de l’ID de transaction non vidée le plus ancien, en transactions. Si cette valeur atteint 2 146 483 648 (2^31 - 1 000 000), la base de données est forcée à passer en mode de lecture seule afin d’éviter le bouclage des ID de transaction. Pour plus d’informations, consultez Prévention des échecs de bouclage de l’ID de transaction dans la documentation PostgreSQL.	Aurora PostgreSQL	Nombre
`NetworkReceiveThroughput`	Quantité de débit réseau reçue des clients par chaque instance du cluster de bases de données Aurora. Ce débit n’inclut pas le trafic réseau entre les instances du cluster de bases de données Aurora et le volume de cluster.	Aurora MySQL et Aurora PostgreSQL	Octets par seconde (la console affiche le nombre de mégaoctets par seconde)
`NetworkThroughput`	Quantité de débit réseau reçue des clients et transmise à ces derniers par chaque instance du cluster de bases de données Aurora. Ce débit n’inclut pas le trafic réseau entre les instances du cluster de bases de données Aurora et le volume de cluster.	Aurora MySQL et Aurora PostgreSQL	Octets par seconde

Métrique	Description	S’applique à	Unités
`NetworkTransmitThroughput`	Quantité de débit réseau envoyée aux clients par chaque instance du cluster de bases de données Aurora. Ce débit n’inclut pas le trafic réseau entre les instances du cluster de bases de données et le volume de cluster.	Aurora MySQL et Aurora PostgreSQL	Octets par seconde (la console affiche le nombre de mégaoctets par seconde)
`NumBinaryLogFiles`	Nombre de fichiers binlog générés.	Aurora MySQL	Nombre
`OldestReplicationSlotLag`	Taille du retard du réplica le plus en retard en termes de données WAL reçues.	Aurora PostgreSQL	Octets
`PurgeBoundary`	Numéro de transaction jusqu’auquel la purge d’InnoDB est autorisée. Si cette métrique n’avance pas pendant de longues périodes, cela indique que la purge d’InnoDB est bloquée par des transactions de longue durée. Pour en savoir plus, vérifiez les transactions actives sur votre cluster de bases de données Aurora MySQL.	Aurora MySQL version 2, versions 2.11 et ultérieures Aurora MySQL version 3, versions 3.08 et ultérieures	Nombre

Métrique	Description	S’applique à	Unités
`PurgeFinishedPoint`	Numéro de transaction jusqu’auquel la purge d’InnoDB est effectuée. Cette métrique peut vous aider à examiner la rapidité de la purge d’InnoDB.	Aurora MySQL version 2, versions 2.11 et ultérieures Aurora MySQL version 3, versions 3.08 et ultérieures	Nombre
`Queries`	Nombre moyen de requêtes exécutées par seconde.	Aurora MySQL	Nombre par seconde
`RDSToAuroraPostgreSQLReplicaLag`	Durée du retard lors de la réplication des mises à jour depuis l’instance RDS PostgreSQL principale vers d’autres nœuds du cluster.	Réplica pour Aurora PostgreSQL	Secondes
`ReadIOPS`	Nombre moyen d' I/O opérations sur disque par seconde, les rapports étant lus et écrits séparément, à intervalles d'une minute.	Aurora MySQL et Aurora PostgreSQL	Nombre par seconde
`ReadIOPSEphemeralStorage`	Nombre moyen d' I/O opérations de lecture du disque vers le stockage éphémère NVMe .	Aurora PostgreSQL	Nombre par seconde
`ReadLatency`	Durée moyenne par I/O opération sur le disque.	Aurora MySQL et Aurora PostgreSQL	Secondes

Métrique	Description	S’applique à	Unités
`ReadLatencyEphemeralStorage`	Durée moyenne par I/O opération de lecture sur disque pour le stockage éphémère NVMe .	Aurora PostgreSQL	Secondes
`ReadThroughput`	Nombre moyen d’octets lus sur le disque par seconde.	Aurora MySQL et Aurora PostgreSQL	Octets par seconde
`ReadThroughputEphemeralStorage`	Nombre moyen d'octets lus sur le disque par seconde pour le stockage éphémère NVMe .	Aurora PostgreSQL	Octets par seconde
`ReplicationSlotDiskUsage`	Quantité d’espace disque consommée par les fichiers d’emplacement de réplication.	Aurora PostgreSQL	Octets
`ResultSetCacheHitRatio`	Pourcentage de demandes traitées par le cache du jeu de résultats.	Aurora MySQL version 2	Pourcentage
`RollbackSegmentHistoryListLength`	Journaux d’annulation qui enregistrent les transactions validées avec des enregistrements marqués pour la suppression. Ces enregistrements sont planifiés pour être traités par l’opération de purge InnoDB.	Aurora MySQL	Nombre
`RowLockTime`	Temps total passé à acquérir des verrous de ligne pour les tables InnoDB.	Aurora MySQL	Millisecondes

Métrique	Description	S’applique à	Unités
`SelectLatency`	Durée moyenne des opérations de sélection.	Aurora MySQL	Millisecondes
`SelectThroughput`	Nombre moyens de requêtes de sélection par seconde.	Aurora MySQL	Nombre par seconde
`ServerlessDatabaseCapacity`	Capacité actuelle d’un cluster de bases de données Aurora Serverless.	Aurora MySQL et Aurora PostgreSQL	Nombre
`StorageNetworkReceiveThroughput`	Quantité de débit réseau reçue du sous-système de stockage Aurora par chaque instance du cluster de bases de données.	Aurora MySQL et Aurora PostgreSQL	Octets par seconde
`StorageNetworkThroughput`	Quantité de débit réseau reçue du sous-système de stockage Aurora et envoyée à celui-ci par chaque instance du cluster de bases de données Aurora.	Aurora MySQL et Aurora PostgreSQL	Octets par seconde
`StorageNetworkTransmitThroughput`	Quantité de débit réseau envoyée au sous-système de stockage Aurora par chaque instance du cluster de bases de données Aurora.	Aurora MySQL et Aurora PostgreSQL	Octets par seconde
`SumBinaryLogSize`	Taille totale des fichiers binlog.	Aurora MySQL	Octets

Métrique	Description	S’applique à	Unités
`SwapUsage`	Quantité d’espace d’échange utilisé. Cette métrique n’est pas disponible pour les classes d’instance de base de données suivantes : • db.r3.*, db.r4.* et db.r7g.* (Aurora MySQL) • db.r7g.* (Aurora PostgreSQL)	Aurora MySQL et Aurora PostgreSQL	Octets
`TempStorageIOPS`	Nombre d’E/S par seconde réalisées en lecture et en écriture sur le stockage local attaché à l’instance de base de données. Cette métrique représente un nombre et est mesurée une fois par seconde. Cette métrique n’est applicable que pour Aurora sans serveur v2.	Aurora MySQL et Aurora PostgreSQL	Nombre par seconde
`TempStorageThroughput`	Volume de données transférés depuis et vers le stockage local associé à l’instance de base de données. Cette métrique représente des octets et est mesurée une fois par seconde. Cette métrique n’est applicable que pour Aurora sans serveur v2.	Aurora MySQL et Aurora PostgreSQL	Octets par seconde

Métrique	Description	S’applique à	Unités
`TransactionAgeMaximum`	Âge de la transaction en cours d’exécution active la plus ancienne.	Aurora MySQL version 3, versions 3.08 et ultérieures	Secondes
`TransactionLogsDiskUsage`	Quantité d’espace disque consommée par les journaux des transactions sur l’instance de base de données Aurora PostgreSQL. Cette métrique est générée uniquement lorsque Aurora PostgreSQL utilise une réplication logique ou. AWS Database Migration Servi Par défaut, Aurora PostgreSQL utilise des enregistrements de journal et non des journaux de transactions. Lorsque les journaux de transactions ne sont pas utilisés, la valeur de cette métrique est `-1`.	Principale pour Aurora PostgreSQL	Octets

Métrique	Description	S’applique à	Unités
`TruncateFinishedPoint`	Identifiant de transaction jusqu’auquel la troncature d’annulation est effectuée.	Aurora MySQL version 2, versions 2.11 et ultérieures Aurora MySQL version 3, versions 3.08 et ultérieures	Nombre
`UpdateLatency`	Le temps moyen pris pour les opérations de mise à jour.	Aurora MySQL	Millisecondes
`UpdateThroughput`	Nombre moyen de mises à jour par seconde.	Aurora MySQL	Nombre par seconde
`WriteIOPS`	Nombre d’enregistrements d’écriture de stockage Aurora générés par seconde. Il s’agit plus ou moins du nombre d’enregistrements de journaux générés par la base de données. Ils ne correspondent pas aux écritures de page de 8 Ko et ne correspondent pas aux paquets réseau envoyés.	Aurora MySQL et Aurora PostgreSQL	Nombre par seconde
`WriteIOPSEphemeralStorage`	Nombre moyen d' I/O opérations d'écriture sur disque dans le stockage éphémère NVMe .	Aurora PostgreSQL	Nombre par seconde

Métrique	Description	S’applique à	Unités
`WriteLatency`	Durée moyenne par I/O opération sur le disque.	Aurora MySQL et Aurora PostgreSQL	Secondes
`WriteLatencyEphemeralStorage`	Durée moyenne par I/O opération d'écriture sur disque pour le stockage éphémère NVMe .	Aurora PostgreSQL	Secondes
`WriteThroughput`	Nombre moyen d’octets écrits dans le stockage persistant chaque seconde.	Aurora MySQL et Aurora PostgreSQL	Octets par seconde
`WriteThroughputEphemeralStorage`	Nombre moyen d'octets écrits sur le disque par seconde pour le stockage éphémère NVMe .	Aurora PostgreSQL	Octets par seconde

Mesures CloudWatch d'utilisation d')

L'espace de `AWS/Usage` noms d'Amazon CloudWatch inclut les mesures d'utilisation au niveau du compte pour vos quotas de service Amazon RDS. CloudWatch collecte automatiquement les statistiques d'utilisation pour tous Régions AWS.

Pour plus d'informations, consultez les statistiques CloudWatch d'utilisation dans le guide de CloudWatch l'utilisateur Amazon. Pour plus d’informations sur les quotas, consultez Quotas et contraintes pour Amazon Aurora et Requesting a quota increase dans le Guide de l’utilisateur de Service Quotas.

Métrique	Description	Unités*
`AuthorizationsPerDBSecurityGroup`	Nombre de règles d’entrée par groupe de sécurité de base de données dans votre Compte AWS. La valeur utilisée est le plus grand nombre de règles d’entrée dans	Nombre

Métrique	Description	Unités*
	un groupe de sécurité de base de données du compte. Les autres groupes de sécurité de base de données du compte peuvent avoir un nombre inférieur de règles d'entrée.	
`CustomEndpointsPerDBCluster`	Nombre de points de terminaison personnalisés par cluster de bases de données dans votre Compte AWS. La valeur utilisée correspond au plus grand nombre de points de terminaison personnalisés dans un cluster de bases de données du compte. Les autres clusters de bases de données du compte peuvent avoir un nombre inférieur de points de terminaison personnalisés.	Nombre
`DBClusterParameterGroups`	Nombre de groupes de paramètres de cluster de bases de données dans votre Compte AWS. Le compte exclut les groupes de paramètres par défaut.	Nombre
`DBClusterRoles`	Le nombre de rôles Gestion des identités et des accès AWS (IAM) associés par cluster de base de données dans votre Compte AWS. La valeur utilisée est le plus grand nombre de rôles IAM associés pour un cluster de bases de données dans le compte. Les autres clusters de bases de données du compte peuvent avoir un nombre inférieur de rôles IAM personnalisés.	Nombre
`DBClusters`	Le nombre de clusters de bases de données Amazon Aurora dans votre Compte AWS.	Nombre
`DBInstanceRoles`	Le nombre de rôles Gestion des identités et des accès AWS (IAM) associés par instance de base de données dans votre Compte AWS. La valeur utilisée est le plus grand nombre de rôles IAM associés pour une instance de base de données dans le compte. Les autres instances de base de données du compte peuvent avoir un nombre inférieur de rôles IAM personnalisés.	Nombre

Métrique	Description	Unités*
DBInstances	Le nombre d'instances de base de données dans votre Compte AWS.	Nombre
DBParameterGroups	Le nombre de groupes de paramètres de base de données dans votre Compte AWS. Le compte exclut les groupes de paramètres de base de données par défaut.	Nombre
DBSubnetGroups	Le nombre de groupes de sous-réseaux de base de données dans votre Compte AWS. Le compte exclut le groupe de sous-réseau par défaut.	Nombre
EventSubscriptions	Nombre d'abonnements aux notifications d'événements dans votre Compte AWS.	Nombre
Integrations	Nombre d'intégrations zéro ETL à Amazon Redshift dans votre Compte AWS.	Nombre
ManualClusterSnapshots	Le nombre d'instantanés de cluster de bases de données créés manuellement dans votre Compte AWS. Le compte exclut les instantanés non valides.	Nombre
OptionGroups	Le nombre de groupes d'options dans votre Compte AWS. Le compte exclut les groupes d'options par défaut.	Nombre
Proxies	Le nombre de proxys RDS présents dans votre AWS compte.	Nombre
ReadReplicasPerMaster	Nombre de réplicas en lecture par instance de base de données dans votre compte. La valeur utilisée est le plus grand nombre de réplicas en lecture pour une instance de base de données dans le compte. Les autres instances de base de données du compte peuvent avoir un nombre inférieur de réplicas en lecture.	Nombre
ReservedDBInstances	Le nombre d'instances réservées de la base de données dans votre Compte AWS. Le compte exclut les instances retirées ou déclinées.	Nombre

Métrique	Description	Unités*
`SubnetsPerDBSubnetGroup`	Nombre de sous-réseaux par groupe de sous-réseaux de base de données dans votre Compte AWS. Le plus grand nombre de sous-réseaux pour un groupe de sous-réseaux de base de données dans le compte. Les autres groupes de sous-réseaux de base de données du compte peuvent avoir un nombre inférieur de sous-réseaux.	Nombre

> **Note**
>
> Amazon RDS ne publie pas d'unités destinées aux statistiques d' CloudWatchutilisation. Les unités n’apparaissent que dans la documentation.

CloudWatch Dimensions Amazon pour

Vous pouvez filtrer les données métriques Aurora en utilisant n'importe quelle dimension du tableau suivant.

Dimension	Filtre les données demandées pour . . .
`DBInstanceIdentifier`	Une instance de base de données spécifique.
`DBClusterIdentifier`	Un cluster de base de données Aurora spécifique.
`DBClusterIdentifier, Role`	Un cluster de base de données Aurora spécifique, en regroupant les métriques par rôle d'instance (WRITER/READER). Par exemple, vous pouvez regrouper des métriques pour toutes les instances READER qui appartiennent à un cluster.
`DbClusterIdentifier, EngineName`	Une combinaison spécifique de cluster de base de données et de nom de moteur Aurora. Par exemple, vous pouvez afficher les métriques `VolumeReadIOPs` pour le cluster `ams1` et le moteur `aurora`.

Dimension	Filtre les données demandées pour . . .
`DatabaseClass`	Toutes les instances d'une classe de base de données. Par exemple, vous pouvez regrouper des métriques pour toutes les instances qui appartiennent à la classe de base de données `db.r5.large` .
`EngineName`	Le nom du moteur identifié uniquement. Par exemple, vous pouvez regrouper des métriques pour toutes les instances ayant le nom de moteur `aurora-postgresql` .
`SourceRegion`	La région spécifiée uniquement. Par exemple, vous pouvez regrouper des métriques pour toutes les instances de base de données de la région `us-east-1` .

Disponibilité des métriques Aurora dans la console Amazon RDS

Les métriques fournies par Amazon Aurora ne sont pas toutes disponibles dans la console Amazon RDS. Vous pouvez consulter ces métriques à l'aide d'autres outils tels que l'AWS CLI et l'API CloudWatch. En outre, certaines métriques disponibles dans la console Amazon RDS s'affichent soit uniquement pour des classes d'instance spécifiques, soit avec des unités de mesure et des noms différents.

Rubriques

- Métriques Aurora disponibles dans la vue Dernière heure
- Métriques Aurora disponibles dans des cas spécifiques
- Métriques Aurora qui ne sont pas disponibles dans la console

Métriques Aurora disponibles dans la vue Dernière heure

Vous pouvez afficher un sous-ensemble de métriques Aurora catégorisées dans la vue par défaut Dernière heure de la console Amazon RDS. Le tableau suivant répertorie les catégories et les métriques associées qui s'affichent dans la console Amazon RDS for une instance Aurora.

Catégorie	Métriques
SQL	ActiveTransactions BlockedTransactions BufferCacheHitRatio CommitLatency CommitThroughput DatabaseConnections DDLLatency DDLThroughput Deadlocks DMLLatency DMLThroughput LoginFailures ResultSetCacheHitRatio SelectLatency SelectThroughput
Système	AuroraReplicaLag AuroraReplicaLagMaximum AuroraReplicaLagMinimum CPUCreditBalance CPUCreditUsage CPUUtilization

Catégorie	Métriques
	`FreeableMemory` `FreeLocalStorage` (Cela ne s'applique pas à Aurora Serverless v2). `NetworkReceiveThroughput`
Déploiement	`AuroraReplicaLag` `BufferCacheHitRatio` `ResultSetCacheHitRatio` `SelectThroughput`

Métriques Aurora disponibles dans des cas spécifiques

En outre, certaines métriques Aurora s'affichent soit uniquement pour des classes d'instance spécifiques, soit uniquement pour des instances de base de données, soit avec des unités de mesure et des noms différents :

- Les métriques `CPUCreditBalance` et `CPUCreditUsage` sont affichées uniquement pour les classes d'instance `db.t2` Aurora MySQL et pour les classes d'instance `db.t3` Aurora PostgreSQL.
- La liste suivante contient les métriques qui s'affichent avec des noms différents :

Métrique	Nom d'affichage
`AuroraReplicaLagMaximum`	Replica lag maximum
`AuroraReplicaLagMinimum`	Replica lag minimum
`DDLThroughput`	DDL
`NetworkReceiveThroughput`	Débit réseau
`VolumeBytesUsed`	[Facturé] Octets de volume utilisés

Métrique	Nom d’affichage
`VolumeReadIOPs`	[Facturé] IOPS en lecture pour le volume
`VolumeWriteIOPs`	[Facturé] IOPS en écriture pour le volume

- Les métriques suivantes s’appliquent à un cluster de bases de données Aurora entier, mais s’affichent uniquement dans la console Amazon RDS pour les instances de base de données d’un cluster de bases de données Aurora :
  - `VolumeBytesUsed`
  - `VolumeReadIOPs`
  - `VolumeWriteIOPs`
- Les métriques suivantes s’affichent en mégaoctets, et non en octets, dans la console Amazon RDS :
  - `FreeableMemory`
  - `FreeLocalStorage`
  - `NetworkReceiveThroughput`
  - `NetworkTransmitThroughput`
- Les métriques suivantes s’appliquent à un cluster de bases de données Aurora PostgreSQL avec Aurora Optimized Reads :
  - `AuroraOptimizedReadsCacheHitRatio`
  - `FreeEphemeralStorage`
  - `ReadIOPSEphemeralStorage`
  - `ReadLatencyEphemeralStorage`
  - `ReadThroughputEphemeralStorage`
  - `WriteIOPSEphemeralStorage`
  - `WriteLatencyEphemeralStorage`
  - `WriteThroughputEphemeralStorage`

Métriques Aurora qui ne sont pas disponibles dans la console

Les métriques Aurora suivantes ne sont pas disponibles dans la console Amazon RDS :

- `AuroraBinlogReplicaLag`

- DeleteLatency
- DeleteThroughput
- EngineUptime
- InsertLatency
- InsertThroughput
- NetworkThroughput
- Queries
- UpdateLatency
- UpdateThroughput

Métriques Amazon CloudWatch pour Analyse des performances d'Amazon RDS

Performance Insights publie automatiquement des métriques dans Amazon CloudWatch. Les mêmes données peuvent être interrogées à partir de Performance Insights, mais le fait que les métriques figurent dans CloudWatch facilite l'ajout d'alarmes CloudWatch, ainsi que l'ajout des métriques à des tableaux de bord CloudWatch existants.

Métrique	Description
DBLoad	Nombre de sessions actives pour la base de données. Vous souhaitez généralement obtenir les données relatives au nombre moyen de sessions actives. Dans Performance Insights, ces données sont interrogées sous la forme `db.load.avg` .
DBLoadCPU	Nombre de sessions actives dans lesquelles le type d'événement d'attente est CPU (UC). Dans Performance Insights, ces données sont interrogées sous la forme `db.load.avg` , filtrées par le type d'événement d'attente `CPU`.

Métrique	Description
DBLoadNonCPU	Nombre de sessions actives dans lesquelles le type d'événement d'attente n'est pas CPU (UC).
DBLoadRelativeToNumVCPUs	Rapport entre la charge de base de données et le nombre d'UC virtuelles de la base de données.

> **Note**
>
> Ces métriques ne sont publiées sur CloudWatch qu'en cas de chargement sur l'instance de bases de données.

Vous pouvez examiner ces métriques à l'aide de la console CloudWatch de l'AWS CLI ou de l'API CloudWatch. Vous pouvez également examiner d'autres indicateurs de métriques de Performance Insights à l'aide d'une fonction spéciale de mathématiques de métriques. Pour plus d'informations, consultez Interrogation d'autres métriques de compteur de Performance Insights dans CloudWatch.

Par exemple, vous pouvez obtenir les statistiques pour la métrique `DBLoad` en exécutant la commande get-metric-statistics.

```
aws cloudwatch get-metric-statistics \
    --region us-west-2 \
    --namespace AWS/RDS \
    --metric-name DBLoad  \
    --period 60 \
    --statistics Average \
    --start-time 1532035185 \
    --end-time 1532036185 \
    --dimensions Name=DBInstanceIdentifier,Value=db-loadtest-0
```

Cet exemple génère une sortie similaire à la suivante.

```
{
```

```
"Datapoints": [
{
"Timestamp": "2021-07-19T21:30:00Z",
"Unit": "None",
"Average": 2.1
},
{
"Timestamp": "2021-07-19T21:34:00Z",
"Unit": "None",
"Average": 1.7
},
{
"Timestamp": "2021-07-19T21:35:00Z",
"Unit": "None",
"Average": 2.8
},
{
"Timestamp": "2021-07-19T21:31:00Z",
"Unit": "None",
"Average": 1.5
},
{
"Timestamp": "2021-07-19T21:32:00Z",
"Unit": "None",
"Average": 1.8
},
{
"Timestamp": "2021-07-19T21:29:00Z",
"Unit": "None",
"Average": 3.0
},
{
"Timestamp": "2021-07-19T21:33:00Z",
"Unit": "None",
"Average": 2.4
}
],
"Label": "DBLoad"
}
```


Pour plus d’informations sur CloudWatch, consultez Qu’est-ce qu’Amazon CloudWatch ? dans le Guide de l’utilisateur Amazon CloudWatch.

Interrogation d’autres métriques de compteur de Performance Insights dans CloudWatch

> **Note**
>
> Si vous activez le mode avancé de Database Insights, Amazon RDS publie les métriques de compteur Performance Insights dans Amazon CloudWatch. Avec Database Insights, vous n’avez pas besoin d’utiliser la fonction de mathématiques de métriques `DB_PERF_INSIGHTS`. Vous pouvez utiliser le tableau de bord Database Insights CloudWatch pour rechercher, interroger et définir des alarmes pour les compteurs de Performance Insights.

Vous pouvez effectuer des requêtes, créer des alarmes et créer des graphiques sur les métriques RDS Performance Insights de CloudWatch. Vous pouvez accéder aux informations relatives à votre cluster de bases de données à l’aide de la fonction de mathématiques de métriques `DB_PERF_INSIGHTS` de CloudWatch. Cette fonction vous permet d’utiliser les métriques Performance Insights qui ne sont pas directement communiquées à CloudWatch pour créer une nouvelle série chronologique.

Vous pouvez utiliser la nouvelle fonction Mathématiques de métriques en cliquant sur le menu déroulant Ajouter des mathématiques dans l’écran Sélectionner une métrique de la console CloudWatch. Vous pouvez l’utiliser pour créer des alarmes et des graphiques sur les métriques Performance Insights ou sur des combinaisons de métriques CloudWatch et Performance Insights, y compris des alarmes haute résolution pour les métriques inférieures à la minute. Vous pouvez également utiliser la fonction par programmation en incluant l’expression de mathématiques de métriques dans une demande `get-metric-data`. Pour plus d’informations, consultez Syntaxe et fonctions mathématiques de métriques et Créer une alarme sur les métriques de compteur Performance Insights à partir d’une base de données AWS.

Métrique de compteur de Performance Insights

Les métriques de compteur sont des métriques de performances de base de données et de système d’exploitation dans le tableau de bord Performance Insights. Vous pouvez établir des corrélations entre ces informations et la charge de la base de données pour identifier et analyser les problèmes de performances. Vous devez ajouter une fonction statistique à la métrique pour obtenir les valeurs de la métrique. Par exemple, les fonctions prises en charge pour la métrique `os.memory.active` sont `.avg`, `.min`, `.max`, `.sum` et `.sample_count`.

Les métriques du compteur sont collectées une fois par minute. La collecte des métriques du système d'exploitation dépend de l'activation ou de la désactivation de la surveillance améliorée. Si la surveillance améliorée est désactivée, les métriques du système d'exploitation sont collectées une fois par minute. Si la surveillance améliorée est activée, les métriques du système d'exploitation sont collectées pour la période sélectionnée. Pour plus d'informations sur l'activation et la désactivation de la surveillance améliorée, consultez Activer et désactiver la surveillance améliorée.

Rubriques

- Compteurs de système d'exploitation Performance Insights
- Compteurs Performance Insights pour Aurora MySQL
- Compteurs Performance Insights pour Aurora PostgreSQL

Compteurs de système d'exploitation Performance Insights

Les compteurs des systèmes d'exploitation suivants, dont le préfixe est `os`, sont disponibles avec la fonctionnalité Analyse des performances pour Aurora PostgreSQL et Aurora MySQL.

Vous pouvez utiliser l'API `ListAvailableResourceMetrics` pour obtenir la liste des métriques de compteur disponibles pour votre instance de base de données. Pour plus d'informations, consultez ListAvailableResourceMetricsle guide de référence des API Amazon RDS Performance Insights.

Compteur	Type	Unité	Métrique	Description
Actif	Mémoire	Kilooctets	os.memory .active	Quantité de mémoire attribuée, en kilo-octets.
Tampons	Mémoire	Kilooctets	os.memory .buffers	Quantité de mémoire utilisée pour la mise en mémoire tampon des I/O demandes avant l'écriture sur le périphérique de

Compteur	Type	Unité	Métrique	Description
				stockage, en kilo-octets.
Mis en cache	Mémoire	Kilooctets	os.memory .cached	Quantité de mémoire utilisée pour la mise en cache des E/S basées sur le système de fichiers, en kilo-octets.
Cache de base de données	Mémoire	Octets	os.memory .db.cache	Quantité de mémoire utilisée pour le cache de pages par le processus de base de données, y compris tmpfs (shmem), en octets.
Taille de résident défini de base de données	Mémoire	Octets	os.memory.db. residentSetSize	Quantité de mémoire utilisée pour le cache anonyme et d'échange par le processus de base de données, sans inclure tmpfs (shmem), en octets.

Compteur	Type	Unité	Métrique	Description
Échange de base de données	Mémoire	Octets	os.memory .db.swap	Quantité de mémoire utilisée pour l'échange par le processus de base de données, en octets.
Non intègre	Mémoire	Kilooctets	os.memory.dirty	Quantité de pages mémoire de la RAM ayant été modifiées mais non écrites dans le bloc de données associé dans le stockage, en kilo-octets.
Free	Mémoire	Kilooctets	os.memory.free	Quantité de mémoire non attribuée, en kilo-octets.
Grandes pages gratuites	Mémoire	Pages	os.memory. hugePagesFree	Nombre de grandes pages gratuites. Les grandes pages sont une fonction du noyau Linux.
Grandes pages Rsvd	Mémoire	Pages	os.memory. hugePagesRsvd	Nombre de grandes pages dédiées.

Compteur	Type	Unité	Métrique	Description
Taille des grandes pages	Mémoire	Kilooctets	os.memory.hugePagesSize	Taille de chaque unité de grandes pages, en kilo-octets.
Grandes pages excéd	Mémoire	Pages	os.memory.hugePagesSurp	Nombre de grandes pages excédentaires disponibles par rapport au nombre total.
Total de grandes pages	Mémoire	Pages	os.memory.hugePagesTotal	Nombre total de grandes pages.
Inactif	Mémoire	Kilooctets	os.memory.inactive	Quantité de pages mémoire moins fréquemment utilisées, en kilo-octets.
Mappé	Mémoire	Kilooctets	os.memory.mapped	Quantité totale de contenu du système de fichiers mappé en mémoire dans un espace d'adressage de processus, en kilo-octets.

Compteur	Type	Unité	Métrique	Description
Nombre d’arrêts de mémoire insuffisante	Mémoire	Tuées	os.memory.outOfMemoryKillCount	Nombre d’arrêts de mémoire insuffisante survenus au cours du dernier intervalle de collecte.
Tables de pages	Mémoire	Kilooctets	os.memory.pageTables	Quantité de mémoire utilisée par les tables de page, en kilo-octets.
Section	Mémoire	Kilooctets	os.memory.slab	Quantité de structures de données noyau réutilisables, en kilo-octets.
Total	Mémoire	Kilooctets	os.memory.total	Quantité totale de mémoire, en kilo-octets.
Écriture différée	Mémoire	Kilooctets	os.memory.writeback	Quantité de pages de modification dans la RAM encore écrites dans le stockage de sauvegarde, en kilo-octets.

Compteur	Type	Unité	Métrique	Description
Invité	Utilisation de l'UC	Pourcentage	os.cpuUtilization.guest	Pourcentage de l'UC en cours d'utilisation par les programmes invités.
Inactif	Utilisation de l'UC	Pourcentage	os.cpuUtilization.idle	Pourcentage de l'UC inactive.
Irq	Utilisation de l'UC	Pourcentage	os.cpuUtilization.irq	Pourcentage de l'UC en cours d'utilisation par les interruptions logicielles.
Nice	Utilisation de l'UC	Pourcentage	os.cpuUtilization.nice	Pourcentage de l'UC en cours d'utilisation par des programmes s'exécutant avec la priorité la plus faible.
Steal	Utilisation de l'UC	Pourcentage	os.cpuUtilization.steal	Pourcentage de l'UC en cours d'utilisation par d'autres machines virtuelles.
Système	Utilisation de l'UC	Pourcentage	os.cpuUtilization.system	Pourcentage de l'UC en cours d'utilisation par le noyau.

Compteur	Type	Unité	Métrique	Description
Total	Utilisation de l'UC	Pourcentage	os.cpuUtilization.total	Pourcentage total de l'UC en cours d'utilisation. Cette valeur inclut la valeur Nice.
Utilisateur	Utilisation de l'UC	Pourcentage	os.cpuUtilization.user	Pourcentage de l'UC en cours d'utilisation par des programmes utilisateurs.
Attente	Utilisation de l'UC	Pourcentage	os.cpuUtilization.wait	Pourcentage de CPU inutilisé pendant l'attente de I/O l'accès.
Rx d'octets de stockage Aurora de stockage Aurora	E/S du disque	Octets par seconde	OS.Diskio.AuroraStorage.auroraStorageBytesRx	Nombre d'octets reçus du stockage Aurora par seconde.
Tx d'octets de stockage Aurora de stockage Aurora	E/S du disque	Octets par seconde	OS.Diskio.AuroraStorage.auroraStorageBytesTx	Nombre d'octets chargés dans le stockage Aurora par seconde.
Profondeur de la file d'attente du disque de stockage Aurora	E/S du disque	Requêtes	OS.Diskio.AuroraStorage.diskQueueDepth	Longueur de la file d'attente du disque de stockage Aurora.

Compteur	Type	Unité	Métrique	Description
Aurora Storage Lire IOs PS	E/S du disque	Demandes par seconde	OS.Diskio.AuroraStorage.Lire PS IOs	Nombre d'opérations de lecture par seconde.
Latence de lecture de stockage Aurora	E/S du disque	Millisecondes	os.diskIO.auroraStorage.readLatency	Latence moyenne d'une I/O demande de lecture vers le stockage Aurora, en millisecondes.
Débit de lecture du stockage Aurora	E/S du disque	Octets par seconde	os.diskIO.auroraStorage.readThroughput	Quantité de débit réseau utilisée par les demandes adressées au cluster DB, en octets par seconde.
Système de stockage Aurora Write IOs PS	E/S du disque	Demandes par seconde	OS.Diskio.AuroraStorage.Write PS IOs	Nombre d'opérations d'écriture par seconde.
Latence d'écriture de stockage Aurora	E/S du disque	Millisecondes	os.diskIO.auroraStorage.writeLatency	Latence moyenne d'une I/O demande d'écriture vers le stockage Aurora, en millisecondes.

Compteur	Type	Unité	Métrique	Description
Débit d’écriture de stockage Aurora	E/S du disque	Octets par seconde	os.diskIO.auroraStorage.writeThroughput	Quantité de débit réseau utilisée par les réponses du cluster DB, en octets par seconde.
Longueur file d’attente moyenne Rdstemp	E/S du disque	Requêtes	OS.Diskio.RDSTEMP.avgQueueLen	Le nombre de demandes en attente dans la file d'attente de l'I/O appareil.
Taille demande moyenne Rdstemp	E/S du disque	Requêtes	OS.Diskio.RDSTEMP.avgReqSz	Le nombre de demandes en attente dans la file d'attente de l'I/O appareil.
Rdstemp en attente	E/S du disque	Millisecondes	os.diskIO.rdstemp.await	Nombre de millisecondes requises pour répondre aux requêtes, y compris le temps d’attente et le temps de service.
Restemp Read OS IOs	E/S du disque	Requêtes	os.diskio.rdstemp.Lire PS IOs	Nombre d’opérations de lecture par seconde.
Ko de lecture Rdstemp	E/S du disque	Kilooctets	os.diskIO.rdstemp.readKb	Nombre total de kilo-octets lus.

Compteur	Type	Unité	Métrique	Description
PS de Ko de lecture Rdstemp	E/S du disque	Kilooctets par seconde	os.diskIO .rdstemp. readKbPS	Nombre de kilo-octets lus par seconde.
PS Rrqm Rdstemp	E/S du disque	Demandes par seconde	os.diskIO .rdstemp.rrqmPS	Nombre de requêtes de lecture fusionnées mises en file d'attente par seconde.
TPS Rdstemp	E/S du disque	Transactions par seconde	os.diskIO .rdstemp.tps	Le nombre de I/O transactions par seconde.
Utilitaire Rdstemp	E/S du disque	Pourcentage	os.diskIO .rdstemp.util	Pourcentage de temps UC pendant lequel les requêtes ont été émises.
Système d'exploitation Restemp Write IOs	E/S du disque	Demandes par seconde	os.diskio .RDStemp.Write PS IOs	Nombre d'opérations d'écriture par seconde.
Ko d'écriture Rdstemp	E/S du disque	Kilooctets	os.diskIO .rdstemp.writeKb	Nombre total de kilo-octets écrits.
PS Ko d'écriture Rdstemp	E/S du disque	Kilooctets par seconde	os.diskIO .rdstemp. writeKbPS	Nombre de kilo-octets écrits par seconde.

Compteur	Type	Unité	Métrique	Description
PS Wrqm Rdstemp	E/S du disque	Demandes par seconde	os.diskIO .rdstemp. wrqmPS	Nombre de requêtes d'écriture fusionnées mises en file d'attente par seconde.
Bloqué	Tâches	Tâches	os.tasks.blocked	Nombre de tâches bloquées.
En cours d'exécution	Tâches	Tâches	os.tasks.running	Nombre de tâches en cours d'exécution.
En veille	Tâches	Tâches	os.tasks.sleeping	Nombre de tâches en veille.
Arrêté(e)	Tâches	Tâches	os.tasks.stopped	Nombre de tâches arrêtées.
Total	Tâches	Tâches	os.tasks.total	Nombre total de tâches.
Zombie	Tâches	Tâches	os.tasks.zombie	Nombre de tâches enfant inactives avec une tâche parent active.
Un	Minute moyenne de charge	Processus	os. loadAvera geMinute.un	Nombre de processus demandant du temps UC au cours de la dernière minute.

Compteur	Type	Unité	Métrique	Description
Quinze	Minute moyenne de charge	Processus	os. loadAvera geMinute.quinze	Nombre de processus demandant du temps UC au cours des 15 dernières minutes.
Cinq	Minute moyenne de charge	Processus	os. loadAvera geMinute.cinq	Nombre de processus demandant du temps UC au cours des 5 dernières minutes.
Mis en cache	Swap	Kilooctets	os.swap.cached	Quantité de mémoire d'échange, en kilo-octets, utilisée en tant que mémoire cache.
Free	Swap	Kilooctets	os.swap.free	Quantité de mémoire d'échange libre, en kilo-octets.
Entrée	Swap	Kilooctets	os.swap.in	Quantité de mémoire, en kilo-octets, échangée depuis le disque.

Compteur	Type	Unité	Métrique	Description
Sortie	Swap	Kilooctets	os.swap.out	Quantité de mémoire, en kilo-octets, échangée vers le disque.
Total	Swap	Kilooctets	os.swap.total	Quantité totale de mémoire d'échange disponible, en kilo-octets.
Nombre maximum de fichiers	Système de fichiers	Fichiers	os.fileSys.maxFiles	Nombre maximal de fichiers pouvant être créés pour le système de fichiers sur tous les volumes de stockage.
Fichiers utilisés	Système de fichiers	Fichiers	os.fileSys.usedFiles	Nombre de fichiers du système de fichiers sur tous les volumes de stockage.
Pourcentage de fichiers utilisés	Système de fichiers	Fichiers	Système d'exploitation .FileSys.usedFilePercent	Pourcentage de fichiers disponibles utilisés sur tous les volumes de stockage.

Compteur	Type	Unité	Métrique	Description
Pourcentage utilisé	Système de fichiers	Pourcentage	os.fileSys.usedPercent	Pourcentage d'espace disque du système de fichiers utilisé sur tous les volumes de stockage.
Utilisé	Système de fichiers	Kilooctets	os.fileSys.used	Quantité d'espace disque utilisée par les fichiers du système de fichiers sur tous les volumes de stockage, en kilo-octets.
Total	Système de fichiers	Kilooctets	os.fileSys.total	Espace disque total disponible pour le système de fichiers sur tous les volumes de stockage, en kilo-octets.
Nombre maximum de fichiers	Système de fichiers	Fichiers	Système d'exploitation .FileSys.<volumeName>Fichiers .max	Nombre maximal de fichiers pouvant être créés pour le volume de stockage.

Compteur	Type	Unité	Métrique	Description
Fichiers utilisés	Système de fichiers	Fichiers	Système d'exploitation .FileSys.<volumeName>Fichiers .Used	Nombre de fichiers dans le volume de stockage.
Pourcentage de fichiers utilisés	Système de fichiers	Fichiers	Système d'exploitation .FileSys.<volumeName>.usedFilePercent	Pourcentage de fichiers disponibles utilisés dans le volume de stockage.
Pourcentage utilisé	Système de fichiers	Pourcentage	Système d'exploitation .FileSys.<volumeName>.Pourcentage utilisé	Pourcentage de l'espace disque du volume de stockage utilisé.
Utilisé	Système de fichiers	Kilooctets	Système d'exploitation .FileSys.<volumeName>.utilisé	Quantité d'espace disque utilisée par les fichiers du volume de stockage, en kilo-octets.
Total	Système de fichiers	Kilooctets	Système d'exploitation .FileSys.<volumeName>.total	Espace disque total disponible dans le volume de stockage, en kilo-octets.

Compteur	Type	Unité	Métrique	Description
Rx	Réseau	Octets par seconde	os.network.rx	Nombre d’octets reçus par seconde.
Tx	Réseau	Octets par seconde	os.network.tx	Nombre d’octets téléchargés par seconde.
Utilisation d’ACU	Général	Pourcentage	os.general.acuUtilization	Pourcentage de la capacité actuelle par rapport à la capacité maximale configurée.
ACU configurée max.	Général	ACUs	os.general.maxConfiguredAcu	Capacité maximale configurée par l'utilisateur, en unités de capacité Aurora (ACUs).
ACU configurée min.	Général	ACUs	os.general. minConfiguredAcu	La capacité minimale configurée par l'utilisateur, en ACUs.
Num VCPUs	Général	v CPUs	os.general.numVCPUs	Le nombre de virtual CPUs (vCPUs) pour l'instance de base de données.

Compteur	Type	Unité	Métrique	Description
Capacité de base de données sans serveur	Général	ACUs	os.general. serverlessDatabaseCapacity	La capacité actuelle de l'instance, en ACUs.

Compteurs Performance Insights pour Aurora MySQL

Les compteurs de base de données suivants sont disponibles avec Performance Insights pour Aurora MySQL.

Rubriques

- Compteurs natifs pour Aurora MySQL
- Compteurs non natifs pour Aurora MySQL

Compteurs natifs pour Aurora MySQL

Les métriques natives sont définies par le moteur de base de données et non par Amazon Aurora. Vous trouverez les définitions de ces métriques natives dans Variables d'état de serveur, dans la documentation sur MySQL.

Compteur	Type	Unité	Métrique
Com_analyze	SQL	Requêtes par seconde	db.SQL.Com_analyze
Com_optimize	SQL	Requêtes par seconde	db.SQL.Com_optimize
Com_select	SQL	Requêtes par seconde	db.SQL.Com_select

Compteur	Type	Unité	Métrique
Innodb_rows_deleted	SQL	Lignes par seconde	db.SQL.Innodb_rows_deleted
Innodb_rows_inserted	SQL	Lignes par seconde	db.SQL.Innodb_rows_inserted
Innodb_rows_read	SQL	Lignes par seconde	db.SQL.Innodb_rows_read
Innodb_rows_updated	SQL	Lignes par seconde	db.SQL.Innodb_rows_updated
Requêtes	SQL	Requêtes par seconde	db.SQL.Queries
Questions	SQL	Requêtes par seconde	db.SQL.Questions
Select_full_join	SQL	Requêtes par seconde	db.SQL.Select_full_join
Select_full_range_join	SQL	Requêtes par seconde	db.SQL.Select_full_range_join
Select_range	SQL	Requêtes par seconde	db.SQL.Select_range
Select_range_check	SQL	Requêtes par seconde	db.SQL.Select_range_check

Compteur	Type	Unité	Métrique
Select_scan	SQL	Requêtes par seconde	db.SQL.Select_scan
Slow_queries	SQL	Requêtes par seconde	db.SQL.Slow_queries
Sort_merge_passes	SQL	Requêtes par seconde	db.SQL.Sort_merge_passes
Sort_range	SQL	Requêtes par seconde	db.SQL.Sort_range
Sort_rows	SQL	Requêtes par seconde	db.SQL.Sort_rows
Sort_scan	SQL	Requêtes par seconde	db.SQL.Sort_scan
Total_query_time	SQL	Millisecondes	db.SQL.Total_query_time
Table_locks_immediate	Locks	Demandes par seconde	db.Locks.Table_locks_immediate
Table_locks_waited	Locks	Demandes par seconde	db.Locks.Table_locks_waited

Compteur	Type	Unité	Métrique
Innodb_row_lock_time	Locks	Milliseconds (moyenne)	db.Locks.Innodb_row_lock_time
Aborted_clients	Users	Connexions	db.Users.Aborted_clients
Aborted_connects	Users	Connexions	db.Users.Aborted_connects
Connexions	Users	Connexions	db.Users.Connections
External_threads_connected	Users	Connexions	db.Users.External_threads_connected
max_connections	Users	Connexions	db.Users.max_connections
Threads_connected	Users	Connexions	db.Users.Threads_connected
Threads_created	Users	Connexions	db.Users.Threads_created
Threads_running	Users	Connexions	db.Users.Threads_running
Created_tmp_disk_tables	Temp	Tables par seconde	db.Temp.Created_tmp_disk_tables
Created_tmp_tables	Temp	Tables par seconde	db.Temp.Created_tmp_tables
Innodb_buffer_pool_pages_data	Cache	Pages	db.Cache.Innodb_buffer_pool_pages_data

Compteur	Type	Unité	Métrique
Innodb_buffer_pool_pages_total	Cache	Pages	db.Cache.Innodb_buffer_pool_pages_total
Innodb_buffer_pool_read_requests	Cache	Pages par seconde	db.Cache.Innodb_buffer_pool_read_requests
Innodb_buffer_pool_reads	Cache	Pages par seconde	db.Cache.Innodb_buffer_pool_reads
Opened_tables	Cache	Tables	db.Cache.Opened_tables
Opened_table_definitions	Cache	Tables	db.Cache.Opened_table_definitions
Qcache_hits	Cache	Requêtes	db.Cache.Qcache_hits

Compteurs non natifs pour Aurora MySQL

Les métriques de compteur non natif sont des compteurs définis par Amazon RDS. Une métrique non native peut être obtenue avec une requête spécifique. Il peut également s'agir d'une métrique dérivée, pour laquelle deux compteurs natifs ou plus sont utilisés dans les calculs de rapport, de taux d'accès ou de latences.

Compteur	Type	Unité	Métrique	Description	Définition
active_transactions	Transa ons	db.Trar ctions.a tive_tra saction	Nombre total de transacti ons actives.	`SELECT COUNT(1) AS active_tr ansactions FROM INFOR MATION_SC HEMA.INNO DB_TRX`	
innodb_buffer_pool_hit_rate	Cache	db.Cac innoDB ffer_poo _hit_rat	Pourcenta ge de lectures pouvant	`100 * innodb_bu ffer_pool _read_req uests /`	

Compteur	Type	Unité	Métrique	Description	Définition
			être réalisées par InnoDB à partir du groupe de mémoires tampons.	(innodb_buffer_pool_read_requests + innodb_buffer_pool_reads)	
innodb_buffer_pool_hits	Cache	Pages par second	db.Cache.innoDB_buffer_pool_hits	Nombre de lectures pouvant être réalisées par InnoDB à partir du groupe de mémoires tampons.	innodb_buffer_pool_read_requests - innodb_buffer_pool_reads

Compteur	Type	Unité	Métrique	Description	Définition
innodb_buffer_pool_usage	Cache	Pourcentage	db.Cache.innoDB_buffer_pool_usage	Pourcentage du groupe de mémoires tampons InnoDB contenant des données (pages). **Note** Cette valeur peut varier lors de l'utilisation de tables compressées. Pour plus d'informations, consultez les informations relatives à `Innodb_buffer_pool_pages_data` et `Innodb_buffer_pool_pages_total` dans Variables d'état de serveur, dans la documenta	`Innodb_buffer_pool_pages_data / Innodb_buffer_pool_pages_total * 100.0`

Compteur	Type	Unité	Métrique	Description	Définition
				tion sur MySQL.	
innodb_deadlocks	Locks	db.Lock innodb_ adlocks	Nombre total de blocages.	`SELECT COUNT AS innodb_de adlocks FROM INFORMATION_SC HEMA.INNO DB_METRICS WHERE NAME= 'lock_dea dlocks'`	
innodb_lo ck_timeouts	Locks	db.Lock innodb_ ck_time ts	Nombre total de blocages ayant expiré.	`SELECT COUNT AS innodb_lo ck_timeouts FROM INFOR MATION_SC HEMA.INNO DB_METRICS WHERE NAME= 'lock_tim eouts'`	
innodb_ro w_lock_waits	Locks	db.Lock innodb_ w_lock_ its	Nombre total de verrouill ages de ligne ayant entraîné une attente.	`SELECT COUNT AS innodb_ro w_lock_waits FROM INFOR MATION_SC HEMA.INNO DB_METRICS WHERE NAME ='lock_ro w_lock_waits'`	

Compteur	Type	Unité	Métrique	Description	Définition
innodb_rows_changed	SQL	db.SQL.innodb_rows_changed	Nombre total d'opérations de ligne InnoDB	db.SQL.Innodb_rows_inserted + db.SQL.Innodb_rows_deleted + db.SQL.Innodb_rows_updated	
query_cache_hit_rate	Cache	Pourcentage	db.Cache.query_cache_hit_rate	Taux d'accès au cache (de requête) de l'ensemble de résultats MySQL.	Qcache_hits / (QCache_hits + Com_select) * 100
temp_disk_tables_percent	Temp	db.Temp.temp_disk_tables_percent	Pourcentage de tables temporaires créées sur le disque par le serveur lors de l'exécution d'instructions.	(db.Temp.Created_tmp_disk_tables / db.Temp.Created_tmp_tables) * 100	

Compteur	Type	Unité	Métrique	Description	Définition
trx_rseg_history_len	Transactions	Aucune	db.Transactions.trx_rseg_history_len	Liste des pages du journal des annulations pour les transactions validées qui est gérée par le système de transactions InnoDB pour implémenter le contrôle de simultanéité multiversion. Pour plus d'informations sur les détails des enregistrements du journal d'annulation, consultez https://dev.mysql.com/doc/refman/8.0/en/innodb-multi-versioning.html dans la documentation MySQL.	`SELECT COUNT AS trx_rseg_history_len FROM INFORMATION_SCHEMA.INNODB_METRICS WHERE NAME='trx_rseg_history_len'`

Compteurs Performance Insights pour Aurora PostgreSQL

Les compteurs de base de données suivants sont disponibles avec Performance Insights pour Aurora PostgreSQL.

Rubriques

- Compteurs natifs pour Aurora PostgreSQL
- Compteurs non natifs pour Aurora PostgreSQL

Compteurs natifs pour Aurora PostgreSQL

Les métriques natives sont définies par le moteur de base de données et non par Amazon Aurora. La section Viewing Statistics de la documentation PostgreSQL fournit les définitions de ces métriques natives.

Compteur	Type	Unité	Métrique
tup_deleted	SQL	Tuples par seconde	db.SQL.tup_deleted
tup_fetched	SQL	Tuples par seconde	db.SQL.tup_fetched
tup_inserted	SQL	Tuples par seconde	db.SQL.tup_inserted
tup_returned	SQL	Tuples par seconde	db.SQL.tup_returned
tup_updated	SQL	Tuples par seconde	db.SQL.tup_updated
blks_hit	Cache	Blocs par seconde	db.Cache.blks_hit
buffers_alloc	Cache	Blocs par seconde	db.Cache.buffers_alloc
buffers_checkpoint	Checkpoint	Blocs par seconde	db.Checkpoint.buffers_checkpoint
checkpoints_req	Checkpoint	Points de contrôle par minute	db.Checkpoint.checkpoints_req
checkpoint_sync_time	Checkpoint	Millisecondes par point de contrôle	db.Checkpoint.checkpoint_sync_time
checkpoints_timed	Checkpoint	Points de contrôle par minute	db.Checkpoint.checkpoints_timed
checkpoint_write_time	Checkpoint	Millisecondes par point de contrôle	db.Checkpoint.checkpoint_write_time
maxwritten_clean	Checkpoint	Arrêts de nettoyage Bgwriter par minute	db.Checkpoint.maxwritten_clean

Compteur	Type	Unité	Métrique
deadlocks	Concurrency	Blocages par minute	db.Concurrency.deadlocks
blk_read_time	I/O	Millisecondes	db.IO.blk_read_time
blks_read	I/O	Blocs par seconde	db.IO.blks_read
buffers_backend	I/O	Blocs par seconde	db.IO.buffers_backend
buffers_backend_fsync	I/O	Blocs par seconde	db.IO.buffers_backend_fsync
buffers_clean	I/O	Blocs par seconde	db.IO.buffers_clean
temp_bytes	Temp	Octets par seconde	db.Temp.temp_bytes
temp_files	Temp	Fichiers par minute	db.Temp.temp_files
xact_commit	Transactions	Validations par seconde	db.Transactions.xact_commit
xact_rollback	Transactions	Restaurations par seconde	db.Transactions.xact_rollback
numbackends	User	Connexions	db.User.numbackends
archived_count	WAL	Fichiers par minute	db.WAL.archived_count

Compteurs non natifs pour Aurora PostgreSQL

Les métriques de compteur non natif sont des compteurs définis par Amazon Aurora. Une métrique non native peut être obtenue avec une requête spécifique. Il peut également s’agir d’une métrique dérivée, pour laquelle deux compteurs natifs ou plus sont utilisés dans les calculs de rapport, de taux d’accès ou de latences.

Compteur	Type	Unité	Métrique	Description	Définition
checkpoint_sync_latency	Checkpoint	Millisecondes	db.Checkpoint.checkpoint_sync_latency	Durée totale consacrée à la partie du traitement de point de contrôle où les fichiers sont synchronisés sur le disque.	`checkpoint_sync_time / (checkpoints_timed + checkpoints_req)`
checkpoint_write_latency	Checkpoint	Millisecondes	db.Checkpoint.checkpoint_write_latency	Durée totale consacrée à la partie du traitement de point de contrôle où les fichiers sont écrits sur le disque.	`checkpoint_write_time / (checkpoints_timed + checkpoints_req)`
local_blks_read	I/O	Blocs	db.IO.local_blks_read	Nombre total de blocs locaux lus.	Ne s'applique pas
local_blk_read_time	I/O	Millisecondes	db.IO.local_blk_read_time	Si `track_io_timing` est activé, le temps total passé à lire des blocs de fichiers de données locaux est enregistré en millisecondes, sinon la valeur est nulle. Pour plus d'informations, consultez track_io_timing.	Ne s'applique pas
num_blocked_sessions	Locks	db.Locks.num_blocked_sessions	Nombre de sessions bloquées.	–	

Compteur	Type	Unité	Métrique	Description	Définition
orcache_blks_hit	I/O	Requêtes	db.IO.orcache_blks_hit	Nombre total de blocs partagés accessibles à partir du cache Optimized Reads.	Ne s'applique pas
orcache_blk_read_time	I/O	Millisecondes	db.IO.orcache_blk_read_time	Si `track_io_timing` est activé, le temps total passé à lire des blocs de fichiers de données à partir du cache Optimized Reads est enregistré en millisecondes, sinon la valeur est nulle. Pour plus d'informations, consultez track_io_timing.	Ne s'applique pas
read_latency	I/O	Millisecondes	db.IO.read_latency	Durée consacrée à la lecture des blocs de fichier de données par les systèmes dorsaux dans cette instance.	`blk_read_time / blks_read`
storage_blks_read	I/O	Blocs	db.IO.storage_blks_read	Nombre total de blocs partagés lus à partir du stockage Aurora.	Ne s'applique pas

Compteur	Type	Unité	Métrique	Description	Définition
storage_blk_read_time	I/O	Millisecondes	db.IO.storage_blk_read_time	Si `track_io_timing` est activé, le temps total passé à lire des blocs de fichiers de données à partir du stockage Aurora est enregistré en millisecondes, sinon la valeur est nulle. Pour plus d'informations, consultez track_io_timing.	Ne s'applique pas
num_blocked_sessions	Locks	db.Locks.num_blocked_sessions	Nombre de sessions bloquées.	–	
active_count	État	Séances	db.state.active_count	Nombre de sessions dans l'état `active`.	Ne s'applique pas
idle_count	État	Séances	db.state.idle_count	Nombre de sessions dans l'état `idle`.	Ne s'applique pas
idle_in_transaction_aborted_count	État	Séances	db.state.idle_in_transaction_aborted_count	Nombre de sessions dans l'état `idle in transaction (aborted)` .	Ne s'applique pas

Compteur	Type	Unité	Métrique	Description	Définition
idle_in_transaction_count	État	Séances	db.state.idle_in_transaction_count	Nombre de sessions dans l'état `idle in transaction` .	Ne s'applique pas
idle_in_transaction_max_time	État	Secondes	db.state.idle_in_transaction_max_time	Durée de la transaction la plus longue dans l'état `idle in transaction` , en secondes.	Ne s'applique pas
logical_reads	SQL	Blocs	db.SQL.logical_reads	Nombre total de blocs ayant trouvé une correspondance et lus.	`blks_hit + blks_read`
queries_started	SQL	Requêtes	db.SQL.queries	Nombre de requêtes lancées.	Ne s'applique pas
queries_finished	SQL	Requêtes	db.SQL.queries	Nombre de requêtes terminées.	Ne s'applique pas
total_query_time	SQL	Millisecondes	db.SQL.total_query_time	Temps total passé à exécuter des instructions, en millisecondes.	Ne s'applique pas
active_transactions	Transactions	Transactions	db.Transactions.active_transactions	Nombre de transactions actives.	Ne s'applique pas
blocked_transactions	Transactions	Transactions	db.Transactions.blocked_transactions	Nombre de transactions bloquées.	Ne s'applique pas

Compteur	Type	Unité	Métrique	Description	Définition
commit_latency	Transactions	Microsecondes	db.Transactions.commit_latency	Durée moyenne des opérations de validation.	`db.Transactions.duration_commits / db.Transactions.xact_commit`
duration_commits	Transactions	Millisecondes	db.Transactions.duration_commits	Temps total de transaction passé au cours de la dernière minute, en millisecondes.	Ne s’applique pas
max_used_xact_ids	Transactions	Transactions	db.Transactions.max_used_xact_ids	Nombre de transactions qui n’ont pas été l’objet d’une opération vacuum.	Ne s’applique pas
oldest_inactive_logical_replication_slot_xid_age	Transactions	Longueur	db.Transactions.oldest_inactive_logical_replication_slot_xid_age	Âge de la transaction la plus ancienne dans un emplacement de réplication logique inactif.	Ne s’applique pas
oldest_active_logical_replication_slot_xid_age	Transactions	Longueur	db.Transactions.oldest_active_logical_replication_slot_xid_age	Âge de la transaction la plus ancienne dans un emplacement de réplication logique actif.	Ne s’applique pas

Compteur	Type	Unité	Métrique	Description	Définition
oldest_reader_feedback_xid_age	Transactions	Longueur	db.Transactions.oldest_reader_feedback_xid_age	Âge de la transaction la plus ancienne d'une transaction de longue durée sur une instance de lecteur Aurora ou une instance de lecteur de base de données globale Aurora.	Ne s'applique pas
oldest_prepared_transaction_xid_age	Transactions	Longueur	db.Transactions.oldest_prepared_transaction_xid_age	Âge de la plus ancienne transaction préparée.	Ne s'applique pas
oldest_running_transaction_xid_age	Transactions	Longueur	db.Transactions.oldest_running_transaction_xid_age	Âge de la transaction en cours d'exécution la plus ancienne.	Ne s'applique pas
max_connections	Utilisateurs	Utilisateurs	db.User.max_connections	Nombre maximum de connexions autorisées pour une base de données, tel que configuré dans le paramètre `max_connections` .	Ne s'applique pas
total_auth_attempts	Utilisateurs	Utilisateurs	db.User.total_auth_attempts	Nombre de tentatives de connexion à cette instance.	Ne s'applique pas

Compteur	Type	Unité	Métrique	Description	Définition
archive_failed_count	WAL	Fichiers par minute	db.WAL.archive_failed_count	Nombre de tentatives infructueuses d'archivage de fichiers WAL, en fichiers par minute.	Ne s'applique pas

Statistiques SQL pour Performance Insights

Les statistiques SQL sont des métriques liées aux performances des requêtes SQL qui sont collectées par Performance Insights. Performance Insights collecte des statistiques pour chaque seconde d'exécution d'une requête et pour chaque appel SQL. Les statistiques SQL sont une moyenne pour la plage de temps sélectionnée.

Un récapitulatif SQL est un composite de toutes les requêtes ayant un modèle donné mais n'ayant pas nécessairement les mêmes valeurs littérales. Le récapitulatif remplace les valeurs littérales par un point d'interrogation. Par exemple, `SELECT * FROM emp WHERE lname= ?`. Ce récapitulatif peut inclure les requêtes enfant suivantes :

```
SELECT * FROM emp WHERE lname = 'Sanchez'
SELECT * FROM emp WHERE lname = 'Olagappan'
SELECT * FROM emp WHERE lname = 'Wu'
```

Tous les moteurs prennent en charge les statistiques SQL pour les requêtes récapitulatives.

Pour obtenir les informations de prise en charge de la région, du moteur de base de données et de la classe d'instance pour cette fonctionnalité, consultez Prise en charge de la classe d'instances, de la région et du moteur de base de données Amazon Aurora pour les fonctionnalités d'analyse des performances.

Rubriques

- Statistiques SQL pour Aurora MySQL
- Statistiques SQL pour Aurora PostgreSQL

Statistiques SQL pour Aurora MySQL

Aurora MySQL collectent des statistiques SQL uniquement au niveau du récapitulatif. Aucune statistique n'est affichée au niveau de l'instruction.

Rubriques

- Statistiques récapitulatives pour Aurora MySQL
- Statistiques à la seconde pour Aurora MySQL
- Statistiques par l'appel pour Aurora MySQL
- Statistiques principales pour Aurora MySQL

Statistiques récapitulatives pour Aurora MySQL

Performance Insights collecte des statistiques de synthèse SQL à partir de la table `events_statements_summary_by_digest`. La table `events_statements_summary_by_digest` est gérée par votre base de données.

La table récapitulative n'a pas de politique d'éviction. Lorsque la table est pleine, la AWS Management Console affiche le message suivant :

```
Performance Insights is unable to collect SQL Digest statistics on new queries because
 the table events_statements_summary_by_digest is full.
Please truncate events_statements_summary_by_digest table to clear the issue. Check the
 User Guide for more details.
```

Dans ce cas, Aurora MySQL n'assure pas le suivi des requêtes SQL. Pour résoudre ce problème, Performance Insights tronque automatiquement la table de synthèse lorsque les deux conditions suivantes sont remplies :

- La table est pleine.
- Performance Insights gère automatiquement le schéma de performance.

  Pour la gestion automatique, le paramètre `performance_schema` doit être défini sur `0` et la Source ne doit pas être définie sur `user`. Si Performance Insights ne gère pas automatiquement le schéma de performance, consultez Présentation du schéma de performance pour Performance Insights sur Aurora MySQL.

Dans la AWS CLI, vérifiez la source d'une valeur de paramètre en exécutant la commande describe-db-parameters.

Statistiques à la seconde pour Aurora MySQL

Les statistiques SQL suivantes sont disponibles pour les clusters de bases de données Aurora MySQL.

Métrique	Unit
db.sql_tokenized.stats.count_star_per_sec	Appels à la seconde
db.sql_tokenized.stats.sum_timer_wait_per_sec	Latence moyenne par seconde (en ms)
db.sql_tokenized.stats.sum_select_full_join_per_sec	Sélections de jointures complètes par seconde
db.sql_tokenized.stats.sum_select_range_check_per_sec	Sélections de vérifications de plages par seconde
db.sql_tokenized.stats.sum_select_scan_per_sec	Sélections d'analyses par seconde
db.sql_tokenized.stats.sum_sort_merge_passes_per_sec	Tris de transmissions de fusion par seconde
db.sql_tokenized.stats.sum_sort_scan_per_sec	Tris d'analyses par seconde
db.sql_tokenized.stats.sum_sort_range_per_sec	Tris de plages par seconde
db.sql_tokenized.stats.sum_sort_rows_per_sec	Tris de lignes par seconde
db.sql_tokenized.stats.sum_rows_affected_per_sec	Lignes affectées par seconde
db.sql_tokenized.stats.sum_rows_examined_per_sec	Lignes examinées par seconde
db.sql_tokenized.stats.sum_rows_sent_per_sec	Lignes envoyées par seconde

Métrique	Unit
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_sec	Créations de tables de disques temporaires par seconde
db.sql_tokenized.stats.sum_created_tmp_tables_per_sec	Créations de tables temporaires par seconde
db.sql_tokenized.stats.sum_lock_time_per_sec	Temps de verrouillage par seconde (en millisecondes)

Statistiques par l’appel pour Aurora MySQL

Les métriques suivantes fournissent les statistiques par appel pour une instruction SQL.

Métrique	Unité
db.sql_tokenized.stats.sum_timer_wait_per_call	Latence moyenne par appel (en millisecondes)
db.sql_tokenized.stats.sum_select_full_join_per_call	Sélections de jointures complètes par appel
db.sql_tokenized.stats.sum_select_range_check_per_call	Sélections de vérifications de plages par appel
db.sql_tokenized.stats.sum_select_scan_per_call	Sélections d’analyses par appel
db.sql_tokenized.stats.sum_sort_merge_passes_per_call	Tris de transmissions de fusion par appel
db.sql_tokenized.stats.sum_sort_scan_per_call	Tris d’analyses par appel
db.sql_tokenized.stats.sum_sort_range_per_call	Tris de plages par appel
db.sql_tokenized.stats.sum_sort_rows_per_call	Tris de lignes par appel
db.sql_tokenized.stats.sum_rows_affected_per_call	Lignes affectées par appel

Métrique	Unité
db.sql_tokenized.stats.sum_rows_examined_per_call	Lignes examinées par appel
db.sql_tokenized.stats.sum_rows_sent_per_call	Lignes envoyées par appel
db.sql_tokenized.stats.sum_created_tmp_disk_tables_per_call	Créations de tables de disques temporaires par appel
db.sql_tokenized.stats.sum_created_tmp_tables_per_call	Créations de tables temporaires par appel
db.sql_tokenized.stats.sum_lock_time_per_call	Temps de verrouillage par appel (en ms)

Statistiques principales pour Aurora MySQL

Les statistiques SQL suivantes sont disponibles pour les clusters de bases de données Aurora MySQL.

Métrique	Unité
db.sql_tokenized.stats.count_star	Calls
db.sql_tokenized.stats.sum_timer_wait	Temps d’attente (en ms)
db.sql_tokenized.stats.sum_select_full_join	Sélection de jointures complètes
db.sql_tokenized.stats.sum_select_range_check	Sélection des vérifications de plage
db.sql_tokenized.stats.sum_select_scan	Sélection des analyses
db.sql_tokenized.stats.sum_sort_merge_passes	Tri des transmissions de fusion
db.sql_tokenized.stats.sum_sort_scan	Tris des analyses
db.sql_tokenized.stats.sum_sort_range	Tri des plages
db.sql_tokenized.stats.sum_sort_rows	Tri des lignes

Métrique	Unité
db.sql_tokenized.stats.sum_rows_affected	Lignes concernées
db.sql_tokenized.stats.sum_rows_examined	Lignes examinées
db.sql_tokenized.stats.sum_rows_sent	Lignes envoyées
db.sql_tokenized.stats.sum_created_tmp_disk_tables	Tables de disques temporaires créées
db.sql_tokenized.stats.sum_created_tmp_tables	Tables temporaires créées
db.sql_tokenized.stats.sum_lock_time	Temps de verrouillage (en ms)

Statistiques SQL pour Aurora PostgreSQL

Pour chaque appel SQL et pour chaque seconde d'exécution d'une requête, Performance Insights collecte des statistiques SQL. Tous les moteurs Aurora collectent des statistiques uniquement au niveau des récapitulatifs.

Vous trouverez ci-dessous des informations sur les statistiques de niveau récapitulatif pour Aurora PostgreSQL.

Rubriques

- Statistiques récapitulatives pour Aurora PostgreSQL
- Statistiques récapitulatives à la seconde pour Aurora PostgreSQL
- Statistiques récapitulatives par appel pour Aurora PostgreSQL
- Statistiques principales pour Aurora PostgreSQL

Statistiques récapitulatives pour Aurora PostgreSQL

Pour afficher les statistiques récapitulatives SQL, la bibliothèque `pg_stat_statements` doit être chargée. Pour les clusters de bases de données Aurora PostgreSQL compatibles avec PostgreSQL 10, cette bibliothèque est chargée par défaut. Pour les clusters de bases de données Aurora PostgreSQL compatibles avec PostgreSQL 9.6, vous devez activer cette bibliothèque manuellement. Pour l'activer manuellement, ajoutez `pg_stat_statements` à `shared_preload_libraries` dans le groupe de paramètres de base de données associé à

l'instance de base de données. Puis, redémarrez votre instance de base de données. Pour plus d'informations, consultez Groupes de paramètres pour Amazon Aurora.

> **Note**
>
> Performance Insights peut uniquement collecter des statistiques pour les requêtes non tronquées dans `pg_stat_activity`. Par défaut, les bases de données PostgreSQL tronquent les requêtes de plus de 1 024 octets. Pour augmenter la taille de la requête, modifiez le paramètre `track_activity_query_size` dans le groupe de paramètres de base de données associé à votre instance de base de données. Lorsque vous modifiez ce paramètre, un redémarrage d'instance de base de données est obligatoire.

Statistiques récapitulatives à la seconde pour Aurora PostgreSQL

Les statistiques récapitulatives SQL suivantes sont disponibles pour les instances de base de données Aurora PostgreSQL.

Métrique	Unité
db.sql_tokenized.stats.calls_per_sec	Appels par seconde
db.sql_tokenized.stats.rows_per_sec	Lignes par seconde
db.sql_tokenized.stats.total_time_per_sec	Exécutions actives moyennes par seconde
db.sql_tokenized.stats.shared_blks_hit_per_sec	Accès en masse par seconde
db.sql_tokenized.stats.shared_blks_read_per_sec	Lectures en masse par seconde
db.sql_tokenized.stats.shared_blks_dirtied_per_sec	Blocs salis par seconde
db.sql_tokenized.stats.shared_blks_written_per_sec	Écritures en masse par seconde
db.sql_tokenized.stats.local_blks_hit_per_sec	Nombre de blocs locaux par seconde
db.sql_tokenized.stats.local_blks_read_per_sec	Lectures par bloc local par seconde

Métrique	Unité
db.sql_tokenized.stats.local_blks_dirtied_per_sec	Bloc local sale par seconde
db.sql_tokenized.stats.local_blks_written_per_sec	Écritures par bloc local par seconde
db.sql_tokenized.stats.temp_blks_written_per_sec	Écritures temporaires par seconde
db.sql_tokenized.stats.temp_blks_read_per_sec	Lectures temporaires par seconde
db.sql_tokenized.stats.blk_read_time_per_sec	Lectures simultanées moyennes par seconde
db.sql_tokenized.stats.blk_write_time_per_sec	Écritures simultanées moyennes par seconde

Statistiques récapitulatives par appel pour Aurora PostgreSQL

Les métriques suivantes fournissent les statistiques par appel pour une instruction SQL.

Métrique	Unité
db.sql_tokenized.stats.rows_per_call	Lignes par appel
db.sql_tokenized.stats.avg_latency_per_call	Latence moyenne par appel (en millisecondes)
db.sql_tokenized.stats.shared_blks_hit_per_call	Accès en masse par appel
db.sql_tokenized.stats.shared_blks_read_per_call	Lectures en masse par appel
db.sql_tokenized.stats.shared_blks_written_per_call	Écritures en masse par appel
db.sql_tokenized.stats.shared_blks_dirtied_per_call	Blocs salis par appel
db.sql_tokenized.stats.local_blks_hit_per_call	Nombre d’accès par bloc local par appel

Métrique	Unité
db.sql_tokenized.stats.local_blks_read_per_call	Lectures par bloc local par appel
db.sql_tokenized.stats.local_blks_dirtied_per_call	Bloc local sale par appel
db.sql_tokenized.stats.local_blks_written_per_call	Écritures de blocs locaux par appel
db.sql_tokenized.stats.temp_blks_written_per_call	Écritures de blocs temporaires par appel
db.sql_tokenized.stats.temp_blks_read_per_call	Lectures de blocs temporaires par appel
db.sql_tokenized.stats.blk_read_time_per_call	Temps de lecture par appel (en ms)
db.sql_tokenized.stats.blk_write_time_per_call	Temps d’écriture par appel (en ms)

Statistiques principales pour Aurora PostgreSQL

Les statistiques SQL suivantes sont disponibles pour les instances de base de données Aurora PostgreSQL.

Métrique	Unité
db.sql_tokenized.stats.calls	Calls
db.sql_tokenized.stats.rows	Lignes
db.sql_tokenized.stats.total_time	Temps total (en ms)
db.sql_tokenized.stats.shared_blks_hit	Accès par blocs
db.sql_tokenized.stats.shared_blks_read	Blocs lus
db.sql_tokenized.stats.shared_blks_dirtied	Blocs sales
db.sql_tokenized.stats.shared_blks_written	Écritures de blocs

Métrique	Unité
db.sql_tokenized.stats.local_blks_hit	Accès par blocs locaux
db.sql_tokenized.stats.local_blks_read	Lectures de blocs locaux
db.sql_tokenized.stats.local_blks_dirtied	Blocs locaux sales
db.sql_tokenized.stats.local_blks_written	Écritures de blocs locaux
db.sql_tokenized.stats.temp_blks_written	Écritures temporaires
db.sql_tokenized.stats.temp_blks_read	Lectures temporaires
db.sql_tokenized.stats.blk_read_time	Lectures simultanées moyennes (en ms)
db.sql_tokenized.stats.blk_write_time	Écritures simultanées moyennes (en ms)

Pour plus d'informations sur ces métriques, consultez pg_stat_statements dans la documentation PostgreSQL.

Métriques du système d'exploitation dans la surveillance améliorée

Amazon Aurora fournit des métriques en temps réel pour le système d'exploitation sur lequel votre cluster de bases de données s'exécute. Aurora fournit les métriques issues de la surveillance améliorée à votre compte Amazon CloudWatch Logs. Les tableaux suivants répertorient les métriques du système d'exploitation disponibles avec Amazon CloudWatch Logs.

Rubriques

- Métriques de système d'exploitation pour Aurora

Métriques de système d’exploitation pour Aurora

Groupe	Métrique	Nom de la console	Description
General	engine	Ne s’applique pas	Moteur de base de données de l’instance de base de données.
	instanceID	Ne s’applique pas	Identifiant de l’instance de base de données.
	instanceResourceID	Ne s’applique pas	Identifiant immuable pour l'instance de base de données unique à une AWS région, également utilisé comme identifiant du flux de journal.
	numVCPUs	Ne s’applique pas	Le nombre de virtuels CPUs pour l'instance de base de données.
	timestamp	Ne s’applique pas	Heure à laquelle la métrique a été évaluée.
	uptime	Ne s’applique pas	Temps d’activité de l’instance de base de données.
	version	Ne s’applique pas	Version du format JSON du flux des métriques du système d’exploitation.
cpuUtilization	guest	Invité UC	Pourcentage de l’UC en cours d’utilisation par les programmes invités.
	idle	Inactivité de l’UC	Pourcentage de l’UC inactive.

Groupe	Métrique	Nom de la console	Description
	`irq`	IRQ UC	Pourcentage de l’UC en cours d’utilisation par les interruptions logicielles.
	`nice`	UC Nice	Pourcentage de l’UC en cours d’utilisation par des programmes s’exécutant avec la priorité la plus faible.
	`steal`	UC Steal	Pourcentage de l’UC en cours d’utilisation par d’autres machines virtuelles.
	`system`	Système UC	Pourcentage de l’UC en cours d’utilisation par le noyau.
	`total`	Total UC	Pourcentage total de l’UC en cours d’utilisation. Cette valeur inclut la valeur `nice`.
	`user`	Utilisateur UC	Pourcentage de l’UC en cours d’utilisation par des programmes utilisateurs.
	`wait`	Attente du processeur	Pourcentage de CPU inutilisé pendant l'attente de I/O l'accès.
`diskIO`	`avgQueueLen`	Taille moyenne de la file d’attente	Le nombre de demandes en attente dans la file d'attente de l' I/O appareil.
	`avgReqSz`	Taille moyenne de la demande	Taille moyenne de requête, en kilo-octets.
	`await`	Disk I/O Wait	Nombre de millisecondes requises pour répondre aux requêtes, y compris le temps d’attente et le temps de service.

Groupe	Métrique	Nom de la console	Description
	`device`	Ne s’applique pas	Identifiant du périphérique de disque en cours d’utilisation.
	`readIOsPS`	E/S lecture	Nombre d’opérations de lecture par seconde.
	`readKb`	Total lecture	Nombre total de kilo-octets lus.
	`readKbPS`	Ko/s lecture	Nombre de kilo-octets lus par seconde.
	`readLatency`	Latence de lecture	Temps écoulé entre la soumission d'une I/O demande de lecture et son achèvement, en millisecondes. Cette métrique est uniquement disponible pour Amazon Aurora.
	`readThroughput`	Débit de lecture	Quantité de débit réseau utilisée par les demandes adressées au cluster DB, en octets par seconde. Cette métrique est uniquement disponible pour Amazon Aurora.
	`rrqmPS`	Rrqms	Nombre de requêtes de lecture fusionnées mises en file d’attente par seconde.
	`tps`	TPS	Le nombre de I/O transactions par seconde.
	`util`	I/O Utilitaire de disque	Pourcentage de temps UC pendant lequel les requêtes ont été émises.
	`writeIOsPS`	E/S écriture	Nombre d’opérations d’écriture par seconde.

Groupe	Métrique	Nom de la console	Description
	`writeKb`	Total écriture	Nombre total de kilo-octets écrits.
	`writeKbPS`	Ko/s écriture	Nombre de kilo-octets écrits par seconde.
	`writeLatency`	Latence en écriture	Temps moyen écoulé entre la soumission d'une I/O demande d'écriture et son achèvement, en millisecondes. Cette métrique est uniquement disponible pour Amazon Aurora.
	`writeThroughput`	Débit d’écriture	Quantité de débit réseau utilisée par les réponses du cluster DB, en octets par seconde. Cette métrique est uniquement disponible pour Amazon Aurora.
	`wrqmPS`	Wrqms	Nombre de requêtes d’écriture fusionnées mises en file d’attente par seconde.
`mountPoint`	Ne s’applique pas	Chemin vers le système de fichiers.	
`fileSys`	`maxFiles`	Nombre maximum d’inodes	Nombre maximum de fichiers pouvant être créés pour le système de fichiers.
	`mountPoint`	Ne s’applique pas	Chemin vers le système de fichiers. Dans ce cas `/rdsdbdata*` , il représente l'agrégat de tous les volumes de stockage.

Groupe	Métrique	Nom de la console	Description
	name	Ne s'applique pas	Nom du système de fichiers.
	total	Total système de fichiers	Quantité totale d'espace disque disponible pour le système de fichiers, en kilo-octets.
	used	Système de fichiers utilisé	Quantité d'espace disque utilisé par des fichiers du système de fichiers, en kilo-octets.
	usedFilePercent	Inodes utilisés	Pourcentage de fichiers disponibles en cours d'utilisation.
	usedFiles	% utilisé	Nombre de fichiers dans le système de fichiers.
	usedPercent	Système de fichiers utilisé	Pourcentage d'espace de disque du système de fichiers en cours d'utilisation.
loadAverageMinute	fifteen	Charge moyenne 15 min	Nombre de processus demandant du temps UC au cours des 15 dernières minutes.
	five	Charge moyenne 5 min	Nombre de processus demandant du temps UC au cours des 5 dernières minutes.
	one	Charge moyenne 1 min	Nombre de processus demandant du temps UC au cours de la dernière minute.
memory	active	Mémoire active	Quantité de mémoire attribuée, en kilo-octets.

Groupe	Métrique	Nom de la console	Description
	`buffers`	Mémoire mise en tampon	Quantité de mémoire utilisée pour la mise en mémoire tampon des I/O demandes avant l'écriture sur le périphérique de stockage, en kilo-octets.
	`cached`	Mémoire mise en cache	Quantité de mémoire utilisée pour la mise en cache des I/O basées sur le système de fichiers.
	`dirty`	Mémoire corrompue	Quantité de pages mémoire de la RAM ayant été modifiées mais non écrites dans le bloc de données associé dans le stockage, en kilo-octets.
	`free`	Mémoire libre	Quantité de mémoire non attribuée, en kilo-octets.
	`hugePages Free`	Grandes pages gratuites	Nombre de grandes pages gratuites. Les grandes pages sont une fonction du noyau Linux.
	`hugePages Rsvd`	Grandes pages Rsvd	Nombre de grandes pages dédiées.
	`hugePages Size`	Taille des grandes pages	Taille de chaque unité de grandes pages, en kilo-octe ts.
	`hugePages Surp`	Grandes pages excéd	Nombre de grandes pages excédentaires disponibles par rapport au nombre total.
	`hugePages Total`	Total de grandes pages	Le nombre total de grandes pages.

Groupe	Métrique	Nom de la console	Description
	`inactive`	Mémoire inactive	Quantité de pages mémoire moins fréquemment utilisées, en kilo-octets.
	`mapped`	Mémoire mappée	Quantité totale de contenu du système de fichiers mappé en mémoire dans un espace d'adressage de processus, en kilo-octets.
	`pageTables`	Tables de pages	Quantité de mémoire utilisée par les tables de page, en kilo-octets.
	`slab`	Mémoire de section	Quantité de structures de données noyau réutilisables, en kilo-octets.
	`total`	Mémoire totale	Quantité totale de mémoire, en kilo-octets.
	`writeback`	Mémoire en écriture différée	Quantité de pages de modification dans la RAM encore écrites dans le stockage de sauvegarde, en kilo-octets.
`network`	`interface`	Ne s'applique pas	Identifiant pour l'interface réseau utilisée pour l'instance de base de données.
	`rx`	RX	Nombre d'octets reçus par seconde.
	`tx`	TX	Nombre d'octets téléchargés par seconde.
`processList`	`cpuUsedPc`	% UC	Pourcentage de l'UC utilisé par le processus.
	`id`	Ne s'applique pas	Identifiant du processus.

Groupe	Métrique	Nom de la console	Description
	`memoryUsedPc`	% MEM	Pourcentage de mémoire utilisé par le processus.
	`name`	Ne s’applique pas	Nom du processus.
	`parentID`	Ne s’applique pas	Identifiant de processus pour le processus parent du processus.
	`rss`	RES	Quantité de RAM allouée au processus, en kilo-octets.
	`tgid`	Ne s’applique pas	Identifiant du groupe de threads. Numéro représentant l’ID du processus auquel appartient le thread. Cet identifiant permet de regrouper les threads d’un même processus.
	`vss`	VIRT	Quantité de mémoire virtuelle allouée au processus, en kilo-octets.
`swap`	`total`	Swap	Quantité de mémoire d’échange disponible, en kilo-octets.
	`in`	Swaps dans	Quantité de mémoire, en kilo-octets, échangée depuis le disque.
	`out`	Swaps vers	Quantité de mémoire, en kilo-octets, échangée vers le disque.
	`free`	Swap libre	Quantité de mémoire d’échange libre, en kilo-octets.
	`cached`	Swap validé	Quantité de mémoire d’échange, en kilo-octets, utilisée en tant que mémoire cache.

Groupe	Métrique	Nom de la console	Description
`tasks`	`blocked`	Tâches bloquées	Nombre de tâches bloquées.
	`running`	Tâches en cours d’exécution	Nombre de tâches en cours d’exécution.
	`sleeping`	Tâches en veille	Nombre de tâches en veille.
	`stopped`	Tâches arrêtées	Nombre de tâches arrêtées.
	`total`	Total de tâches	Nombre total de tâches.
	`zombie`	Tâches zombies	Nombre de tâches enfant inactives avec une tâche parent active.

Surveillance des événements, des journaux et des flux dans un cluster de bases de donnéesAmazon Aurora

Lorsque vous surveillez vos bases de données Amazon Aurora et vos autres solutions AWS, votre objectif est de maintenir les éléments suivants :

- Fiabilité
- Disponibilité
- Performances
- Sécurité

Surveillance des métriques d'un cluster de bases de données Amazon Aurora explique la surveillance de votre cluster à l'aide de métriques. Une solution complète doit également surveiller les événements, les fichiers journaux et les flux d'activité de base de données. AWS vous fournit les outils de surveillance suivants :

- Amazon EventBridge est un service de bus d'événements sans serveur qui permet de connecter facilement vos applications à des données provenant de diverses sources. EventBridge fournit un flux de données en temps réel à partir de vos propres applications, d'applications de type logiciel en tant que service (SaaS) et de services AWS. EventBridge achemine ces données vers des cibles telles que AWS Lambda. Cela vous permet de surveiller les événements qui se produisent dans les services et de créer des architectures basées sur les événements. Pour plus d'informations, consultez le Guide de l'utilisateur Amazon EventBridge.
- Amazon CloudWatch Logs permet de surveiller, de stocker et d'accéder à vos fichiers journaux provenant des instances Amazon Aurora, d'AWS CloudTrail et d'autres sources. Amazon CloudWatch Logs peut surveiller les informations contenues dans les fichiers journaux et vous avertir lorsque certains seuils sont atteints. Vous pouvez également archiver vos données de journaux dans une solution de stockage hautement durable. Pour plus d'informations, consultez le Guide de l'utilisateur Amazon CloudWatch Logs.
- AWS CloudTrail capture les appels d'API et les événements associés effectués par votre Compte AWS ou au nom de ce dernier. CloudTrail livre les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes qui ont appelé AWS, l'adresse IP source à partir de laquelle les appels ont été émis, ainsi que le moment où les appels ont eu lieu. Pour plus d'informations, consultez le Guide de l'utilisateur AWS CloudTrail.

- Database Activity Streams est une fonctionnalité Amazon Aurora qui fournit un flux en temps quasi réel de l'activité dans votre cluster de base de données. Amazon Aurora envoie les activités vers un flux de données Amazon Kinesis. Le flux Kinesis est créé automatiquement. Dans Kinesis, vous pouvez configurer des services AWS tels qu'Amazon Data Firehose et AWS Lambda pour consommer le flux et stocker les données.

Rubriques

Affichage des journaux, des événements et des flux dans la console Amazon RDS

Amazon RDS s'intègre avec Services AWS pour afficher des informations sur les journaux, les événements et les flux d'activité de base de données dans la console RDS.

L'onglet Logs & events (Journaux et événements) de votre cluster de bases de données Aurora affiche les informations suivantes :

- Politiques et activités de scalabilité automatique : affiche les politiques et les activités relatives à la fonction de scalabilité automatique d'Aurora. Ces informations s'affichent uniquement dans l'onglet Logs & events (Journaux et événements) au niveau du cluster.
- Des alarmes Amazon CloudWatch : affiche toutes les alarmes de métriques que vous avez configurées pour l'instance de base de données dans votre cluster Aurora. Si vous n'avez pas configuré d'alarmes, vous pouvez les créer dans la console RDS.
- Événements récents : affiche un récapitulatif des événements (changements d'environnement) pour votre instance ou cluster de bases de données Aurora. Pour plus d'informations, consultez Affichage d'événements Amazon RDS.

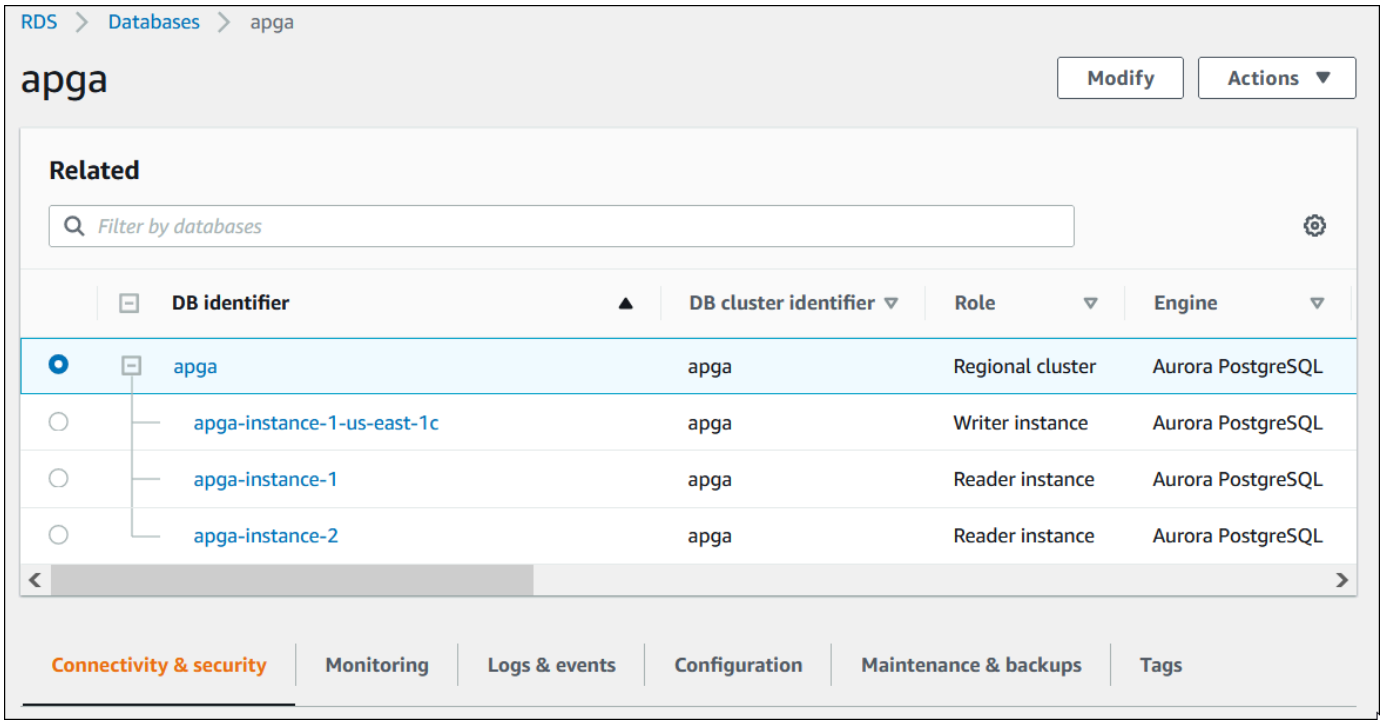
- Journaux : affiche les fichiers journaux de base de données générés par une instance de base de données dans votre cluster Aurora. Pour plus d'informations, consultez Surveillance des fichiers journaux Amazon Aurora.

L'onglet Configuration (Configuration) affiche des informations sur les flux d'activité de base de données.

Pour afficher les journaux, les événements et les flux de votre cluster de base de données Aurora dans la console RDS

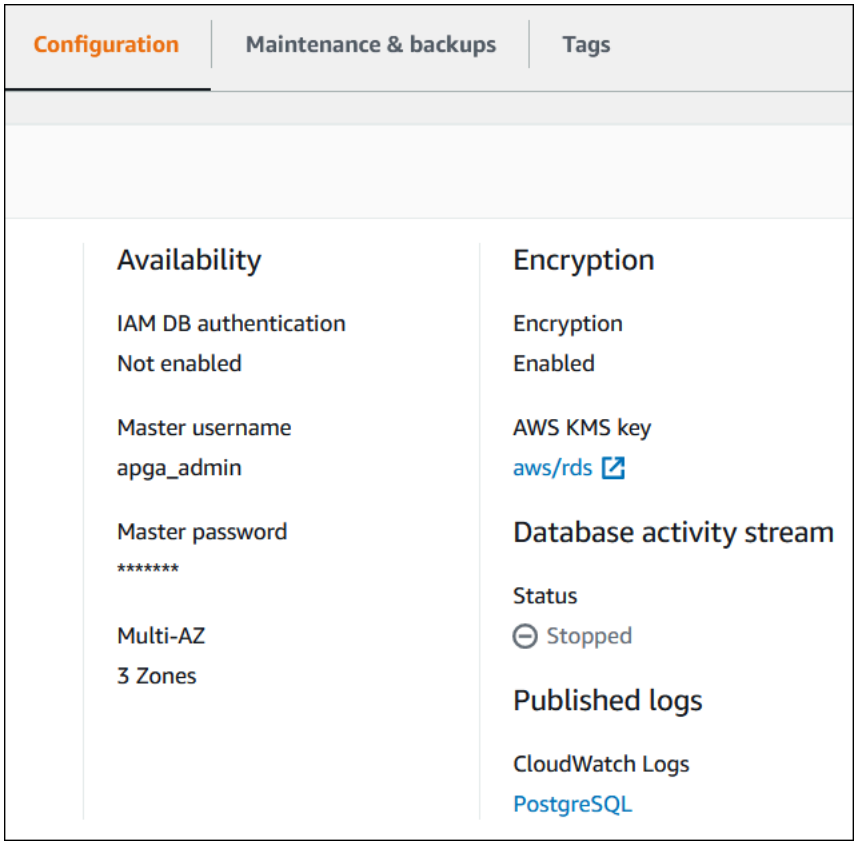
1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Sélectionnez le nom du cluster d' de base de données Aurora que vous souhaitez surveiller.

   La page Bases de données s'affiche. L'exemple suivant illustre un cluster de bases de données Amazon Aurora PostgreSQL nommé apga.



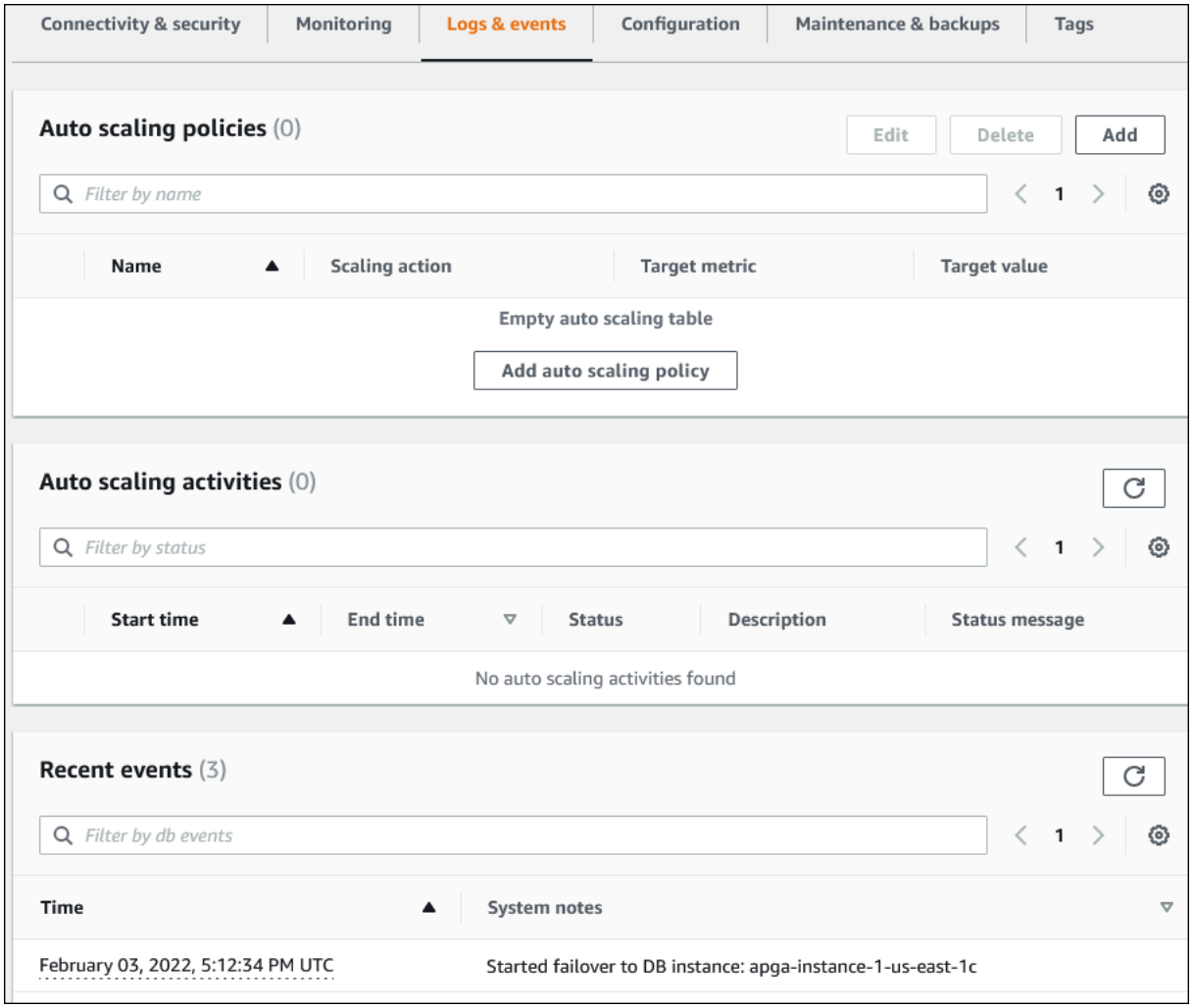
4. Faites défiler vers le bas et choisissez Configuration (Configuration).

   L'exemple suivant illustre l'état des flux d'activité de base de données pour votre cluster.



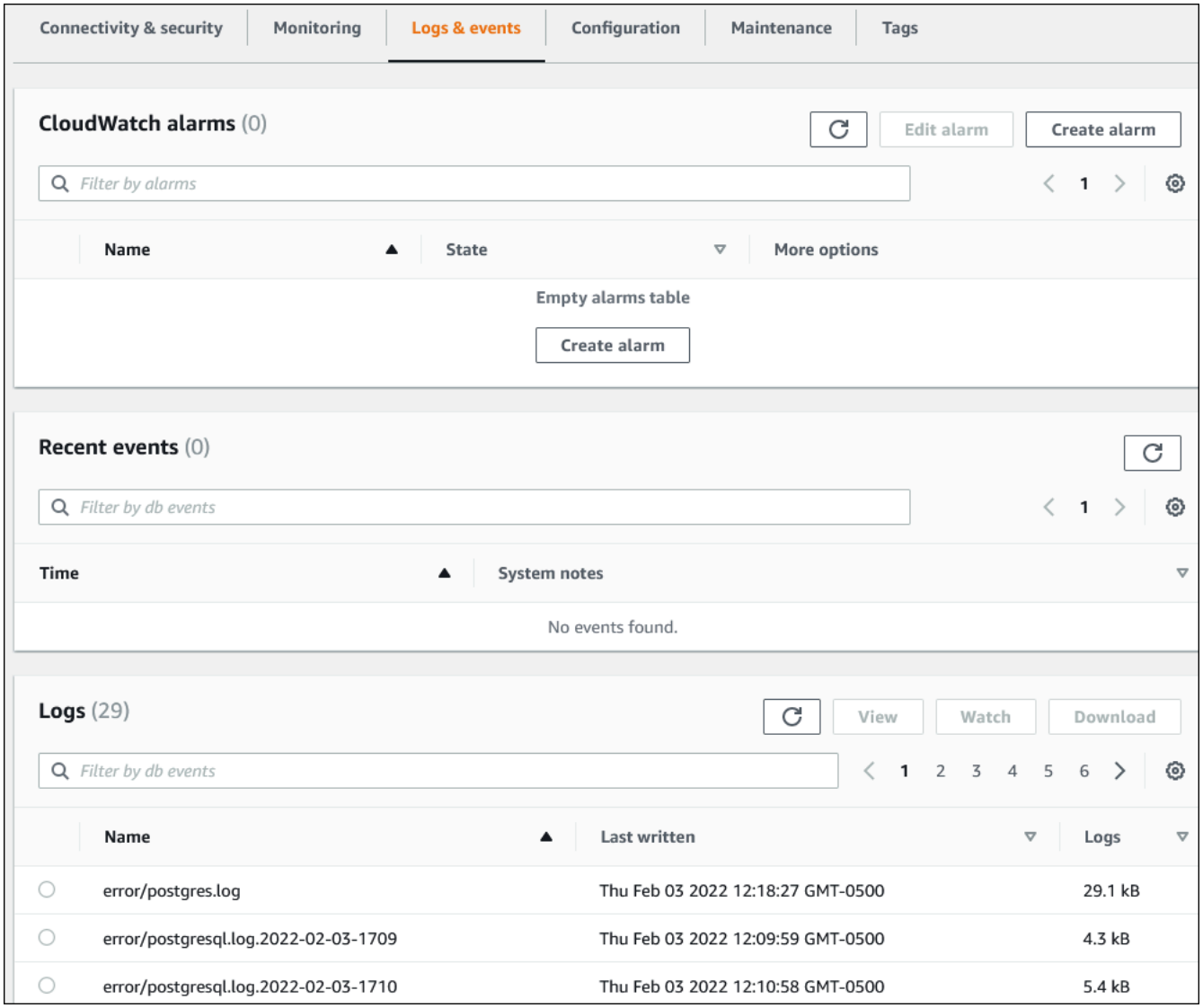
5. Choisissez Logs & events (Journaux et événements).

   La section Logs & events (Journaux et événements) et événements s'affiche.



6. Choisissez une instance de base de données dans votre cluster Aurora, puis choisissez Logs & events (Journaux et événements) pour l'instance.

   L'exemple suivant montre que le contenu est différent entre la page de l'instance de base de données et la page du cluster de bases de données. La page de l'instance de base de données affiche les journaux et les alarmes.


Connectivity & security
Monitoring
Logs & events
Configuration
Maintenance
Tags
CloudWatch alarms (0)
Edit alarm
Create alarm
Filter by alarms
1
Name
State
More options
Empty alarms table
Create alarm
Recent events (0)
Filter by db events
1
Time
System notes
No events found.
Logs (29)
View
Watch
Download
Filter by db events
1 2 3 4 5 6
Name
Last written
Logs
error/postgres.log
Thu Feb 03 2022 12:18:27 GMT-0500
29.1 kB
error/postgresql.log.2022-02-03-1709
Thu Feb 03 2022 12:09:59 GMT-0500
4.3 kB
error/postgresql.log.2022-02-03-1710
Thu Feb 03 2022 12:10:58 GMT-0500
5.4 kB

Surveillance des événements Amazon Aurora

Un événement indique un changement dans un environnement. Il peut s'agir d'un environnement AWS, d'un service partenaire ou d'une application SaaS, ou d'une application ou d'un service personnalisé. Pour obtenir la description des événements Aurora, consultez Catégories d'événements et messages d'événements pour Aurora.

Rubriques

Présentation des événements pour Aurora

Un événement RDS indique un changement dans l'environnement Aurora. Par exemple, Amazon Aurora génère un événement quand un correctif est appliqué à un cluster de bases de données. Amazon Aurora fournit des événements à EventBridge en quasi-temps réel.

> **Note**
>
> Amazon RDS émet les événements dans la mesure du possible. Nous vous recommandons d'éviter d'écrire des programmes en fonction de la présence d'événements de notification ou de leur ordre, car il peut ne pas y en avoir ou ils peuvent ne pas être dans l'ordre défini.

Amazon RDS enregistre les événements qui concernent les ressources suivantes :

- clusters de bases de données

  Pour obtenir une liste des événements du cluster, consultez Événements de cluster de bases de données.
- Instances DB

  Pour obtenir une liste des événements d'instance de base de données, consultez Événements d'instance de base de données.

- Groupes de paramètres de base de données

  Pour obtenir une liste des événements de groupe de paramètres de base de données, consultez Événements de groupe de paramètres de base de données.
- Groupes de sécurité DB

  Pour obtenir la liste des événements relatifs aux groupes de sécurité de la base de données, consultez Événements de groupe de sécurité de base de données.
- Instantanés du cluster de bases de données

  Pour obtenir une liste des événements d'instantanés du cluster de bases de données, consultez Événements d'instantané de cluster de bases de données.
- Événements RDS Proxy

  Pour obtenir une liste des événements RDS Proxy, consultez Événements RDS Proxy.
- Événements de déploiement bleu/vert

  Pour obtenir la liste des événements de déploiement bleu/vert, consultez Événements de déploiement bleu/vert.

Les informations collectées sont les suivantes :

- Date et heure de l'événement.
- Nom de la source et type de source de l'événement
- Message associé à l'événement
- Les notifications d'événements incluent des balises datant du moment où le message a été envoyé et peuvent ne pas refléter les balises au moment où l'événement s'est produit.

Affichage d’événements Amazon RDS

Vous pouvez récupérer les informations suivantes sur les événements pour vos ressources Amazon Aurora :

- Nom de la ressource
- Type de ressource
- Heure de l’événement
- Résumé du message de l’événement

Vous pouvez accéder aux événements dans les sections suivantes de la AWS Management Console :

- L’onglet Événements, qui affiche les événements des 24 dernières heures.
- Le tableau Événements récents de la section Journaux et événements de l’onglet Bases de données, qui peut afficher les événements survenus au cours des deux dernières semaines.

Vous pouvez également récupérer les événements en utilisant la commande AWS CLI describe-events, ou l’opération DescribeEvents de l’API RDS. Si vous utilisez l’AWS CLI ou l’API RDS pour afficher les événements, vous pouvez récupérer les événements datant de 14 jours maximum.

> **Note**
>
> Si vous devez stocker des événements pendant des périodes plus longues, vous pouvez envoyer des événements Amazon RDS à EventBridge. Pour plus d’informations, consultez Création d’une règle qui se déclenche sur un événement Amazon Aurora.

Pour la description des événements Amazon Aurora, consultez Catégories d’événements et messages d’événements pour Aurora.

Pour accéder à des informations détaillées sur les événements utilisant AWS CloudTrail, y compris les paramètres de requête, consultez Événements CloudTrail.

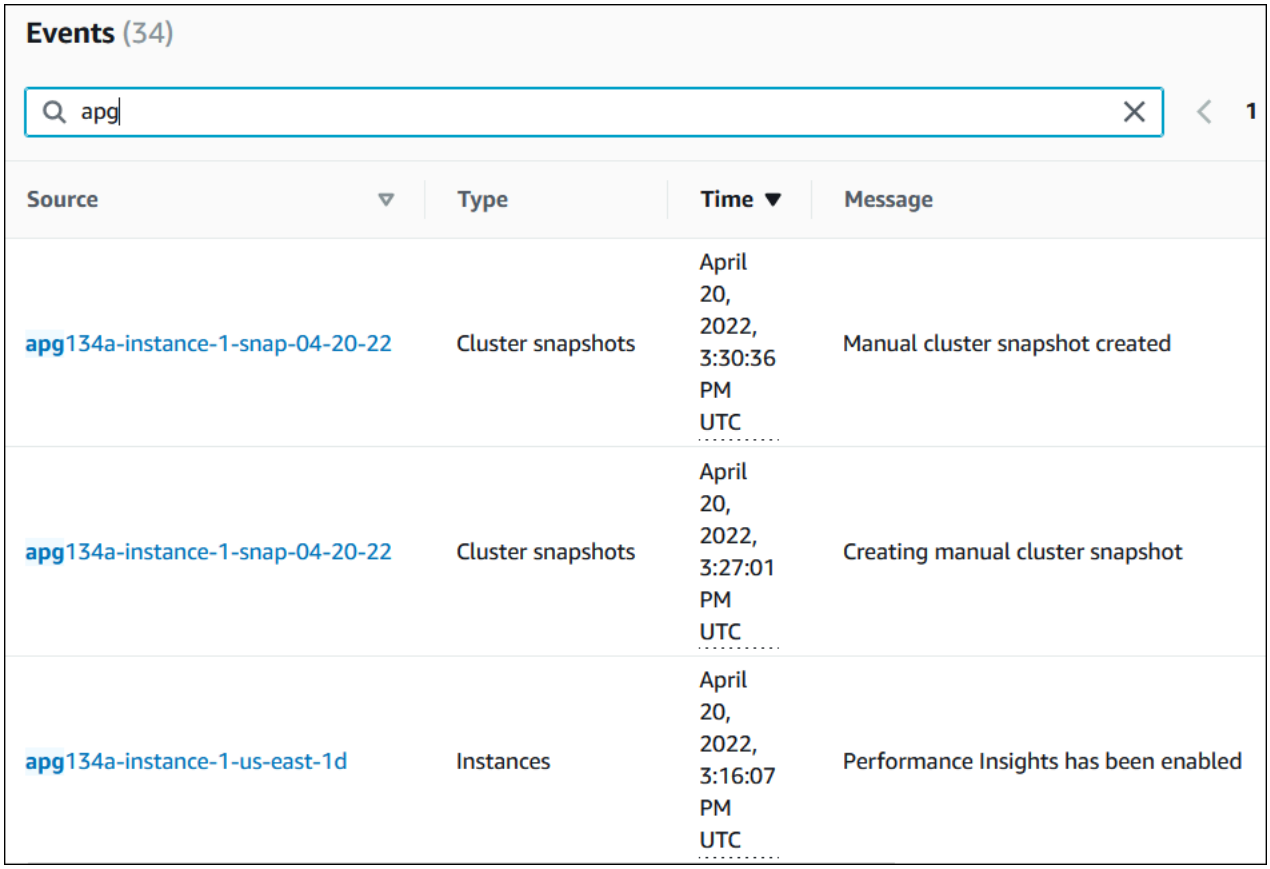
Console

Pour voir tous les événements Amazon RDS des dernières 24 heures

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, sélectionnez Events.

   Les événements disponibles s'affichent sous forme de liste.
3. (Facultatif) Entrez un terme de recherche pour filtrer vos résultats.

   L'exemple suivant montre une liste d'événements filtrés par les caractères **apg**.



AWS CLI

Pour afficher tous les événements générés au cours de la dernière heure, appelez la commande describe-events sans paramètres.

```
aws rds describe-events
```

L'exemple de sortie suivant montre qu'une instance de cluster de bases de données a commencé à se rétablir.

```
{
    "Events": [
        {
            "EventCategories": [
                "recovery"
            ],
            "SourceType": "db-instance",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mycluster-instance-1",
            "Date": "2022-04-20T15:02:38.416Z",
            "Message": "Recovery of the DB instance has started. Recovery time will
 vary with the amount of data to be recovered.",
            "SourceIdentifier": "mycluster-instance-1"
        }, ...
```

Pour afficher tous les événements Amazon RDS des 10 080 dernières minutes (sept jours), appelez la commande AWS CLI describe-events et définissez le paramètre `--duration` sur `10080`.

```
aws rds describe-events --duration 10080
```

L'exemple suivant montre les événements dans la plage de temps spécifiée pour l'instance de base de données *`test-instance`*.

```
aws rds describe-events \
    --source-identifier test-instance \
    --source-type db-instance \
    --start-time 2022-03-13T22:00Z \
    --end-time 2022-03-13T23:59Z
```

L'exemple de sortie suivant montre l'état d'une sauvegarde.

```
{
    "Events": [
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
            ],
```

```
            "Message": "Backing up DB instance",
            "Date": "2022-03-13T23:09:23.983Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        },
        {
            "SourceType": "db-instance",
            "SourceIdentifier": "test-instance",
            "EventCategories": [
                "backup"
            ],
            "Message": "Finished DB Instance backup",
            "Date": "2022-03-13T23:15:13.049Z",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance"
        }
    ]
}
```

« Hello, World! »

Vous pouvez afficher tous les événements des instances Amazon RDS des 14 derniers jours en appelant l'opération d'API RDS DescribeEvents et en définissant le paramètre `Duration` sur `20160`.

Utiliser la notification d'événements d'Amazon RDS

Amazon RDS utilise Amazon Simple Notification Service (Amazon SNS) pour adresser une notification lorsqu'un événement Amazon RDS se produit. Ces notifications peuvent être faites sous n'importe quelle forme prise en charge par Amazon SNS pour une région AWS, telle qu'un e-mail, un SMS ou un appel à un point de terminaison HTTP.

Rubriques

Présentation des notifications d'événements Amazon RDS.

Amazon RDS regroupe les événements en catégories auxquelles vous pouvez vous abonner afin d'être informé lorsqu'un événement de cette catégorie se produit.

Rubriques

Ressources RDS éligibles à l’abonnement à un événement

Pour Amazon Aurora, les événements se produisent à la fois au niveau du cluster et de l’instance de base de données.Vous pouvez vous abonner à une catégorie d’événement pour les ressources suivantes :

- instance de base de données
- Cluster DB
- Instantané de cluster DB
- Groupe de paramètres de base de données
- Security Group DB
- RDS Proxy (Proxy RDS)
- Versions de moteur personnalisées

Par exemple, si vous vous abonnez à la catégorie de sauvegarde d’une instance de base de données donnée, vous recevez une notification chaque fois que survient un événement lié à la sauvegarde et qui affecte l’instance de base de données. Si vous vous abonnez à la catégorie de modification de configuration pour une instance de base de données, vous recevez une notification en cas de modification de l’instance de base de données. Vous recevez également une notification en cas de modification d’un abonnement à une notification d’événements.

Vous pouvez créer plusieurs abonnements différents. Par exemple, vous pouvez vouloir créer un abonnement qui reçoit toutes les notifications d’événements pour l’ensemble des instances de base de données, et un autre incluant uniquement les événements critiques pour un sous-ensemble des instances de base de données. Pour le deuxième abonnement, spécifiez une ou plusieurs instances de base de données dans le filtre.

Procédure de base pour s’abonner aux notifications d’événement Amazon RDS

La procédure d’abonnement à une notification d’événement Amazon RDS est la suivante :

1. Vous créez un abonnement à une notification d’événement Amazon RDS à l’aide de la console Amazon RDS, de la AWS CLI ou de l’API.

   Amazon RDS utilise l’ARN d’une rubrique Amazon SNS pour identifier chaque abonnement. La console Amazon RDS crée l’ARN lorsque vous créez l’abonnement. Créez l’ARN à l’aide de la console Amazon SNS, de la AWS CLI ou de l’API Amazon SNS.

2. Amazon RDS envoie un e-mail d'approbation ou un SMS aux adresses que vous avez fournies avec votre abonnement.
3. Pour confirmer votre abonnement, cliquez sur le lien dans la notification que vous avez reçue.
4. La console Amazon RDS met à jour la section My Event Subscriptions (Mes abonnements aux événements) avec le statut de votre abonnement.
5. Amazon RDS commence à envoyer les notifications aux adresses que vous avez fournies lors de la création de l'abonnement.

Pour en savoir plus sur la gestion des identités et des accès lors de l'utilisation d'Amazon SNS, consultez Gestion des identités et des accès dans Amazon SNS dans le Guide du développeur Amazon Simple Notification Service.

Vous pouvez utiliser AWS Lambda pour traiter les notifications d'événements à partir d'une instance de base de données. Pour plus d'informations, consultez Utilisation d'AWS Lambda avec Amazon RDS dans le Guide du développeur AWS Lambda.

Livraison des notifications d'événements RDS

Amazon RDS envoie les notifications d'événements aux adresses que vous fournissez lorsque vous créez l'abonnement. La notification peut inclure des attributs de message fournissant des métadonnées structurées relatives au message. Pour plus d'informations sur les attributs de message, consultez Catégories d'événements et messages d'événements pour Aurora.

Les notifications d'événement peuvent prendre jusqu'à cinq minutes pour être livrées.

> **Important**
>
> Amazon RDS ne garantie pas l'ordre des événements envoyés dans un flux d'événements. L'ordre des événements est susceptible de changer.

Lorsqu'Amazon SNS envoie une notification à un point de terminaison HTTP ou HTTPS abonné, le corps du message POST envoyé au point de terminaison contient un document JSON. Pour plus d'informations, consultez Formats de message et JSON Amazon SNS dans le Manuel du développeur Amazon Simple Notification Service.

Vous pouvez configurer SNS pour vous avertir avec des messages texte. Pour plus d'informations, consultez SMS dans le Guide du développeur Amazon Simple Notification Service.

Pour désactiver les notifications sans supprimer un abonnement, sélectionnez Non pour Activé dans la console Amazon RDS. Vous pouvez également définir le paramètre `Enabled` à `false` en utilisant la AWS CLI ou l'API Amazon RDS.

Facturation des notifications d'événement Amazon RDS

La facturation de la notification d'événement Amazon RDS s'effectue via Amazon SNS. Des frais Amazon SNS s'appliquent en cas d'utilisation de la notification d'événement. Pour plus d'informations sur la tarification Amazon SNS, consultez Tarification Amazon Simple Notification Service.

Exemples d'événements Aurora utilisant Amazon EventBridge

Les exemples suivants illustrent différents types d'événements Aurora au format JSON. Pour accéder à un tutoriel qui vous montre comment capturer et afficher les événements au format JSON, consultez Tutoriel : journaliser les changements d'état de l'instance de base de données à l'aide d'Amazon EventBridge.

Rubriques

- Exemple d'événement de cluster de bases de données
- Exemple d'événement de groupe de paramètres de base de données
- Exemple d'événement d'instantané de cluster de bases de données

Exemple d'événement de cluster de bases de données

Voici un exemple d'événement de cluster de bases de données au format JSON. L'événement montre que le cluster nommé `my-db-cluster` a été corrigé. L'ID de l'événement est `RDS-EVENT-0173`.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Cluster Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster"
  ],
```

```
  "detail": {
    "EventCategories": [
      "notification"
    ],
    "SourceType": "CLUSTER",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-db-cluster",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Database cluster has been patched",
    "SourceIdentifier": "my-db-cluster",
    "EventID": "RDS-EVENT-0173"
  }
}
```

Exemple d'événement de groupe de paramètres de base de données

Voici un exemple d'événement de groupe de paramètres de base de données au format JSON. L'événement indique que le paramètre `time_zone` a été mis à jour dans le groupe de paramètres `my-db-param-group`. L'ID de l'événement est RDS-EVENT-0037.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Parameter Group Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PARAM",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:pg:my-db-param-group",
    "Date": "2018-10-06T12:26:13.882Z",
    "Message": "Updated parameter time_zone to UTC with apply method immediate",
    "SourceIdentifier": "my-db-param-group",
    "EventID": "RDS-EVENT-0037"
  }
}
```

Exemple d’événement d’instantané de cluster de bases de données

Voici un exemple d’événement d’instantané de cluster de bases de données au format JSON. L’événement montre la création de l’instantané nommé `my-db-cluster-snapshot`. L’ID de l’événement est RDS-EVENT-0074.

```
{
  "version": "0",
  "id": "844e2571-85d4-695f-b930-0153b71dcb42",
  "detail-type": "RDS DB Cluster Snapshot Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-10-06T12:26:13Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:rds:us-east-1:123456789012:cluster-snapshot:rds:my-db-cluster-snapshot"
  ],
  "detail": {
    "EventCategories": [
      "backup"
    ],
    "SourceType": "CLUSTER_SNAPSHOT",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster-snapshot:rds:my-db-
cluster-snapshot",
    "Date": "2018-10-06T12:26:13.882Z",
    "SourceIdentifier": "my-db-cluster-snapshot",
    "Message": "Creating manual cluster snapshot",
    "EventID": "RDS-EVENT-0074"
  }
}
```

Octroi d’autorisations de publication de notifications dans une rubrique Amazon SNS

Pour autoriser Amazon RDS à publier des notifications sur une rubrique Amazon Simple Notification Service (Amazon SNS), associez une politique Gestion des identités et des accès AWS (IAM) à la rubrique de destination. Pour plus d’informations sur les autorisations, consultez Exemples de cas pour le contrôle d’accès Amazon Simple Notification Service dans le Guide du développeur Amazon Simple Notification Service.

Par défaut, une rubrique Amazon SNS dispose d’une politique permettant à toutes les ressources Amazon RDS d’un même compte d’y publier des notifications. Vous pouvez attacher une politique personnalisée pour autoriser les notifications entre comptes ou pour restreindre l’accès à certaines ressources.

Voici un exemple de politique IAM que vous attachez à la rubrique Amazon SNS de destination. Il limite la rubrique aux instances de base de données dont les noms correspondent au préfixe spécifié. Pour utiliser cette politique, spécifiez les valeurs suivantes :

- `Resource` : Amazon Resource Name (ARN) de votre rubrique Amazon SNS
- `SourceARN` : ARN de votre ressource RDS
- `SourceAccount`— Votre Compte AWS identifiant

Pour consulter la liste des types de ressources et leurs caractéristiques ARNs, consultez la section Ressources définies par Amazon RDS dans le Service Authorization Reference.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.rds.amazonaws.com"
      },
      "Action": [
        "sns:Publish"
      ],
      "Resource": "arn:aws:sns:us-east-1:123456789012:topic_name",
      "Condition": {
```

```
        "ArnLike": {
          "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:prefix-*"
        },
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

Abonnement à la notification d'événement Amazon RDS

La solution la plus simple pour créer un abonnement consiste à utiliser la console RDS. Si vous choisissez de créer un abonnement à une notification d'événement à l'aide de la CLI ou de l'API, vous devez créer une rubrique Amazon Simple Notification Service et vous abonner à cette rubrique avec la console Amazon SNS ou l'API Amazon SNS. Vous devrez également conserver l'Amazon Resource Name (ARN) de la rubrique, car il est utilisé lors de la soumission de commandes de la CLI ou d'opérations d'API. Pour plus d'informations sur la création d'une rubrique SNS et sur l'abonnement à cette rubrique, consultez Mise en route d'Amazon SNS dans le Manuel du développeur d'Amazon Simple Notification Service.

Vous pouvez spécifier le type de source dont vous voulez être informé et la source Amazon RDS qui déclenche l'événement :

Source type (Type de source)

Type de source. Par exemple, Source Type (Type de source) pourrait être Instances. Vous devez choisir un type de source.

Ressources à inclure

Les ressources Amazon RDS qui génèrent les événements. Par exemple, vous pouvez choisir Select specific instances (Sélectionner des instances spécifiques), puis myDBInstance1.

Le tableau suivant montre le résultat lorsque vous spécifiez ou ne spécifiez pas les ***ressources*** à inclure.

Ressources à inclure	Description	exemple
Spécifié	RDS vous notifie de tous les événements pour la ressource spécifiée uniquement.	Si votre Source type (Type de source) est Instances et que votre ressource est myDBInstance1, RDS vous notifie tous les événements pour myDBInstance1 uniquement.

Ressources à inclure	Description	exemple
Non spécifiée	RDS vous notifie les événements pour le type de source spécifié pour toutes vos ressources Amazon RDS.	Si votre Source type (Type de source) est Instances, RDS vous notifie tous les événement s liés aux instances dans votre compte.

Par défaut, un abonné d'une rubrique Amazon SNS reçoit chaque message publié dans la rubrique. Pour recevoir uniquement un sous-ensemble des messages, l'abonné doit attribuer une politique de filtre à l'abonnement à la rubrique. Pour plus d'informations sur le filtrage des messages SNS, consultez Filtrage des messages Amazon SNS dans le Guide du développeur Amazon Simple Notification Service.

Console

Pour s'abonner à la notification d'événement RDS

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Abonnements aux événements.
3. Dans le volet Abonnements aux événements, choisissez Créer un abonnement aux événements.
4. Entrez les détails de votre abonnement comme suit :

    a. Dans le champ Nom, entrez un nom pour l'abonnement à la notification d'événements.

    b. Pour Send notification to: (Envoyer les notifications à), effectuez l'une des opérations suivantes :

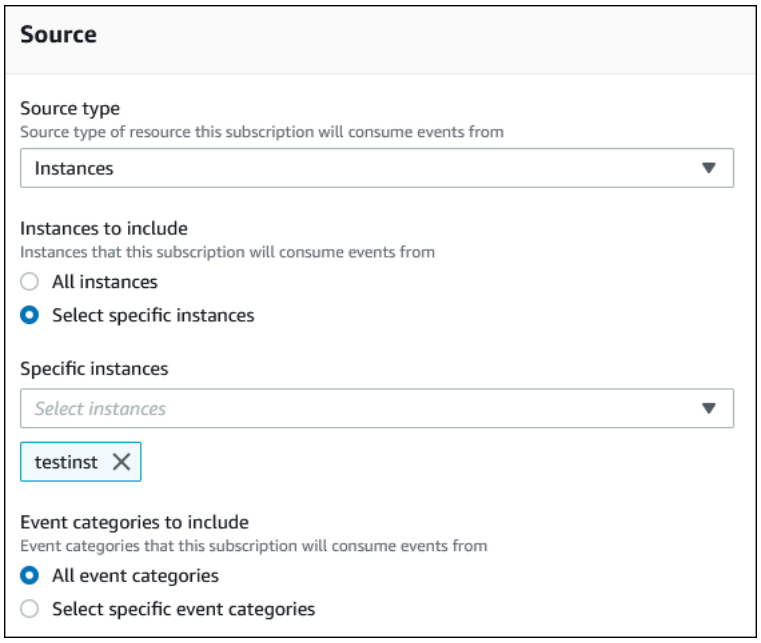
    - Choisissez New email topic (Nouvelle rubrique d'e-mail). Saisissez un nom pour votre rubrique d'e-mail et une liste de bénéficiaires. Nous vous recommandons de configurer les abonnements aux événements sur la même adresse e-mail que celle du contact principal de votre compte. Les recommandations, les événements de service et les messages de santé personnels sont envoyés via différents canaux. Les abonnements sur la même adresse e-mail garantissent que tous les messages sont regroupés en un seul endroit.
    - Choisissez Amazon Resource Name (ARN). Choisissez ensuite l'ARN Amazon SNS existant pour une rubrique Amazon SNS.

Si vous souhaitez utiliser une rubrique pour laquelle le chiffrement côté serveur (SSE) a été activé, accordez à Amazon RDS les autorisations nécessaires pour accéder à la AWS KMS key. Pour en savoir plus, consultez Activer la compatibilité entre des sources d'événements à partir de services AWS et de rubriques chiffrées dans le Guide du développeur Amazon Simple Notification Service.

c. Pour Type de source, choisissez un type de source. Par exemple, choisissez (Groupes de paramètres)Clusters ou Cluster snapshots (Instantanés de clusters).

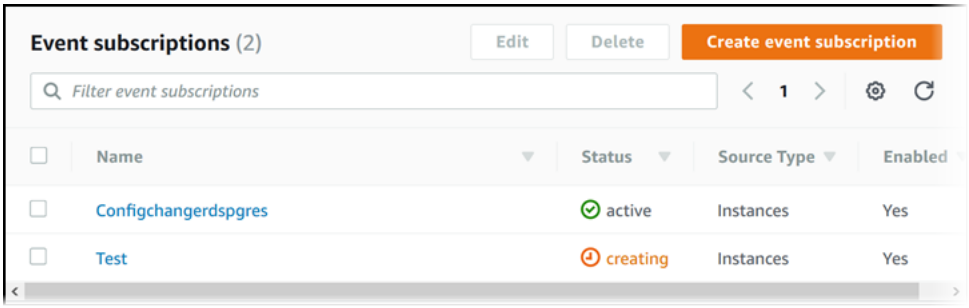
d. Choisissez les catégories d'événements et les ressources pour lesquelles vous souhaitez recevoir des notifications d'événements.

L'exemple suivant configure les notifications d'événements pour l'instance de base de données nommée `testinst`.



e. Choisissez Créer.

La console Amazon RDS indique que l'abonnement est en cours de création.



AWS CLI

Pour vous abonner à la notification d'événement RDS, utilisez la commande AWS CLI de l'`create-event-subscription`. Incluez les paramètres requis suivants :

- `--subscription-name`
- `--sns-topic-arn`

Example

Pour Linux, macOS ou Unix :

```
aws rds create-event-subscription \
    --subscription-name myeventsubscription \
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS \
    --enabled
```

Pour Windows :

```
aws rds create-event-subscription ^
    --subscription-name myeventsubscription ^
    --sns-topic-arn arn:aws:sns:us-east-1:123456789012:myawsuser-RDS ^
    --enabled
```

« Hello, World! »

Pour vous abonner à la notification d'événement Amazon RDS, appelez la fonction d'API Amazon RDS `CreateEventSubscription`. Incluez les paramètres requis suivants :

- `SubscriptionName`
- `SnsTopicArn`

Balises et attributs de notifications d'événements Amazon RDS

Lorsqu'Amazon RDS envoie une notification d'événement à Amazon Simple Notification Service (SNS) ou à Amazon EventBridge, la notification contient des attributs de message et des balises d'événement. RDS envoie les attributs du message séparément avec le message, tandis que les balises d'événement se trouvent dans le corps du message. Utilisez les attributs des messages et les balises Amazon RDS pour ajouter des métadonnées à vos ressources. Vous pouvez modifier ces balises avec vos propres notations sur les instances de base de données, les clusters Aurora, etc. Pour plus d'informations sur le balisage des ressources Amazon RDS, consultez Marquage des ressources Amazon Aurora et Amazon RDS.

Par défaut, Amazon SNS et Amazon EventBridge reçoivent tous les messages qui leur sont envoyés. SNS et EventBridge peuvent filtrer le message et envoyer des notifications au mode de communication de votre choix, tel qu'un e-mail, un SMS ou un appel à un point de terminaison HTTP.

> **Note**
>
> La notification envoyée par e-mail ou SMS ne comportera pas de balises d'événement.

La table suivante montre les attributs de message pour les événements RDS envoyés à l'abonné à la rubrique.

Attribut d'événement Amazon RDS	Description
EventID	Identifiant pour le message de l'événement RDS, par exemple, RDS-EVENT-0006.
Ressource	L'identifiant ARN de la ressource émettant l'événement, par exemple `arn:aws:rds:ap-southeast-2:123456789012:db:database-1` .

Les balises RDS fournissent des données sur la ressource affectée par l'événement de service. RDS ajoute l'état actuel des balises dans le corps du message lorsque la notification est envoyée à SNS ou EventBridge.

Pour plus d'informations sur le filtrage des attributs de message pour SNS, consultez Filtrage des messages Amazon SNS dans le Guide du développeur Amazon Simple Notification Service.

Pour plus d’informations sur le filtrage des balises d’événements pour EventBridge, consultez Opérateurs de comparaison à utiliser dans les modèles d’événements sur Amazon EventBridge dans le Guide de l’utilisateur Amazon EventBridge.

Pour plus d’informations sur le filtrage des balises basées sur les données utiles pour SNS, consultez. Présentation du filtrage des messages basé sur les données utiles pour Amazon SNS

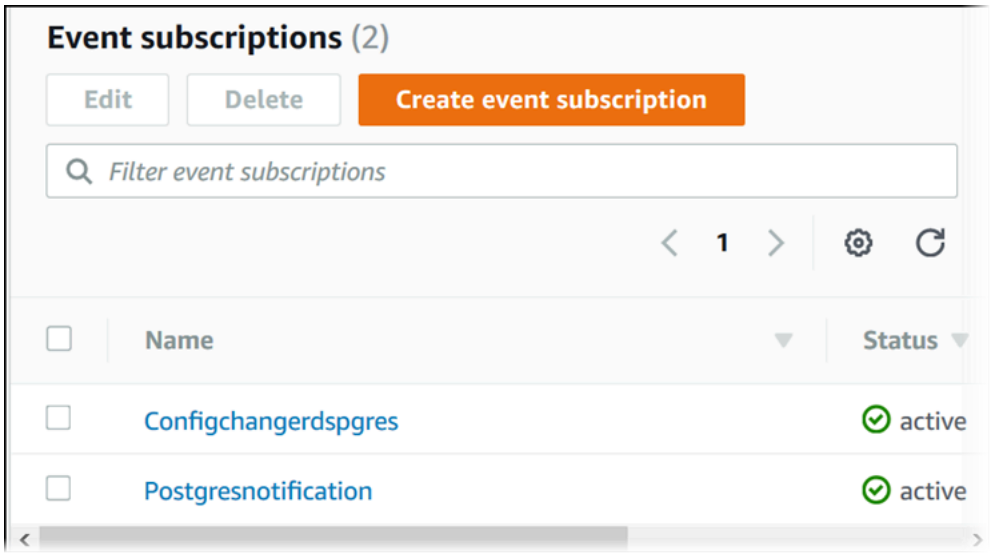
Liste des abonnements aux notifications d'événements Amazon RDS

Vous pouvez afficher vos abonnements aux notifications d'événement Amazon RDS.

Console

Pour afficher vos abonnements aux notifications d'événements Amazon RDS

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Abonnements aux événements. Le volet Abonnements aux événements affiche tous les abonnements aux notifications d'événements.



AWS CLI

Pour afficher vos abonnements aux notifications d'événements Amazon RDS, utilisez la commande de l'AWS CLI `describe-event-subscriptions`.

Example

L'exemple suivant décrit tous les abonnements aux événements.

```
aws rds describe-event-subscriptions
```

L'exemple suivant décrit l'abonnement `myfirsteventsubscription`.

```
aws rds describe-event-subscriptions --subscription-name myfirsteventsubscription
```

« Hello, World! »

Pour afficher vos abonnements aux notifications d'événements Amazon RDS, appelez l'action d'API Amazon RDS `DescribeEventSubscriptions`.

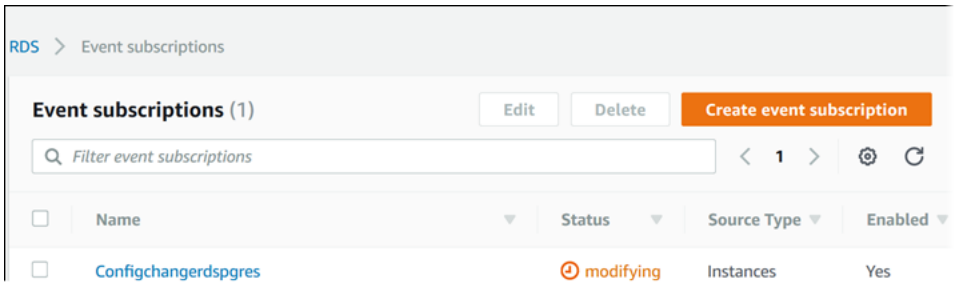
Modification d’un abonnement aux notifications d’événements Amazon RDS

Une fois que vous avez créé un abonnement, vous pouvez en changer le nom, l’identifiant de la source, les catégories ou l’ARN de la rubrique.

Console

Pour modifier un abonnement aux notifications d’événements Amazon RDS

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Abonnements aux événements.
3. Dans le volet Abonnements aux événements, choisissez l’abonnement que vous voulez modifier, puis choisissez Modifier.
4. Apportez les modifications requises à l’abonnement dans la section Cible ou Source.
5. Choisissez Edit. La console Amazon RDS indique que l’abonnement est en cours de modification.



AWS CLI

Pour modifier un abonnement aux notifications d’événements Amazon RDS, utilisez la commande de l’AWS CLI `modify-event-subscription`. Incluez le paramètre requis suivant :

- `--subscription-name`

Example

Le code suivant active `myeventsubscription`.

Pour Linux, macOS ou Unix :

```
aws rds modify-event-subscription \
    --subscription-name myeventsubscription \
    --enabled
```

Pour Windows :

```
aws rds modify-event-subscription ^
    --subscription-name myeventsubscription ^
    --enabled
```

« Hello, World! »

Pour modifier un événement Amazon RDS, appelez l'opération d'API Amazon RDS `ModifyEventSubscription`. Incluez le paramètre requis suivant :

- `SubscriptionName`

Ajout d'un identifiant source à un abonnement aux notifications d'événements Amazon RDS

Vous pouvez ajouter un identifiant source (la source Amazon RDS générant l'événement) à l'abonnement existant.

Console

Vous pouvez facilement ajouter ou supprimer des identifiants source à l'aide de la console Amazon RDS en les sélectionnant ou en annulant leur sélection lors de la modification d'un abonnement. Pour plus d'informations, consultez Modification d'un abonnement aux notifications d'événements Amazon RDS.

AWS CLI

Pour ajouter un identifiant de source à un abonnement aux notifications d'événements Amazon RDS, utilisez la commande de l'AWS CLI `add-source-identifier-to-subscription`. Incluez les paramètres requis suivants :

- `--subscription-name`
- `--source-identifier`

Example

L'exemple suivant ajoute l'identifiant source `mysqldb` à l'abonnement `myrdseventsubscription`

Pour Linux, macOS ou Unix :

```
aws rds add-source-identifier-to-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Pour Windows :

```
aws rds add-source-identifier-to-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

« Hello, World! »

Pour ajouter un identifiant source à un abonnement aux notifications d'événements Amazon RDS, appelez l'API Amazon RDS `AddSourceIdentifierToSubscription`. Incluez les paramètres requis suivants :

- `SubscriptionName`
- `SourceIdentifier`

Suppression d’un identifiant source d’un abonnement aux notifications d’événements Amazon RDS

Vous pouvez supprimer un identifiant source (la source Amazon RDS générant l’événement) d’un abonnement si vous ne souhaitez plus être informé des événements de cette source.

Console

Vous pouvez facilement ajouter ou supprimer des identifiants source à l’aide de la console Amazon RDS en les sélectionnant ou en annulant leur sélection lors de la modification d’un abonnement. Pour plus d’informations, consultez Modification d’un abonnement aux notifications d’événements Amazon RDS.

AWS CLI

Pour supprimer un identifiant source d’un abonnement aux notifications d’événements Amazon RDS, utilisez la commande de l’AWS CLI `remove-source-identifier-from-subscription`. Incluez les paramètres requis suivants :

- `--subscription-name`
- `--source-identifier`

Example

L’exemple suivant supprime l’identifiant source `mysqldb` de l’abonnement `myrdseventsubscription`.

Pour Linux, macOS ou Unix :

```
aws rds remove-source-identifier-from-subscription \
    --subscription-name myrdseventsubscription \
    --source-identifier mysqldb
```

Pour Windows :

```
aws rds remove-source-identifier-from-subscription ^
    --subscription-name myrdseventsubscription ^
    --source-identifier mysqldb
```

« Hello, World! »

Pour supprimer un identifiant source d’un abonnement aux notifications d’événements Amazon RDS, utilisez la commande d’API `RemoveSourceIdentifierFromSubscription` de l’Amazon RDS. Incluez les paramètres requis suivants :

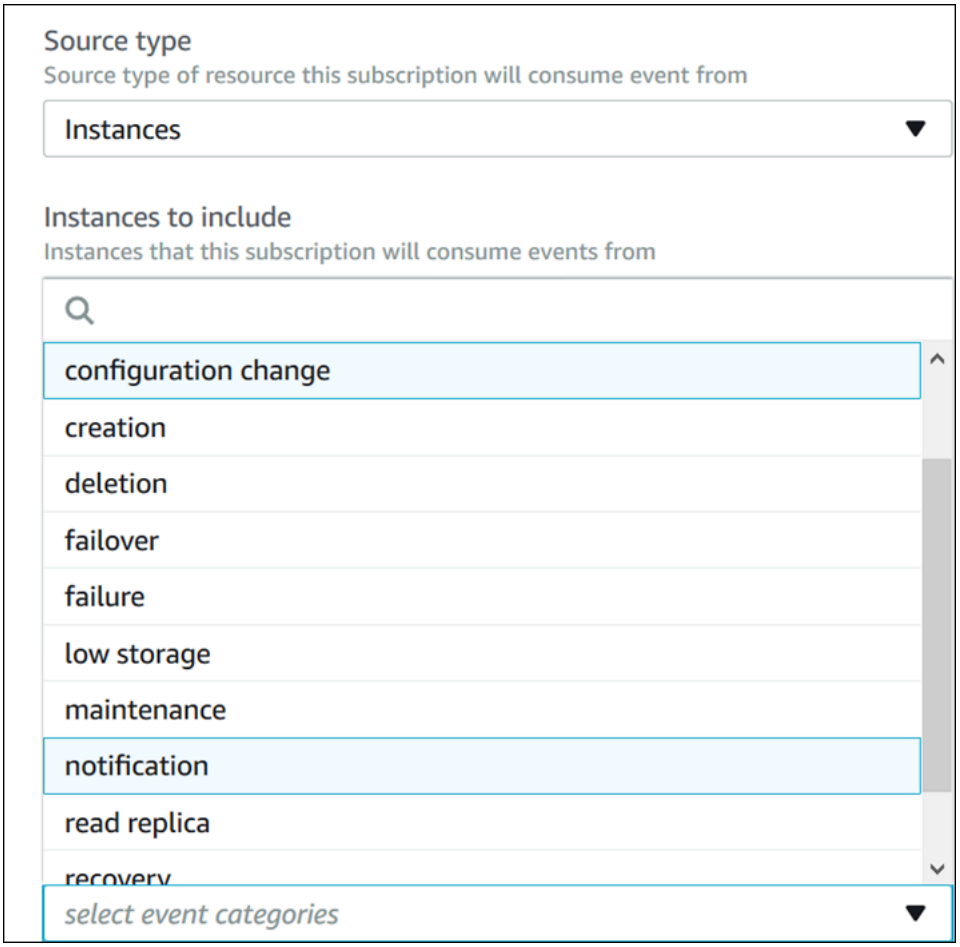
- `SubscriptionName`
- `SourceIdentifier`

Affichage des catégories aux notifications d'événements Amazon RDS

Tous les événements d'un type de ressource sont regroupés en catégories. Pour afficher la liste des catégories disponibles, utilisez les procédures suivantes.

Console

Lorsque vous créez ou modifiez un abonnement aux notifications d'événements, les catégories d'événements sont affichées dans la console Amazon RDS. Pour plus d'informations, consultez Modification d'un abonnement aux notifications d'événements Amazon RDS.



AWS CLI

Pour lister les catégories de notifications d'événements Amazon RDS, utilisez la commande de l'AWS CLI `describe-event-categories`. Cette commande n'a aucun paramètre requis.

Example

```
aws rds describe-event-categories
```

« Hello, World! »

Pour lister les catégories de notifications d’événements Amazon RDS, utilisez la commande d’API `DescribeEventCategories` de l’Amazon RDS. Cette commande n’a aucun paramètre requis.

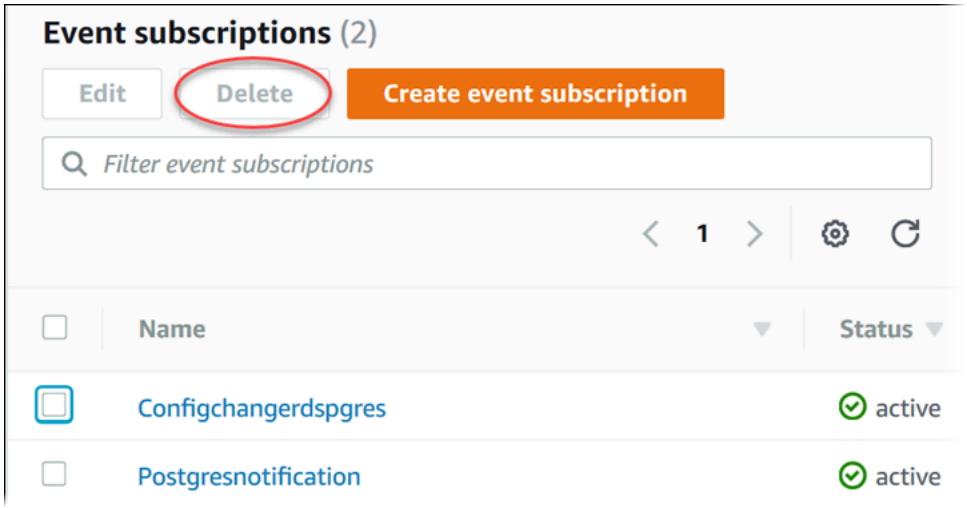
Suppression d’un abonnement aux notifications d’événements Amazon RDS

Vous pouvez supprimer un abonnement lorsque vous n’en avez plus besoin. Tous les abonnés à la rubrique ne reçoivent plus les notifications d’événements spécifiées par l’abonnement.

Console

Pour supprimer un abonnement aux notifications d’événements Amazon RDS

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez DB Event Subscriptions (Abonnements aux événements de base de données).
3. Dans le volet My DB Event Subscriptions (Mes abonnements aux événements de base de données), cliquez sur l’abonnement que vous souhaitez supprimer.
4. Sélectionnez Delete.
5. La console Amazon RDS indique que l’abonnement est en cours de suppression.



AWS CLI

Pour supprimer un abonnement aux notifications d’événements Amazon RDS, utilisez la commande de l’AWS CLI `delete-event-subscription`. Incluez le paramètre requis suivant :

- `--subscription-name`

Example

L'exemple suivant supprime l'abonnement `myrdssubscription`.

```
aws rds delete-event-subscription --subscription-name myrdssubscription
```

« Hello, World! »

Pour supprimer un abonnement aux notifications d'événements Amazon RDS, utilisez la commande `DeleteEventSubscription` de l'API. Incluez le paramètre requis suivant :

- `SubscriptionName`

Création d’une règle qui se déclenche sur un événement Amazon Aurora

Grâce à Amazon EventBridge, vous pouvez automatiser les services AWS et répondre à des événements système tels que des problèmes de disponibilité d’application ou des modifications de ressources.

Rubriques

Tutoriel : journaliser les changements d’état de l’instance de base de données à l’aide d’Amazon EventBridge

Dans ce tutoriel, vous créez une fonction AWS Lambda qui enregistre les changements d’état d’une instance . Vous créez ensuite une règle qui exécute la fonction chaque fois qu’il y a un changement d’état d’une instance de base de données RDS existante. Le didacticiel suppose que vous avez d’une petite instance de test en cours d’exécution, que vous pouvez arrêter momentanément.

> **Important**
>
> N’appliquez pas ce tutoriel à une instance de base de données de production en cours d’exécution.

Rubriques

Étape 1 : création d’une fonction AWS Lambda

Créez une fonction Lambda pour enregistrer les événements de changement d’état. Vous spécifiez cette fonction lors de la création de votre règle.

Pour créer une fonction Lambda

1. Ouvrez la console AWS Lambda à l’adresse https://console.aws.amazon.com/lambda/.

2. Si vous utilisez Lambda pour la première fois, une page de bienvenue s’affiche. Sélectionnez Pour commencer. Sinon, choisissez Créer la fonction.
3. Choisissez Créer à partir de scratch.
4. Sur la page Create function (Créer une fonction), procédez de la façon suivante :

    a. Saisissez un nom et une description pour la fonction Lambda. Par exemple, nommez la fonction **`RDSInstanceStateChange`**.

    b. Dans Runtime, sélectionnez Node.js 16x.

    c. Pour Architecture, choisissez x86_64.

    d. Pour Execution role (Rôle d’exécution), effectuez l’une des opérations suivantes :

    - Choisissez Create a new role with basic Lambda permissions (Créer un rôle avec les autorisations Lambda standard).
    - Pour Existing role (Rôle existant), sélectionnez Use an existing role (Utiliser un rôle existant). Choisissez le rôle que vous voulez utiliser.

    e. Choisissez Créer une fonction.
5. Sur la page RDSInstanceStateChange, procédez comme suit :

    a. Dans Code source (Source de code), sélectionnez index.js.

    b. Dans le panneau index.js, supprimez le code existant.

    c. Saisissez le code suivant :

```
console.log('Loading function');

exports.handler = async (event, context) => {
    console.log('Received event:', JSON.stringify(event));
};
```

    d. Choisissez Deploy (Déployer).

Étape 2 : création d’une règle

Créez une règle pour exécuter votre fonction Lambda chaque fois que vous lancez une instance Amazon RDS.

Pour créer la règle EventBridge

1. Ouvrez la console Amazon EventBridge à l'adresse https://console.aws.amazon.com/events/.
2. Dans le panneau de navigation, choisissez Rules.
3. Choisissez Create rule.
4. Saisissez un nom et une description pour la règle. Par exemple, entrez **`RDSInstanceStateChangeRule`**.
5. Sélectionnez Rule with an event pattern (Règle avec un modèle d'événement), puis sélectionnez Next (Suivant).
6. Pour Source d'événement, choisissez Événements AWS ou événements partenaires EventBridge.
7. Faites défiler la page vers le bas jusqu'à la section Event pattern (Modèle d'événement).
8. Pour Event source (Source d'événement), choisissez Services AWS.
9. Pour Service AWS, choisissez Relational Database Service (RDS).
10. Pour Event type (Type d'événement), sélectionnez RDS DB Instance Event (événement d'instance de base de données RDS).
11. Laissez le modèle d'événement par défaut. Ensuite, sélectionnez Suivant.
12. Pour Types de cibles, choisissez service AWS.
13. Pour Select a target (Sélectionner une cible), choisissez Lambda Function (Fonction Lambda).
14. Dans Function (Fonction), choisissez la fonction Lambda que vous avez créée. Ensuite, sélectionnez Suivant.
15. Dans la rubrique Configure tags (Configurer les balises), sélectionnez Next (Suivant).
16. Passez en revue les étapes de votre règle. Puis, choisissez Create rule (Créer une règle).

Étape 3 : test de la règle

Pour tester votre règle, arrêtez une instance de base de données RDS. Après avoir attendu quelques minutes que l'instance s'arrête, vous pouvez vérifier que votre fonction Lambda a été appelée.

Pour tester votre règle en arrêtant une instance de base de données

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Arrêter une instance de base de données RDS.
3. Ouvrez la console Amazon EventBridge à l'adresse https://console.aws.amazon.com/events/.

4. Dans le panneau de navigation, cliquez sur Rules (Règles), puis sur le nom de la règle que vous avez créée.
5. Dans Détails des règles, choisissez Surveillance.

   Vous êtes ensuite redirigé vers la console Amazon CloudWatch. Si vous n'êtes pas redirigé, cliquez sur Afficher les métriques dans CloudWatch.
6. Dans All metrics (Toutes les métriques), cliquez sur le nom de la règle que vous avez créée.

   Le graphique doit indiquer que la règle a été appelée.
7. Dans le panneau de navigation, sélectionnez Groupes de journaux.
8. Cliquez sur le nom du groupe de journaux pour votre fonction Lambda (/aws/lambda/***function-name***).
9. Choisissez le nom du flux de journaux pour afficher les données fournies par la fonction concernant l'instance que vous avez lancée. Vous devez voir un événement reçu semblable à ce qui suit :

```
{
    "version": "0",
    "id": "12a345b6-78c9-01d2-34e5-123f4ghi5j6k",
    "detail-type": "RDS DB Instance Event",
    "source": "aws.rds",
    "account": "111111111111",
    "time": "2021-03-19T19:34:09Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:rds:us-east-1:111111111111:db:testdb"
    ],
    "detail": {
        "EventCategories": [
            "notification"
        ],
        "SourceType": "DB_INSTANCE",
        "SourceArn": "arn:aws:rds:us-east-1:111111111111:db:testdb",
        "Date": "2021-03-19T19:34:09.293Z",
        "Message": "DB instance stopped",
        "SourceIdentifier": "testdb",
        "EventID": "RDS-EVENT-0087"
    }
}
```

Pour voir plus d’exemples d’événements RDS au format JSON, consultez Présentation des événements pour Aurora.

10. (Facultatif) Lorsque vous avez terminé, vous pouvez ouvrir la console Amazon RDS et lancer l’instance que vous avez arrêtée.

Catégories d’événements et messages d’événements pour Aurora

Amazon RDS génère un nombre important d'événements dans des catégories auxquelles vous pouvez vous abonner à l'aide de la console Amazon RDS ou de l'API. AWS CLI

Rubriques

Événements de cluster de bases de données

Le tableau suivant affiche la catégorie d’événement et la liste des événements lorsqu’un cluster de bases de données est le type source.

> **Note**
>
> Aucune catégorie d’événement n’existe pour Aurora Serverless dans le type d’événement de Cluster de bases de données. Les événements Aurora sans serveur vont de RDS-EVENT-0141 à RDS-EVENT-0149.

Catégorie	ID d’événement RDS	Message	Remarques
modification de configuration	RDS-EVENT-0016	Réinitialisation des informations d’identification principales.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
modification de configuration	RDS-EVENT-0179	Les flux d'activité de base de données sont démarrés sur votre cluster de bases de données.	Pour plus d'informations, consultez Surveillance d'Amazon Aurora à l'aide des flux d'activité de base de données.
modification de configuration	RDS-EVENT-0180	Les flux d'activité de base de données sont arrêtés sur votre cluster de bases de données.	Pour plus d'informations, consultez Surveillance d'Amazon Aurora à l'aide des flux d'activité de base de données.
création	RDS-EVENT-0170	cluster de bases de données créé.	Aucune
suppression	RDS-EVENT-0171	cluster de bases de données supprimé.	Aucune
basculement	RDS-EVENT-0069	Le basculement du cluster a échoué. Vérifiez l'état de santé de vos instances de cluster et réessayez.	Aucune
basculement	RDS-EVENT-0070	Promouvoir à nouveau la primaire précédente :*name*.	Aucune
basculement	RDS-EVENT-0071	Basculement terminé vers l'instance de base de données :*name*.	Aucune
basculement	RDS-EVENT-0072	Démarrage du même basculement AZ vers l'instance de base de données :*name*.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
basculement	RDS-EVENT-0073	Basculement cross-AZ lancé vers une instance de base de données :*name*.	Aucune
échec	RDS-EVENT-0083	Amazon RDS n'a pas pu créer d'informations d'identification pour accéder à votre compartiment Amazon S3 pour votre cluster *name* de base de données. Cela est dû au fait que le rôle IAM d'ingestion d'instantanés S3 n'est pas correctement configuré dans votre compte ou que le compartiment Amazon S3 spécifié est introuvable. Reportez-vous à la section sur la résolution des problèmes dans la documentation Amazon RDS pour plus de détails.	Pour plus d'informations, consultez Migration physique depuis MySQL à l'aide de Percona XtraBackup et Amazon S3.
échec	RDS-EVENT-0143	Le cluster de base de données n'a pas pu *units* passer de à *units* pour cette raison :*reason*.	Échec du dimensionnement pour le cluster de bases de données Aurora Serverless.
échec	RDS-EVENT-0354	Vous ne pouvez pas créer le cluster de base de données en raison de ressources incompatibles. *message*.	*message*Cela inclut des détails sur la panne.

Catégorie	ID d’événement RDS	Message	Remarques
échec	RDS-EVENT-0355	Le cluster de base de données ne peut pas être créé en raison de limites de ressources insuffisantes. *message*.	*message*Cela inclut des détails sur la panne.
échec	RDS-EVENT-0387	Impossible de partitionner les instances de base de données dans le cluster de base de données *name* pour l'application de correctifs.	Les mises à niveau du système d’exploitation pour les instances de base de données du cluster de bases de données ont échoué.
basculement global	RDS-EVENT-0181	Le passage global au cluster de base de données *name* dans la région a commencé. *name*	Cet événement se rapporte à une opération de bascule (précédemment appelée « basculement planifié géré »). Le processus peut être retardé, car d’autres opérations sont en cours d’exécution sur le cluster de bases de données.

Catégorie	ID d'événement RDS	Message	Remarques
basculement global	RDS-EVENT-0182	L'ancien cluster de base de données principal *name* de Region s'est *name* correctement arrêté.	Cet événement se rapporte à une opération de bascule (précédemment appelée « basculement planifié géré »). L'ancienne instance principale de la base de données globale n'accepte pas les écritures. Tous les volumes sont synchronisés.
basculement global	RDS-EVENT-0183	En attente de la synchronisation des données entre les membres du cluster global. Retards actuels par rapport au cluster de bases de données principal : *reason*.	Cet événement se rapporte à une opération de bascule (précédemment appelée « basculement planifié géré »). Un retard de réplication se produit pendant la phase de synchronisation du basculement global de la base de données.
basculement global	RDS-EVENT-0184	Le nouveau cluster de base de données principal *name* de Region *name* a été promu avec succès.	Cet événement se rapporte à une opération de bascule (précédemment appelée « basculement planifié géré »). La topologie de volume de la base de données globale est rétablie avec le nouveau volume principal.

Catégorie	ID d'événement RDS	Message	Remarques
basculement global	RDS-EVENT-0185	Le passage global au cluster de base de données *name* dans la région est terminé. *name*	Cet événement se rapporte à une opération de bascule (précédemment appelée « basculement planifié géré »). La bascule globale des bases de données est terminée sur le cluster de base de données principal . La mise en ligne des réplicas peut prendre beaucoup de temps une fois le basculement terminé.
basculement global	RDS-EVENT-0186	Le passage global au cluster de base de données *name* dans Region *name* est annulé.	Cet événement se rapporte à une opération de bascule (précédemment appelée « basculement planifié géré »).
basculement global	RDS-EVENT-0187	Le passage global au cluster de base de données *name* dans Region a échoué. *name*	Cet événement se rapporte à une opération de bascule (précédemment appelée « basculement planifié géré »).
basculement global	RDS-EVENT-0238	Le basculement global vers le cluster de base de données de *name* la région *name* est terminé.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
basculement global	RDS-EVENT-0239	Le basculement global vers le cluster *name* de base de données de Region *name* a échoué.	Aucune
basculement global	RDS-EVENT-0240	La resynchronisation des membres du cluster de base de données *name* dans Region a commencé *name* après un basculement global.	Aucune
basculement global	RDS-EVENT-0241	La resynchronisation des membres du cluster de base de données *name* dans Region est terminée *name* après un basculement global.	Aucune
basculement global	RDS-EVENT-0397	Aurora a fini de modifier le nom DNS utilisé par le point de terminaison d'enregistreur.	Aucune
basculement global	RDS-EVENT-0423	En attente de synchronisation des données avec le cluster de bases de données cible. Le cluster de bases de données cible actuel est en retard par rapport au cluster de bases de données principal : `%s`.	Cet événement se rapporte à une opération de bascule (précédemment appelée « basculement planifié géré »). Un retard de réplication se produit pendant la phase de synchronisation du basculement global de la base de données.

Catégorie	ID d'événement RDS	Message	Remarques
maintenance	RDS-EVENT-0156	Une mise à niveau de version mineure du moteur de base de données est disponible pour le cluster de bases de données.	Aucune
maintenance	RDS-EVENT-0173	La version du moteur de cluster de bases de données a été mise à niveau.	Les correctifs ont été appliqués au cluster de bases de données.
maintenance	RDS-EVENT-0174	Le cluster de bases de données est dans un état qui ne peut pas être mis à niveau.	Aucune
maintenance	RDS-EVENT-0176	La version majeure du moteur de cluster de bases de données a été mise à niveau.	Aucune
maintenance	RDS-EVENT-0177	La mise à niveau du cluster de bases de données est en cours.	Aucune
maintenance	RDS-EVENT-0286	La mise à niveau de *version_number* la version du moteur du cluster de base de données Le cluster reste en ligne.	Aucune
maintenance	RDS-EVENT-0287	Une mise à niveau nécessaire du système d'exploitation a été détectée.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
maintenance	RDS-EVENT-0288	La mise à niveau du système d'exploitation du cluster est en cours de démarrage.	Aucune
maintenance	RDS-EVENT-0289	La mise à niveau du système d'exploitation du cluster est terminée.	Aucune
maintenance	RDS-EVENT-0290	Le cluster de base de données a été corrigé : version source *version_number* =>*new_version_number* .	Aucune
maintenance	RDS-EVENT-0363	Préparation de la mise à niveau en cours : *cluster_name*	Les prévérifications de mise à niveau ont commencé pour le cluster de bases de données.
maintenance	RDS-EVENT-0388	Démarrage de correctifs hors ligne pour les instances de base de données de la partition *name*/*name*pour le cluster de base de données *name* :*partition_n* .	Démarrage des mises à niveau du système d'exploitation pour les instances de base de données du cluster de bases de données.

Catégorie	ID d'événement RDS	Message	Remarques
maintenance	RDS-EVENT-0389	Nous n'avons pas pu mettre à niveau le système d'exploitation de votre cluster de bases de données. Vous pouvez attendre la fenêtre de maintenance suivante ou vous pouvez mettre à niveau le système d'exploitation de votre cluster de bases de données manuellement.	Aucune
maintenance	RDS-EVENT-0424	Le cluster *name* de base de données exécute une version *version* supérieure à la version de mise à niveau cible *version* pour le cluster global. Nous ne recommandons pas d'avoir un cluster secondaire sur une version supérieure à celle du cluster global, car cela peut entraîner des problèmes lors du basculement ou de la bascule. Envisagez de mettre à niveau votre cluster global en conséquence.	Aucune
notification	RDS-EVENT-0076	Impossible de migrer de *name* vers*name*. Motif :*reason*.	La migration vers un cluster de bases de données Aurora a échoué.

Catégorie	ID d'événement RDS	Message	Remarques
notification	RDS-EVENT-0077	Impossible de convertir *name*. *name*à InnoDB. Motif :*reason*.	Une tentative de conversion d'une table de la base de données source en InnoDB a échoué lors de la migration vers un cluster de bases de données Aurora.
notification	RDS-EVENT-0085	Impossible de mettre à niveau le cluster de base de données *name* car l'instance *name* a un statut de*name*. Résolvez le problème ou supprimez l'instance et réessayez.	Une erreur s'est produite lors de la tentative de corriger le cluster de bases de données Aurora. Vérifiez le statut de votre instance, résolvez le problème et réessayez . Pour plus d'informations, consultez Entretien d'un cluster de bases de données Amazon Aurora.
notification	RDS-EVENT-0141	Dimensionnement du cluster de base de données de *units* à *units* pour cette raison :*reason*.	Dimensionnement initié pour le cluster de bases de données Aurora Serverless.
notification	RDS EVENT-0142	Le cluster de base de données est passé de *units* à. *units*	Dimensionnement terminé pour le cluster de bases de données Aurora Serverless.
notification	RDS EVENT-0144	Le cluster de bases de données est mis en pause.	Une pause automatique a été initiée pour le cluster de bases de données Aurora sans serveur.

Catégorie	ID d'événement RDS	Message	Remarques
notification	RDS-EVENT-0145	Le cluster de bases de données est mis en pause.	Le cluster de bases de données Aurora Serverless a été mis en pause.
notification	RDS-EVENT-0146	La pause a été annulée pour le cluster de bases de données.	La pause a été annulée pour le cluster de bases de données Aurora Serverless.
notification	RDS-EVENT-0147	Le cluster de bases de données est en cours de reprise.	Une opération de reprise a été initiée pour le cluster de bases de données Aurora Serverless.
notification	RDS EVENT-0148	Le cluster de bases de données a repris.	L'opération de reprise pour le cluster de bases de données Aurora Serverless est terminée.
notification	RDS-EVENT-0149	Le cluster de base de données est passé de *`units`* à*`units`*, mais le dimensionnement n'a pas été fluide pour cette raison :*`reason`*.	Dimensionnement transparent terminé avec l'option force pour le cluster de bases de données Aurora Serverless. Les connexions peuvent avoir été interrompues en fonction des besoins.
notification	RDS-EVENT-0150	Le cluster de bases de données s'est arrêté.	Aucune
notification	RDS-EVENT-0151	Le cluster de bases de données a démarré.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
notification	RDS-EVENT-0152	L'arrêt du cluster de bases de données a échoué.	Aucune
notification	RDS-EVENT-0153	Le cluster de bases de données est en cours de démarrage dans la mesure où il a dépassé le temps maximum autorisé pour son arrêt.	Aucune
notification	RDS-EVENT-0172	Cluster renommé de *name* à*name*.	Aucune
notification	RDS-EVENT-0234	Échec de la tâche d'exportation.	La tâche d'exportation de cluster de bases de données a échoué.
notification	RDS-EVENT-0235	Annulation de la tâche d'exportation.	La tâche d'exportation de cluster de bases de données a été annulée.
notification	RDS-EVENT-0236	Tâche d'exportation terminée.	La tâche d'exportation de cluster de bases de données est terminée.
notification	RDS-EVENT-0386	Les instances de base de données du cluster de base de données *name* ont été partitionnées :*list_of_partitions* . Le cluster de bases de données est en ligne.	Les mises à niveau du système d'exploitation pour les instances de base de données du cluster de bases de données ont abouti.

Catégorie	ID d'événement RDS	Message	Remarques
notification	RDS-EVENT-0426	RDS ne peut pas configure r le cluster de base de données *name* comme source de réplication en raison de transactions inactives ou de longue durée. Attendez que les transactions soient terminées ou annulez-les, puis réessayez.	Aucune

Événements d'instantané de cluster de bases de données

Le tableau suivant affiche la catégorie d'événement et la liste des événements lorsqu'un instantané de cluster de bases de données est le type source.

Catégorie	ID d'événement RDS	Message	Remarques
sauvegarde	RDS-EVENT-0074	Création d'un instantané de cluster manuel.	Aucune
sauvegarde	RDS-EVENT-0075	Instantané de cluster manuel créé.	Aucune
notification	RDS-EVENT-0162	La tâche d'exportation de l'instantané de cluster a échoué.	Aucune
notification	RDS-EVENT-0163	La tâche d'exportation de l'instantané de cluster a été annulée.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
notification	RDS-EVENT-0164	La tâche d'exportation de l'instantané de cluster est terminée.	Aucune
sauvegarde	RDS-EVENT-0168	Création d'instantané de cluster automatisé.	Aucune
sauvegarde	RDS-EVENT-0169	Instantané de cluster automatisé créé.	Aucune
échec	RDS-EVENT-0489	Tâche d'exportation *name* : L'extraction de la table a échoué *name* en raison de*message*.	Cet événement n'est pas pris en charge par une exportation plus rapide.
notification	RDS-EVENT-0491	Tâche d'exportation *name* : La table *name* sera exportée dans un mode monothread plus lent. Veuillez vous attendre à un retard dans la finalisation de l'exportation.	Cet événement n'est pas pris en charge par une exportation plus rapide.
notification	RDS-EVENT-0492	Tâche d'exportation *name* : le tableau *name* est en cours et la taille exportée est de *number* Go.	Cet événement ne sera envoyé que pour les tables dont la réalisation prend plus d'une heure. Cet événement n'est pas pris en charge par une exportation plus rapide.
notification	RDS-EVENT-0493	La tâche d'exportation *name* est en *name* cours.	Aucune

Catégorie	ID d’événement RDS	Message	Remarques
		Tâche d'exportation *name* : la progression de l'exportation est de *number* % avec *number* les *number* tables en cours et les tables terminées. La taille totale des tables finies est de *number* Go.	Cet événement n'est pas pris en charge par une exportation plus rapide.

Événements d’instance de base de données

Le tableau suivant recense la catégorie d’événement et la liste des événements lorsqu’une instance de base de données est le type source.

Catégorie	ID d’événement RDS	Message	Remarques
disponibilité	RDS-EVENT-0004	L’instance de base de données s’est arrêtée.	Aucune
disponibilité	RDS-EVENT-0006	Instance de base de données redémarrée.	Aucune
disponibilité	RDS-EVENT-0022	Erreur lors du redémarrage de MySQL :*message*.	Une erreur s’est produite lors du redémarrage d’Aurora MySQL ou de RDS for MariaDB.
disponibilité	RDS-EVENT-0419	Amazon RDS n'a pas pu accéder à la clé de KMS chiffrement de l'instance *name* de base de données. Cette base de données sera placée dans un état	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
		inaccessible. Reportez-vous à la section sur la résolution des problèmes dans la documentation Amazon RDS pour plus de détails.	
retour sur trace	RDS-EVENT-0131	La fenêtre de retour sur trace réelle est plus petite que la fenêtre de retour sur trace cible que vous avez spécifiée. Envisagez de réduire le nombre d'heures de votre fenêtre de retour sur trace cible.	Pour plus d'informations sur le retour sur trace, consultez Retour en arrière d'un cluster de bases de données Aurora.
retour sur trace	RDS-EVENT-0132	La fenêtre de retour sur trace réelle est identique à la fenêtre de retour sur trace cible.	Aucune
modification de configuration	RDS-EVENT-0011	Mis à jour pour utiliser DBParameter le groupe*name*.	Aucune
modification de configuration	RDS-EVENT-0012	Application de la modification à la classe d'instance de base de données.	Aucune
modification de configuration	RDS-EVENT-0014	Fin de l'application de la modification à la classe d'instance de base de données.	Aucune

Catégorie	ID d’événement RDS	Message	Remarques
modification de configuration	RDS-EVENT-0017	Fin de l’application de la modification au stockage alloué.	Aucune
modification de configuration	RDS-EVENT-0025	Fin de l’application de la modification pour effectuer une conversion vers une instance de base de données multi-AZ.	Aucune
modification de configuration	RDS-EVENT-0029	Fin de l’application de la modification pour effectuer une conversion vers une instance de base de données standard (mono-AZ).	Aucune
modification de configuration	RDS-EVENT-0033	Certains *number* utilisateurs correspondent au nom d'utilisateur principal ; seuls ceux qui ne sont pas liés à un hôte spécifique sont réinitialisés.	Aucune
modification de configuration	RDS-EVENT-0067	Réinitialisation de votre mot de passe impossible. Informations sur les erreurs :*message*.	Aucune
modification de configuration	RDS-EVENT-0078	L'intervalle de surveillance a été modifié en*number*.	La configuration Supervision améliorée a été modifiée.

Catégorie	ID d’événement RDS	Message	Remarques
modification de configuration	RDS-EVENT-0092	Fin de la mise à jour du groupe de paramètres de la base de données.	Aucune
création	RDS-EVENT-0005	Instance de base de données créée.	Aucune
suppression	RDS-EVENT-0003	Instance de base de données supprimée.	Aucune
échec	RDS-EVENT-0035	Instance de base de données insérée dans*state*. *message*.	L’instance de base de données a des paramètres non valides. Par exemple, si l’instance de base de données n’a pas pu démarrer, un paramètre lié à la mémoire étant défini avec une valeur trop élevée pour cette classe d’instance, votre action consiste à modifier le paramètre et à redémarrer l’instance de base de données.
échec	RDS-EVENT-0036	Instance de base de données dans*state*. *message*.	L’instance de base de données se trouve sur un réseau non compatible. Certains sous-réseaux spécifiés ne IDs sont pas valides ou n'existent pas.

Catégorie	ID d'événement RDS	Message	Remarques
échec	RDS-EVENT-0079	Amazon RDS n'a pas été en mesure de créer des informations d'identification pour une surveillance améliorée. Cette fonction a été désactivée. Cela est probablement dû au fait que votre compte rds-monitoring-role n'est pas présent et qu'il n'est pas configuré correctement. Reportez-vous à la section sur la résolution des problèmes dans la documentation Amazon RDS pour plus de détails.	La supervision améliorée ne peut pas être activée sans le rôle IAM de surveillance améliorée. Pour obtenir des informations sur la création du rôle IAM, consultez Pour créer un rôle IAM pour la surveillance améliorée Amazon RDS.
échec	RDS-EVENT-0080	Amazon RDS n'a pas pu configurer la surveillance améliorée sur votre instance : *name* cette fonctionnalité a été désactivée. Cela est probablement dû au fait que votre compte rds-monitoring-role n'est pas présent et qu'il n'est pas configuré correctement. Reportez-vous à la section sur la résolution des problèmes dans la documentation Amazon RDS pour plus de détails.	La surveillance améliorée a été désactivée en raison d'une erreur lors de la modification de la configuration. Il est probable que le rôle IAM de surveillance améliorée ne soit pas configuré correctement. Pour obtenir des informations sur la création du rôle IAM de surveillance améliorée, consultez Pour créer un rôle IAM pour la surveillance améliorée Amazon RDS.

Catégorie	ID d’événement RDS	Message	Remarques
échec	RDS-EVENT-0082	Amazon RDS n'a pas pu créer d'informations d'identification pour accéder à votre compartiment Amazon S3 pour votre instance *name* de base de données. Cela est dû au fait que le rôle IAM d’ingestion d’instantanés S3 n’est pas correctement configuré dans votre compte ou que le compartiment Amazon S3 spécifié est introuvable. Reportez-vous à la section sur la résolution des problèmes dans la documentation Amazon RDS pour plus de détails.	Aurora n’a pas pu copier les données de sauvegarde d’un compartiment Amazon S3. Il est probable que les autorisations devant permettre à Aurora d’accéder au compartiment Amazon S3 soient mal configurées. Pour plus d’informations, consultez Migration physique depuis MySQL à l'aide de Percona XtraBackup et Amazon S3.
échec	RDS-EVENT-0353	L'instance de base de données ne peut pas être créée en raison de limites de ressources insuffisantes. *message*.	*message*Cela inclut des détails sur la panne.

Catégorie	ID d’événement RDS	Message	Remarques
échec	RDS-EVENT-0418	Amazon RDS ne parvient pas à accéder à la clé de KMS chiffrement de l'instance *name* de base de données. Cela est probablement dû à la désactivation de la clé ou à l’impossibilité pour Amazon RDS d’y accéder. Si cela continue, la base de données sera placée dans un état inaccessible. Reportez-vous à la section sur la résolution des problèmes dans la documentation Amazon RDS pour plus de détails.	Aucune
échec	RDS-EVENT-0420	Amazon RDS peut désormais accéder avec succès à la clé de KMS chiffrement de l'instance *name* de base de données.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
stockage faible	RDS-EVENT-0007	L'espace de stockage alloué est épuisé. Allouez du stockage supplémentaire pour résoudre le problème.	L'espace de stockage alloué pour l'instance de base de données a été consommé. Pour résoudre ce problème, allouez de l'espace de stockage supplémentaire à l'instance de base de données. Pour plus d'informations, consultez la FAQ RDS. Vous pouvez surveiller l'espace de stockage pour une instance de base de données à l'aide de la métrique Free Storage Space (Espace de stockage libre).
stockage faible	RDS-EVENT-0089	La capacité de stockage gratuite pour l'instance de base de données : *percentage* de *name* est faible par rapport au stockage provisionné [Stockage provisionné :*size*, Stockage gratuit :*size*]. Vous pouvez augmenter le stockage provisionné pour résoudre ce problème.	L'instance de base de données a consommé plus de 90 % de son stockage alloué. Vous pouvez surveiller l'espace de stockage pour une instance de base de données à l'aide de la métrique Free Storage Space (Espace de stockage libre).

Catégorie	ID d’événement RDS	Message	Remarques
stockage faible	RDS-EVENT-0227	L'espace de stockage de votre cluster Aurora est dangereusement bas, il ne reste que des *amount* téraoctets. Veuillez prendre des mesures pour réduire la charge de stockage de votre cluster.	Le sous-système de stockage Aurora manque d’espace.
maintenance	RDS-EVENT-0026	Application de correctifs hors ligne à l’instance de base de données.	La maintenance hors connexion de l’instance de base de données est en cours. L’instance de base de données n’est pas disponible actuellement.
maintenance	RDS-EVENT-0027	Application de correctifs hors ligne à l’instance de base de données terminée.	La maintenance hors connexion de l’instanc e de base de données est terminée. L’instance de base de données est désormais disponible.
maintenance	RDS-EVENT-0047	Instance de base de données corrigée.	Aucune
maintenance	RDS-EVENT-0155	Une mise à niveau de version mineure du moteur de base de données est disponible pour l’instance de base de données.	Aucune
maintenance	RDS-EVENT-0178	La mise à niveau de l’instance de base de données est en cours.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
maintenance	RDS-EVENT-0422	RDS remplacera l'hôte de l'instance de base de données *name* en raison d'une action de maintenance en attente.	Aucune
notification	RDS-EVENT-0044	*message*	Il s'agit d'une notification émise par l'opérateur. Pour plus d'informations, consultez le message de l'événement.
notification	RDS-EVENT-0048	La mise à niveau du moteur de base de données est retardée, car cette instance comporte des réplicas en lecture qui doivent d'abord être mis à niveau.	L'application des correctifs de l'instance de base de données a été retardée.
notification	RDS-EVENT-0087	Instance de base de données arrêtée.	Aucune
notification	RDS-EVENT-0088	Instance de base de données démarrée.	Aucune
notification	RDS-EVENT-0365	Les fichiers de fuseau horaire ont été mis à jour. Redémarrez votre instance RDS pour que les modifications prennent effet.	Aucune

Catégorie	ID d’événement RDS	Message	Remarques
notification, sans serveur	RDS-EVENT-0370	Pause initiée pour l’instanc e de base de données.	Une nouvelle tentative de suspension d’une instance de base de données Aurora Serverless v2 inactive a été lancée.
notification, sans serveur	RDS-EVENT-0371	La pause a été annulée pour l’instance de base de données.	Une tentative de suspensio n d’une instance de base de données Aurora Serverless v2 inactive a échoué en raison de la charge de travail.
notification, sans serveur	RDS-EVENT-0372	L’instance de base de données a été suspendue avec succès.	L’instance de base de données Aurora Serverless v2 a été mise en pause.
notification, sans serveur	RDS-EVENT-0373	Reprise initiée pour l’instance de base de données.	L’instance de base de données Aurora Serverles s v2 a commencé à reprendre en raison d’une nouvelle charge de travail ou d’une nouvelle activité administrative ou de maintenance.
notification, sans serveur	RDS-EVENT-0374	Reprise de l’instance de base de données avec succès.	L’instance de base de données Aurora Serverless v2 a repris.

Catégorie	ID d'événement RDS	Message	Remarques
notification	RDS-EVENT-0385	La topologie du cluster est mise à jour.	Des modifications de DNS ont été apportées au cluster de bases de données pour l'instance de base de données. Cela inclut lorsque de nouvelles instances de base de données sont ajoutées ou supprimées, ou en cas de basculement.
notification, base de données globale	RDS-EVENT-0390	La tentative de blocage des écritures pour le cluster de base de données *cluster_id* dans Region *region_id* a réussi.	Aurora a commencé à bloquer les écritures au niveau de la couche de stockage en vue du basculement ou de la bascule d'une base de données globale Aurora.
notification, base de données globale	RDS-EVENT-0391	La tentative de blocage des écritures pour le cluster de base de données *cluster_id* dans Region *region_id* a expiré.	Aurora n'a pas pu bloquer les écritures au niveau de la couche de stockage en vue du basculement ou de la bascule d'une base de données globale Aurora. Le basculement ou la bascule se poursuivra, mais vous devrez peut-être récupérer les données récemment écrites à partir de l'instant ané du cluster principal d'origine.

Catégorie	ID d'événement RDS	Message	Remarques
réplica en lecture	RDS-EVENT-0045	La réplication s'est arrêtée.	Ce message s'affiche lorsqu'il y a une erreur lors de la réplication. Pour déterminer le type d'erreur, consultez Troubleshooting a MySQL read replica problem.
réplica en lecture	RDS-EVENT-0046	Reprise de la réplication pour le réplica en lecture.	Ce message s'affiche lorsque vous créez un réplica en lecture, ou comme message de surveillance lorsque vous confirmez que la réplication fonctionne correctement. Si ce message fait suite à une notification `RDS-EVENT-0045` , la réplication a repris suite à une erreur ou à un arrêt de la réplication.
réplica en lecture	RDS-EVENT-0057	Le streaming de réplication a été suspendu.	Aucune
récupération	RDS-EVENT-0020	La récupération de l'instance de base de données a démarré. Le temps de récupération varie selon la quantité de données à restaurer.	Aucune
ponctuelle	RDS-EVENT-0021	La récupération de l'instance de base de données est terminée.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
ponctuelle	RDS-EVENT-0023	Demande de capture instantanée urgente :*message*.	Une sauvegarde manuelle a été demandée, mais Amazon RDS est actuellement en cours de création d'un instantané de base de données. Soumettez à nouveau la demande après qu'Amazon RDS a terminé l'instantané de base de données.
récupération	RDS-EVENT-0052	La récupération de l'instance multi-AZ a démarré.	Le temps de récupération varie selon la quantité de données à restaurer.
récupération	RDS-EVENT-0053	La récupération de l'instance multi-AZ est terminée. En attente de basculement ou d'activation.	Ce message indique qu'Amazon RDS a préparé votre instance de base de données pour lancer un basculement vers l'instance secondaire si nécessaire.
ponctuelle	RDS-EVENT-0361	La récupération de l'instance de base de données de secours a démarré.	L'instance de base de données de secours est régénérée pendant le processus de régénération. Les performances de la base de données sont affectées pendant le processus de régénération.

Catégorie	ID d’événement RDS	Message	Remarques
ponctuelle	RDS-EVENT-0362	La régénération de l’instance de base de données de secours est terminée.	L’instance de base de données de secours est régénérée pendant le processus de régénération. Les performances de la base de données sont affectées pendant le processus de régénération.
restauration	RDS-EVENT-0019	Restauré depuis une instance de base de données *name* vers*name*.	L'instance de base de données a été restaurée à partir d'une point-in-time sauvegarde.
application de correctifs de sécurité	RDS-EVENT-0230	La mise à jour du système est disponible pour votre instance de base de données. Pour obtenir des informations sur l’application des mises à niveau, consultez « Entretien d’une instance de base de données » dans le Guide de l’utilisateur RDS.	Un nouveau correctif mineur du système d’exploitation est disponible pour votre instance de base de données. Pour obtenir des informations sur l’application de mises à jour, consultez Mises à jour du système d’exploitation pour les clusters de bases de données Aurora.

Catégorie	ID d’événement RDS	Message	Remarques
maintenance	RDS-EVENT-0425	Amazon RDS ne peut pas effectuer de mise à niveau du système d’exploitation, car aucune adresse IP n’est disponible dans les sous-réseaux spécifiés. Choisissez des sous-réseaux avec des adresses IP disponibles et réessayez.	Aucune
maintenance	RDS-EVENT-0429	Amazon RDS ne peut pas effectuer la mise à niveau du système d'exploitation en raison d'une capacité insuffisante disponible pour le type d'*type*instance dans la zone de *zone* disponibilité	Aucune

Catégorie	ID d’événement RDS	Message	Remarques
maintenance	RDS-EVENT-0501	Le certificat de serveur de l'instance de base de données Amazon RDS nécessite une rotation dans le cadre d'une action de maintenance en attente.	Le certificat de serveur de l'instance de base de données nécessite une rotation dans le cadre d'une action de maintenance en attente. Amazon RDS redémarre votre base de données pendant cette maintenance pour terminer la rotation des certifica ts. Pour planifier cette maintenance, allez dans l'onglet Maintenance et sauvegardes et choisisse z Appliquer maintenant ou Planifier pour la fenêtre de maintenance suivante. Si la modification n'est pas planifiée, Amazon RDS l'applique automatiq uement dans votre fenêtre de maintenance à la date d'application automatique indiquée dans votre action de maintenance.

Catégorie	ID d'événement RDS	Message	Remarques
maintenance	RDS-EVENT-0502	Amazon RDS a planifié une rotation des certificats de serveur pour l'instance de base de données lors de la prochaine fenêtre de maintenance. Cette maintenance nécessitera le redémarrage de la base de données.	Aucune

Événements de groupe de paramètres de base de données

Le tableau suivant affiche la catégorie d'événement et la liste des événements lorsqu'un groupe de paramètres de base de données est le type source.

Catégorie	ID d'événement RDS	Message	Remarques
modification de configuration	RDS-EVENT-0037	Paramètre mis *name* à jour *value* avec méthode d'application*method*.	Aucune

Événements de groupe de sécurité de base de données

Le tableau suivant affiche la catégorie d'événement et la liste des événements lorsqu'un groupe de sécurité de base de données est le type source.

> **Note**
>
> Les groupes de sécurité de base de données sont des ressources pour EC2 -Classic. EC2-Classic a été retiré le 15 août 2022. Si vous n'avez pas migré de EC2 -Classic vers un VPC, nous vous recommandons de migrer dès que possible. Pour plus d'informations, consultez

Migrer de EC2 -Classic vers un VPC dans le guide de l'utilisateur EC2 Amazon et sur le EC2blog -Classic Networking is Retiring — Here's How to Prepare.

Catégorie	ID d'événement RDS	Message	Remarques
modification de configuration	RDS-EVENT-0038	Modification appliquée au groupe de sécurité.	Aucune
échec	RDS-EVENT-0039	Révocation de l'autorisation en tant que*user*.	Le groupe de sécurité détenu par *user* n'existe pas. L'autorisation pour le groupe de sécurité a été révoquée, car elle n'est pas valide.

Événements de groupe de partitions de base de données

Le tableau suivant affiche la catégorie d'événement et la liste des événements lorsqu'un groupe de partitions de base de données est le type source.

Catégorie	ID d'événement RDS	Message	Remarques
échec	RDS-EVENT-0324	La tâche de chargement des données a échoué.	La tâche de chargement des données a échoué pour la raison indiquée dans le message d'erreur.
maintenance	RDS-EVENT-0302	Votre action est requise pour finaliser une tâche de division de partition en attente *job_ID* pour l'ID de sous-cluster *number* dans	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
		le groupe de partitions de base de données. *name* Pour terminer l'opération, invoquez le code SQL : `SELECT rds_aurora.limitless_finalize_split_shard( job_ID);`	
maintenance	RDS-EVENT-0303	La tâche de finalisation de la partition fractionnée *job_ID* a commencé pour l'ID du sous-cluster *number* dans le groupe de partitions de base de données. *name*	Aucune
maintenance	RDS-EVENT-0304	La tâche de fractionnement de la partition *job_ID* s'est terminée avec succès pour l'ID du sous-cluster *number* dans le groupe de partitions de base de données. *name* Une nouvelle partition avec un ID de sous-cluster *number* a été ajoutée au groupe de partitions de base de données. *name*	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
maintenance	RDS-EVENT-0305	La tâche de fractionnement de la partition *job_ID* a échoué pour l'ID du sous-cluster *number* dans le groupe de partitions de base de données. *name*	Aucune
maintenance	RDS-EVENT-0366	La tâche de fractionnement de la partition *job_ID* a commencé pour l'ID du sous-cluster *number* dans le groupe de partitions de base de données. *name*	Aucune
maintenance	RDS-EVENT-0367	*job_ID*La tâche d'ajout d'un routeur a commencé dans le groupe *name* de partitions de base de données.	Aucune
maintenance	RDS-EVENT-0368	La tâche d'ajout d'un routeur *job_ID* s'est terminée avec succès. Un nouveau routeur avec un ID de sous-cluster *number* a été ajouté au groupe de partitions de base de données. *name*	Aucune
maintenance	RDS-EVENT-0369	*job_ID*La tâche d'ajout d'un routeur a échoué dans le groupe *name* de partition s de base de données.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
maintenance	RDS-EVENT-0394	*job_ID*La tâche d'ajout d'un routeur a été annulée dans le groupe *name* de partitions de base de données.	Aucune
maintenance	RDS-EVENT-0395	La tâche Split Shard *job_ID* a été annulée dans le groupe de partitions de base de données. *name*	Aucune
notification	RDS-EVENT-0321	Initialisation de l'infrast ructure pour la tâche de chargement de données.	Aucune
notification	RDS-EVENT-0322	La tâche de chargement des données est en cours.	Aucune
notification	RDS-EVENT-0323	La tâche de chargement des données s'est terminée avec succès.	Aucune
notification	RDS-EVENT-0325	Annulation du travail de chargement de données conformément à la demande du client.	Aucune
notification	RDS-EVENT-0326	Tâche de chargement de données annulée.	Aucune

Événements RDS Proxy

Le tableau suivant recense la catégorie d'événement et la liste des événements lorsqu'un proxy RDS Proxy est le type source.

Catégorie	ID d'événement RDS	Message	Remarques
modification de configuration	RDS-EVENT-0204	Proxy *name* de base de données modifié par RDS.	Aucune
modification de configuration	RDS-EVENT-0207	RDS a modifié le point final du proxy *name* de base de données.	Aucune
modification de configuration	RDS-EVENT-0213	RDS a détecté l'ajout de l'instance de base de données et l'a automatiquement ajoutée au groupe cible du proxy *name* de base de données.	Aucune
modification de configuration	RDS-EVENT-0214	RDS a détecté la suppression de l'instance de base de données *name* et l'a automatiquement supprimée du groupe cible du proxy *name name* de base de données.	Aucune
modification de configuration	RDS-EVENT-0215	RDS a détecté la suppression du cluster de base de données *name* et l'a automatiquement supprimé du groupe cible du proxy *name name* de base de données.	Aucune
création	RDS-EVENT-0203	RDS a créé un proxy *name* de base de données.	Aucune

Catégorie	ID d'événement RDS	Message	Remarques
création	RDS-EVENT-0206	RDS a créé un point de terminaison *name* pour le proxy *name* de base de données.	Aucune
suppression	RDS-EVENT-0205	RDS a supprimé le proxy *name* de base de données.	Aucune
suppression	RDS-EVENT-0208	RDS a supprimé le point de terminaison *name* pour le proxy *name* de base de données.	Aucune
échec	RDS-EVENT-0243	RDS n'a pas pu fournir de capacité pour le proxy *name* car il n'y a pas assez d'adresses IP disponibl es dans vos sous-rése aux :. *name* Pour résoudre ce problème, veillez à ce que vos sous-réseaux aient le nombre minimum d'adresses IP inutilisées, comme recommandé dans la documentation Proxy RDS.	Pour déterminer le nombre recommandé pour votre classe d'instances, consultez Planification de la capacité des adresses IP.
échec	RDS-EVENT-0275	RDS a limité certaines connexions au proxy de base de données. *name* Le nombre de demandes de connexion simultanée du client au proxy a dépassé la limite.	Aucune

Événements de déploiement bleu/vert

Le tableau suivant indique la catégorie d'événements et la liste des événements dont le type source est un blue/green déploiement.

Pour plus d'informations sur les blue/green déploiements, consultezUtilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données.

Catégorie	ID d'événement Amazon RDS	Message	Remarques
création	RDS-EVENT-0244	Les tâches de déploiement bleu/vert sont terminées. Vous pouvez apporter plus de modifications aux bases de données de l'environnement vert ou effectuer un basculement du déploiement.	Aucune
échec	RDS-EVENT-0245	La création du blue/green déploiement a échoué car*reason*.	Aucune
suppression	RDS-EVENT-0246	Le déploiement bleu/vert a été supprimé.	Aucune
notification	RDS-EVENT-0247	Basculer entre le mode de démarrage et le mode *blue* de démarrage. *green*	Aucune
notification	RDS-EVENT-0248	Le passage au numérique a été effectué lors blue/green du déploiement.	Aucune
échec	RDS-EVENT-0249	Le passage au numérique a été annulé lors blue/green du déploiement.	Aucune

Catégorie	ID d'événement Amazon RDS	Message	Remarques
notification	RDS-EVENT-0259	Basculement du cluster de base de données de au cluster démarré. *blue green*	Aucune
notification	RDS-EVENT-0260	Passage du cluster de base de données au cluster terminé. *blue green* Renommé *blue blue-old* en et *green* en*blue*.	Aucune
échec	RDS-EVENT-0261	Le passage du cluster de base de données de au cluster de base de données *green* a été annulé *blue* en raison de. *reason*	Aucune
notification	RDS-EVENT-0311	La synchronisation de séquence pour le passage du cluster de base de données *blue* à *green* a été initiée. La bascule lors de l'utilisation de séquences peut entraîner une durée d'indisponibilité prolongée.	Aucune
notification	RDS-EVENT-0312	La synchronisation de séquence pour le passage du cluster de base de données *blue* à *green* est terminée.	Aucune

Catégorie	ID d'événement Amazon RDS	Message	Remarques
échec	RDS-EVENT-0314	La synchronisation des séquences pour le passage du cluster de base de données *blue* à *green* a été annulée car les séquences n'ont pas pu être synchronisées.	Aucune
notification	RDS-EVENT-0409	*message*	Aucune

Surveillance des fichiers journaux Amazon Aurora

Chaque moteur de base de données RDS génère des journaux auxquels vous pouvez accéder pour l'audit et le dépannage. Le type de journaux dépend de votre moteur de base de données.

Vous pouvez accéder aux journaux de base de données pour les instances de base de données à l'aide de la AWS Management Console, de l'AWS Command Line Interface (AWS CLI) ou de l'API Amazon RDS. Vous ne pouvez pas afficher, visualiser ou télécharger les journaux des transactions.

> **Note**
>
> Dans certains cas, les journaux contiennent des données cachées. L'AWS Management Console peut donc afficher du contenu dans un fichier journal et le fichier journal peut s'avérer vide lorsque vous le téléchargez.

Rubriques

Liste et affichage des fichiers journaux de base de données

Vous pouvez afficher les fichiers journaux de la base de données de votre moteur de base de données Amazon Aurora à l'aide de la commande AWS Management Console. Vous pouvez répertorier les fichiers journaux disponibles pour téléchargement ou surveillance à l'aide de l'AWS CLI ou de l'API Amazon RDS.

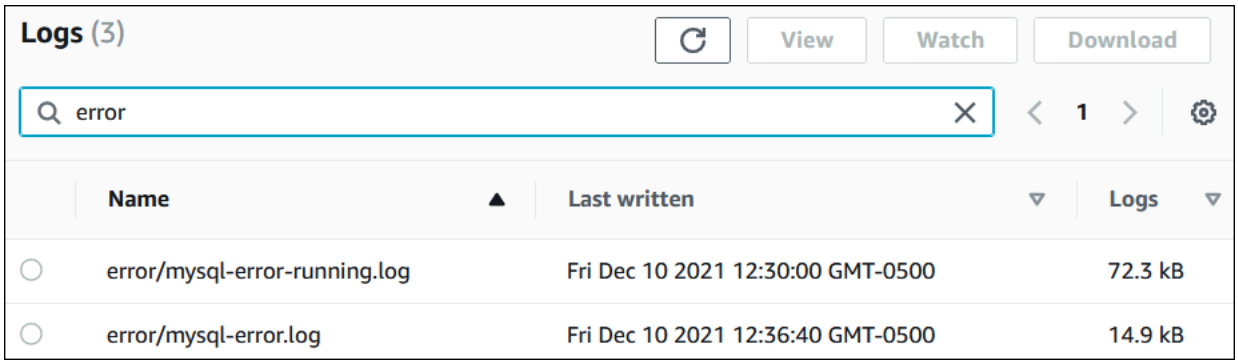
> **Note**
>
> Vous ne pouvez pas afficher les fichiers journaux des clusters de bases de données Aurora Serverless v1 dans la console RDS. Vous pouvez toutefois les visualiser dans la console Amazon CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.

Console

Pour afficher un fichier journal de base de données

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le nom de l'instance de base de données qui contient le fichier journal que vous voulez consulter.
4. Choisissez l'onglet Logs & events (Journaux et événements).
5. Faites défiler jusqu'à la section Journaux.
6. (Facultatif) Entrez un terme de recherche pour filtrer vos résultats.

   L'exemple suivant liste les journaux filtrés par l'élément de texte **`error`**.



7. Sélectionnez le journal que vous souhaitez afficher, puis cliquez sur View (Afficher).

AWS CLI

Pour répertorier les fichiers journaux de base de données disponibles pour une instance de base de données, utilisez la commande AWS CLI de l'`describe-db-log-files`.

L'exemple suivant renvoie une liste des fichiers journaux pour une instance DB nommée `my-db-instance`.

Example

```
aws rds describe-db-log-files --db-instance-identifier my-db-instance
```

API RDS

Pour répertorier les fichiers journaux de base de données disponibles pour une instance de base de données, utilisez l'action `DescribeDBLogFiles` de l'API Amazon RDS.

Téléchargement d'un fichier journal de base de données

Vous pouvez utiliser la AWS Management Console, AWS CLI ou l'API pour télécharger un fichier journal de base de données.

Console

Pour télécharger un fichier journal de base de données

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le nom de l'instance de base de données qui contient le fichier journal que vous voulez consulter.
4. Choisissez l'onglet Logs & events (Journaux et événements).
5. Faites défiler jusqu'à la section Journaux.
6. Dans la section Journaux, sélectionnez le bouton en regard du journal que vous voulez télécharger, puis choisissez Télécharger.
7. Ouvrez le menu contextuel (clic droit) pour le lien fourni, puis choisissez Enregistrer le lien sous. Saisissez l'emplacement souhaité pour l'enregistrement du fichier journal, puis cliquez sur Enregistrer.

AWS CLI

Pour télécharger un fichier journal de base de données, utilisez la commande AWS CLI de l'`download-db-log-file-portion`. Par défaut, cette commande télécharge uniquement la portion la plus récente d'un fichier journal. Vous pouvez toutefois télécharger un fichier complet en spécifiant le paramètre `--starting-token 0`.

L'exemple suivant montre comment télécharger le contenu d'un fichier journal appelé log/ERROR.4 et le stocker dans un fichier local appelé errorlog.txt.

Example

Pour Linux, macOS ou Unix :

```
aws rds download-db-log-file-portion \
    --db-instance-identifier myexampledb \
    --starting-token 0 --output text \
    --log-file-name log/ERROR.4 > errorlog.txt
```

Pour Windows :

```
aws rds download-db-log-file-portion ^
    --db-instance-identifier myexampledb ^
    --starting-token 0 --output text ^
    --log-file-name log/ERROR.4 > errorlog.txt
```

API RDS

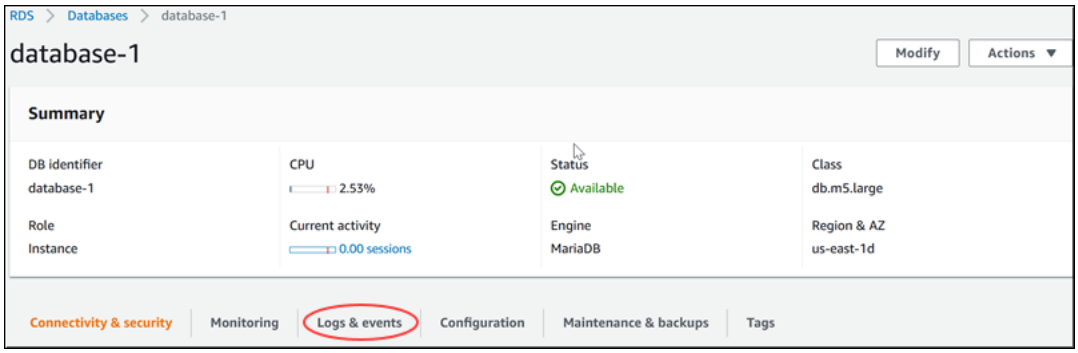
Pour télécharger un fichier journal de base de données, utilisez l'action `DownloadDBLogFilePortion` de l'API Amazon RDS.

Consultation d'un fichier journal de base de données

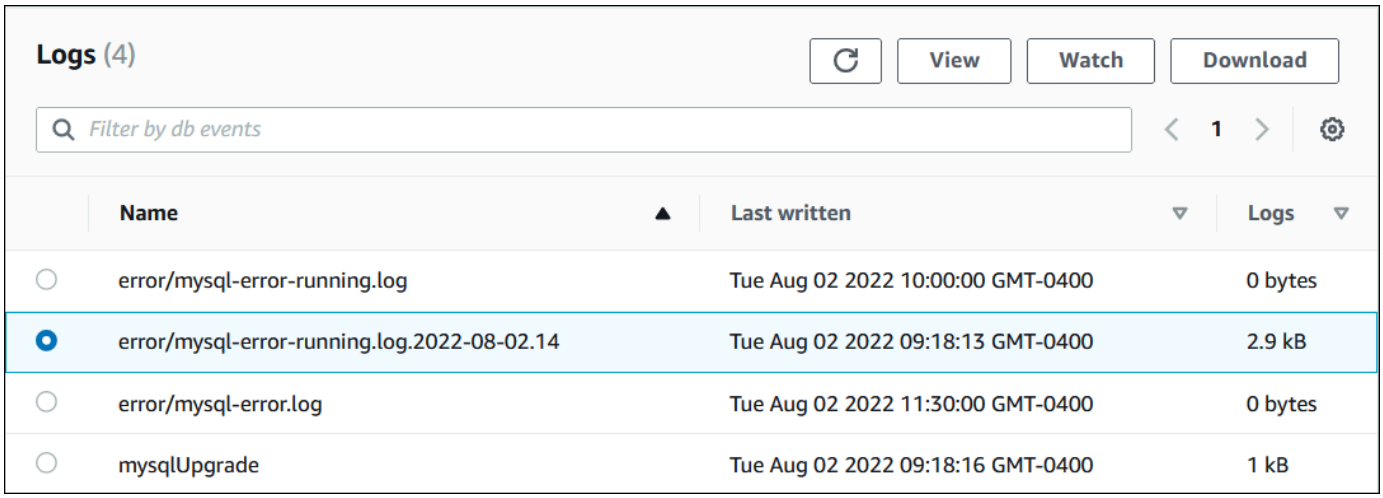
Surveiller un fichier journal de base de données revient à suivre le fichier sur un système UNIX ou Linux. Vous pouvez afficher un fichier journal en utilisant la AWS Management Console. RDS rafraîchit la queue du journal toutes les cinq secondes.

Pour consulter un fichier journal de base de données

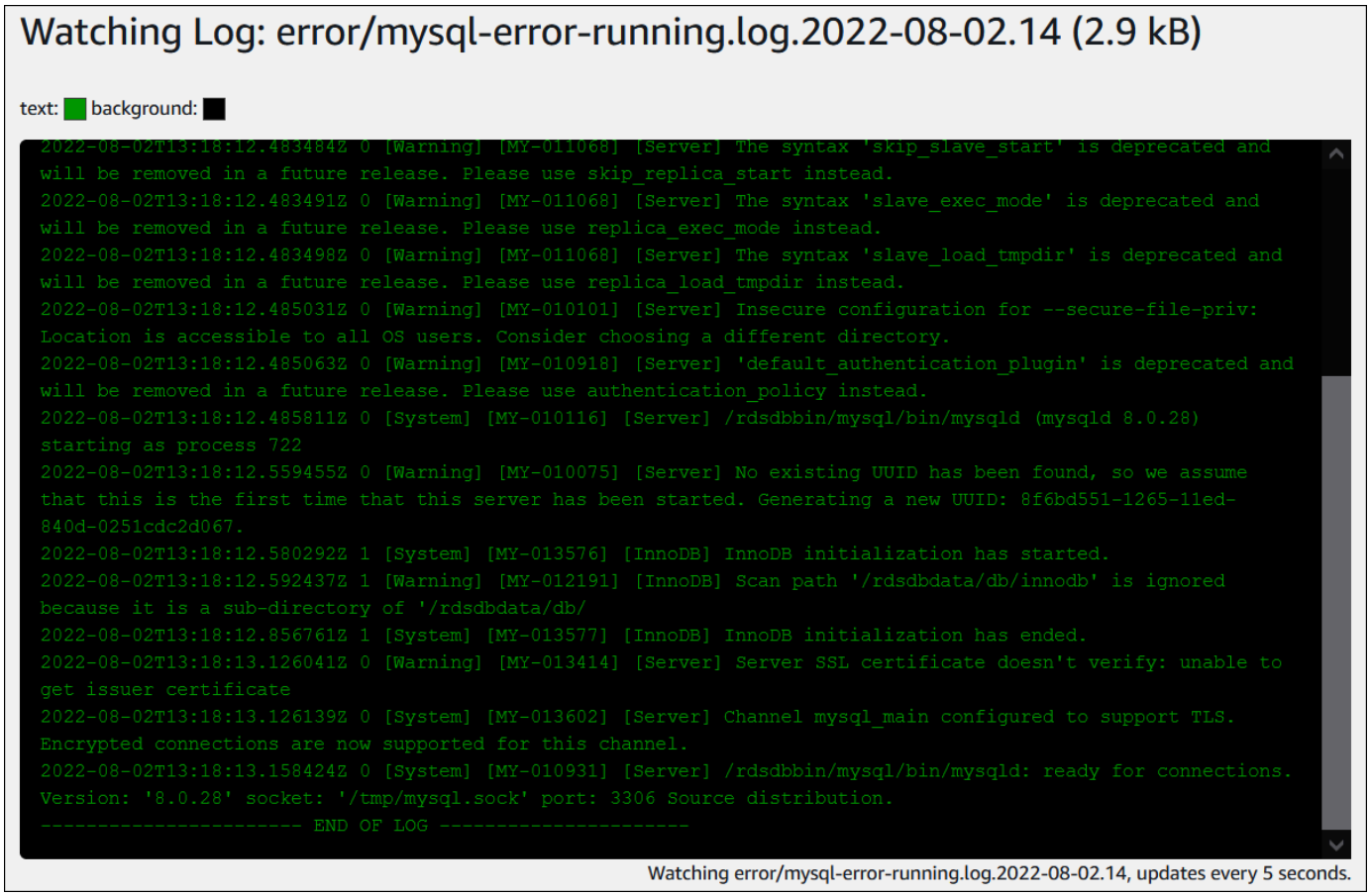
1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le nom de l'instance de base de données qui contient le fichier journal que vous voulez consulter.
4. Choisissez l'onglet Logs & events (Journaux et événements).



5. Dans la section Journaux, choisissez un fichier journal, puis Consulter.



RDS affiche la queue du journal, comme dans l'exemple MySQL suivant.



Publication des journaux de base de données dans Amazon CloudWatch Logs

Dans une base de données sur site, les journaux de la base de données résident sur le système de fichiers. Amazon RDS ne fournit pas d'accès hôte aux journaux de la base de données sur le système de fichiers de votre cluster de bases de données. Pour cette raison, Amazon RDS vous permet d'exporter les journaux de la base de données vers Amazon CloudWatch Logs. CloudWatch Logs vous permet d'effectuer une analyse en temps réel des données de journaux. Vous pouvez également stocker les données dans un stockage hautement durable et gérer les données grâce à l'agent CloudWatch Logs.

Rubriques

- Présentation de l'intégration de RDS avec CloudWatch Logs
- Décider des journaux à publier dans CloudWatch Logs

- Spécification des journaux à publier dans CloudWatch Logs
- Recherche et filtrage de vos journaux dans CloudWatch Logs

Présentation de l'intégration de RDS avec CloudWatch Logs

Dans CloudWatch Logs, un flux de journaux est une séquence d'événements de journaux qui partagent la même source. Chaque source distincte de journaux dans CloudWatch Logs constitue un flux de journaux distinct. Un groupe de journaux est un groupe de flux de journaux qui partagent les mêmes paramètres de conservation, de surveillance et de contrôle d'accès.

Amazon Aurora diffuse en continu les enregistrements des journaux de votre cluster de bases de données vers un groupe de journaux. Par exemple, vous possédez un groupe de journaux `/aws/rds/cluster/`*`cluster_name`*`/`*`log_type`* pour chaque type de journaux que vous publiez. Ce groupe de journaux se trouve dans la même région AWS que l'instance de base de données qui génère le journal.

AWS conserve les données de journaux publiées dans CloudWatch Logs pendant une période indéterminée, sauf si vous précisez une durée de conservation. Pour plus d'informations, consultez Modification de la conservation des données de journaux dans CloudWatch Logs.

Décider des journaux à publier dans CloudWatch Logs

Chaque moteur de base de données RDS prend en charge son propre ensemble de journaux. Pour en savoir plus sur les options de votre moteur de base de données, consultez les rubriques suivantes :

- the section called "Publication de journaux Aurora MySQL dans CloudWatch Logs"
- the section called "Publication de journaux Aurora PostgreSQL sur CloudWatch Logs"

Spécification des journaux à publier dans CloudWatch Logs

Vous spécifiez les journaux à publier dans la console. Assurez-vous que vous avez un rôle lié au service dans Gestion des identités et des accès AWS (IAM). Pour plus d'informations sur les rôles liés à un service, consultez Utilisation des rôles liés à un service pour Amazon Aurora.

Pour spécifier les journaux à publier

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

2. Dans la panneau de navigation, choisissez Bases de données.
3. Effectuez l'une des actions suivantes :
   - Choisissez Create database (Créer une base de données).
   - Choisissez une base de données dans la liste, puis sélectionnez Modify (Modifier).
4. Dans Logs exports (Exportations de journaux), choisissez les journaux à publier.

   L'exemple suivant spécifie le journal d'audit, les journaux d'erreurs, le journal général, le journal d'instance, le journal d'erreurs d'authentification de base de données IAM, et le journal des requêtes lentes pour un cluster de bases de données Aurora MySQL.

Recherche et filtrage de vos journaux dans CloudWatch Logs

Vous pouvez rechercher des entrées de journal qui correspondent à des critères spécifiés à partir de la console CloudWatch Logs. Vous pouvez accéder aux journaux soit par la console RDS, qui vous conduit à la console CloudWatch Logs, soit directement à partir de la console CloudWatch Logs.

Pour rechercher les journaux de votre RDS à l'aide de la console RDS

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Choisissez un cluster de bases de données ou une instance de base de données.
4. Choisissez Configuration.
5. Sous Published logs (Journaux publiés), choisissez le journal de la base de données que vous souhaitez afficher.

Pour effectuer une recherche dans vos journaux RDS à l'aide de la console CloudWatch Logs

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation, choisissez Groupes de journaux.
3. Dans la zone de filtre, entrez **`/aws/rds`**.
4. Pour Log Groups, choisissez le nom du groupe de journaux contenant le flux de journal devant faire l'objet de la recherche.
5. Pour Log Streams, choisissez le nom du flux de journal devant faire l'objet de la recherche.
6. Sous Journal des événements, saisissez la syntaxe du filtre à utiliser.

Pour plus d'informations, consultez Searching and filtering log data (Recherche et filtrage des données de journal) dans le Guide de l'utilisateur d'Amazon CloudWatch Logs Pour obtenir un tutoriel de blog expliquant comment surveiller les journaux RDS, consultez Création d'une surveillance proactive des bases de données pour Amazon RDS avec Amazon CloudWatch Logs, AWS Lambda et Amazon SNS.

Lecture du contenu des fichiers journaux avec REST

Amazon RDS fournit un point de terminaison REST qui permet d'accéder aux fichiers journaux des instances de base de données. Ceci est utile si vous devez écrire une application pour diffuser en continu le contenu de fichiers journaux Amazon RDS.

La syntaxe est la suivante :

```
GET /v13/downloadCompleteLogFile/DBInstanceIdentifier/LogFileName HTTP/1.1
Content-type: application/json
host: rds.region.amazonaws.com
```

Les paramètres suivants sont obligatoires :

- *DBInstanceIdentifier*—le nom assigné par le client de l'instance de base de données qui contient le fichier journal que vous souhaitez télécharger.
- *LogFileName*—le nom du fichier journal à télécharger.

La réponse contient les contenus du fichier journal demandé, en tant que flux.

L'exemple suivant télécharge le fichier journal appelé log/ERROR.6 pour l'instance de base de données appelée sample-sql dans la région us-west-2.

```
GET /v13/downloadCompleteLogFile/sample-sql/log/ERROR.6 HTTP/1.1
host: rds.us-west-2.amazonaws.com
X-Amz-Security-Token: AQoDYXdzEIH//////////
wEa0AIXLhngC5zp9CyB1R6abwKrXHVR5efnAVN3XvR7IwqKYalFSn6UyJuEFTft9nObglx4QJ+GXV9cpACkETq=
X-Amz-Date: 20140903T233749Z
X-Amz-Algorithm: AWS4-HMAC-SHA256
X-Amz-Credential: AKIADQKE4SARGYLE/20140903/us-west-2/rds/aws4_request
X-Amz-SignedHeaders: host
X-Amz-Content-SHA256: e3b0c44298fc1c229afbf4c8996fb92427ae41e4649b934de495991b7852b855
X-Amz-Expires: 86400
X-Amz-Signature: 353a4f14b3f250142d9afc34f9f9948154d46ce7d4ec091d0cdabbcf8b40c558
```

Si vous spécifiez une instance de base de données qui n’existe pas, la réponse se compose de l’erreur suivante :

- `DBInstanceNotFound`—*`DBInstanceIdentifier`* ne fait pas référence à une instance de base de données existante. (HTTP status code: 404)

Fichiers journaux de base de données Aurora MySQL

Vous pouvez surveiller les journaux Aurora MySQL directement via la console Amazon RDS, l'API Amazon RDS, l'AWS CLI ou des kits SDK AWS. Vous pouvez également accéder aux journaux MySQL en dirigeant les journaux vers une table de base de données de la base de données principale et interroger cette table. Vous pouvez utiliser l'utilitaire mysqlbinlog pour télécharger un journal binaire.

Pour plus d'informations sur l'affichage, le téléchargement ou la consultation des journaux de base de données basés sur des fichiers, consultez Surveillance des fichiers journaux Amazon Aurora.

Rubriques

Présentation des journaux de base de données Aurora MySQL

Vous pouvez surveiller les types de fichiers journaux Aurora MySQL suivants :

- Journal des erreurs
- Journal des requêtes lentes
- Journal général
- Journal d’audit
- Journal d’instance
- Journal des erreurs d’authentification de base de données IAM

Le journal des erreurs Aurora MySQL est généré par défaut. Vous pouvez générer le journal des requêtes lentes et le journal général en définissant les paramètres nécessaires dans votre groupe de paramètres de base de données.

Rubriques

- Journal d’audit Aurora MySQL
- Journal d’instance Aurora MySQL
- Renouvellement et rétention des journaux pour Aurora MySQL
- Publication de journaux Aurora MySQL dans Amazon CloudWatch Logs

Journaux des erreurs Aurora MySQL

Aurora MySQL écrit les erreurs dans le fichier `mysql-error.log`. Le nom du fichier journal comporte l’heure à laquelle le fichier a été généré (au format UTC). Les fichiers journaux comportent également un horodatage permettant de déterminer le moment où les entrées du journal ont été écrites.

Aurora MySQL écrit dans le journal des erreurs uniquement au moment du démarrage, de l’arrêt et lorsqu’une erreur survient. Une instance de base de données peut fonctionner pendant des heures ou des jours sans qu’aucune nouvelle entrée soit écrite dans le journal des erreurs. Si aucune entrée récente ne figure, cela signifie que le serveur n’a pas rencontré d’erreur justifiant une entrée de journal.

Par défaut, les journaux des erreurs sont filtrés de sorte que seuls les événements inattendus tels que les erreurs soient affichés. Toutefois, les journaux des erreurs contiennent également des informations supplémentaires sur la base de données, comme la progression des requêtes, qui ne sont pas affichées. Par conséquent, même en l’absence d’erreurs réelles, la taille des journaux des erreurs peut augmenter en raison des activités en cours sur la base de données. Et même si vous pouvez voir une certaine taille en octets ou en kilo-octets pour les journaux d’erreurs contenus dans la AWS Management Console, ils peuvent être vide lorsque vous les téléchargez.

Aurora MySQL écrit `mysql-error.log` sur le disque toutes les 5 minutes. Il ajoute le contenu du journal à `mysql-error-running.log`.

Aurora MySQL assure la rotation du fichier `mysql-error-running.log` toutes les heures.

> **Note**
>
> La période de conservation des journaux est différente pour Amazon RDS et Aurora.

Journal des requêtes lentes et journal général Aurora MySQL

Vous pouvez écrire le journal des requêtes lentes et le journal général Aurora MySQL dans un fichier ou dans une table de base de données. Pour ce faire, définissez les paramètres de votre groupe de paramètres de base de données. Pour plus d'informations sur la création et la modification d'un groupe de paramètres DB, consultez Groupes de paramètres pour Amazon Aurora. Vous devez définir ces paramètres avant de pouvoir consulter le journal des requêtes lentes ou le journal général dans la console Amazon RDS ou à l'aide de l'API Amazon RDS, de la CLI Amazon RDS ou de kits SDK AWS.

Vous pouvez contrôler la journalisation Aurora MySQL à l'aide des paramètres de cette liste :

- `slow_query_log` : Pour créer le journal des requêtes lentes, définir sur 1. La valeur par défaut est 0.
- `general_log` : Pour créer le journal général, définir sur 1. La valeur par défaut est 0.
- `long_query_time` : Pour empêcher l'enregistrement des requêtes rapides dans le journal des requêtes lentes, indiquez la valeur de la durée d'exécution de requête la plus courte devant être journalisée, en secondes. La valeur par défaut est de 10 secondes et la valeur minimum est 0. Si log_output = FILE, vous pouvez indiquer une valeur à virgule flottante avec une résolution en microseconde. Si log_output = TABLE, vous devez indiquer un nombre entier avec une résolution en seconde. Seules les requêtes dont la durée d'exécution dépasse la valeur `long_query_time` sont journalisées. Par exemple, si vous définissez `long_query_time` sur 0,1, les requêtes s'exécutant pendant moins de 100 millisecondes ne sont pas enregistrées.
- `log_queries_not_using_indexes` : Pour enregistrer toutes les requêtes n'utilisant pas d'index dans le journal des requêtes lentes, définir sur 1. Les requêtes n'utilisant pas d'index sont journalisées même si la durée de leur exécution est inférieure à la valeur du paramètre `long_query_time`. La valeur par défaut est 0.
- `log_output` *`option`* : Vous pouvez spécifier l'une des options suivantes pour le paramètre `log_output`.
  - TABLEAU – Écrit les requêtes générales dans le tableau `mysql.general_log` et les requêtes lentes dans le tableau `mysql.slow_log`.
  - FICHIER – Écrit les fichiers des requêtes générales et lentes dans le fichier système.
  - AUCUNE– Désactive la journalisation.

  Pour Aurora MySQL versions 2 et 3, la valeur par défaut pour `log_output` est `FILE`.

Pour que les données d'une requête lente apparaissent dans Amazon CloudWatch Logs, les conditions suivantes doivent être remplies :

- CloudWatch Logs doit être configuré pour inclure les journaux des requêtes lentes.
- `slow_query_log` doit être activé.
- `log_output` doit être défini sur `FILE`.
- La durée de la requête doit être plus longue que celle configurée pour `long_query_time`.

Pour plus d'informations sur le journal des requêtes lentes et le journal général, accédez aux rubriques suivantes dans la documentation MySQL :

- Journal des requêtes lentes
- Journal des requêtes générales

Journal d'audit Aurora MySQL

La journalisation d'audit pour Aurora MySQL se nomme « audit avancé ». Pour activer l'audit avancé, définissez certains paramètres de cluster de bases de données. Pour plus d'informations, consultez Utilisation de l'Audit avancé avec un cluster de bases de données Amazon Aurora MySQL.

Journal d'instance Aurora MySQL

Aurora crée un fichier journal distinct pour les instances de base de données pour lesquelles la pause automatique est activée. le fichier instance.log enregistre toutes les raisons pour lesquelles ces instances de base de données n'ont pas pu être mises en pause comme prévu. Pour plus d'informations sur le comportement des fichiers journaux d'instance et la fonctionnalité de pause automatique d'Aurora, consultez Surveillance des activités de Aurora sans serveur v2 pause et de reprise.

Renouvellement et rétention des journaux pour Aurora MySQL

Lorsque la journalisation est activée, Amazon Aurora procède à la rotation ou à la suppression des fichiers journaux à intervalles réguliers. Cette précaution permet de limiter la possibilité qu'un fichier journal volumineux ne bloque l'utilisation de la base de données ou n'affecte les performances. Aurora MySQL gère la rotation et la suppression des journaux comme suit :

- La taille des fichiers journaux des erreurs Aurora MySQL est limitée à 15 % maximum de l'espace de stockage local pour une instance de base de données. Pour maintenir ce seuil, les journaux

sont automatiquement renouvelés toutes les heures. Aurora MySQL supprime les journaux au bout de 30 jours ou lorsque 15 % de l'espace disque est atteint. Si la taille de l'ensemble des fichiers journaux après la suppression dépasse le seuil, les fichiers journaux les plus anciens sont supprimés jusqu'à ce que la taille des fichiers journaux ne soit plus supérieure au seuil.

- Aurora MySQL supprime les journaux d'audit, généraux et de requêtes lentes au bout de 24 heures ou lorsque 15 % du stockage est utilisé.
- Lorsque la journalisation `FILE` est activée, les fichiers journaux généraux et les fichiers journaux des requêtes lentes sont examinés toutes les heures et ceux datant de plus de 24 heures sont supprimés. Dans certains cas, la taille des fichiers journaux combinés restant après la suppression peut dépasser le seuil de 15 % de l'espace local d'une instance de base de données. Le cas échéant, les fichiers journaux les plus anciens sont supprimés jusqu'à ce que la taille des fichiers journaux ne soit plus supérieure au seuil.
- Lorsque la journalisation `TABLE` est activée, les tables des journaux ne font l'objet d'aucune rotation ou suppression. Les tables des journaux sont tronquées lorsque la taille de tous les journaux combinés est trop grande. Vous pouvez vous abonner à la catégorie d'événement `low storage` pour être informé lorsque les tables de journaux doivent faire l'objet d'une rotation ou d'une suppression manuelle pour libérer de l'espace. Pour plus d'informations, consultez Utiliser la notification d'événements d'Amazon RDS.

  Vous pouvez effectuer une rotation manuelle de la table `mysql.general_log` en appelant la procédure `mysql.rds_rotate_general_log`. Vous pouvez effectuer une rotation de la table `mysql.slow_log` en appelant la procédure `mysql.rds_rotate_slow_log`.

  Lors de la rotation manuelle des tables de journaux, la table de journal actuelle est copiée vers une table de journal de sauvegarde et les entrées de la table de journal actuelle sont supprimées. Si la table de journal de sauvegarde existe déjà, elle est supprimée avant que la table de journal actuelle ne soit copiée dans la sauvegarde. Si besoin, vous pouvez interroger la table de journal de sauvegarde. La table de journal de sauvegarde de la table `mysql.general_log` est nommée `mysql.general_log_backup`. La table de journal de sauvegarde de la table `mysql.slow_log` est nommée `mysql.slow_log_backup`.
- Les journaux d'audit Aurora MySQL font l'objet d'une rotation lorsque la taille des fichiers atteint 100 Mo. Ils sont supprimés au bout de 24 heures.
- Amazon RDS fait pivoter les fichiers journaux d'erreurs d'authentification de base de données IAM supérieurs à 10 Mo. Amazon RDS supprime les fichiers journaux d'erreurs d'authentification de base de données IAM datant de plus de cinq jours ou de plus de 100 Mo.

Pour utiliser les journaux depuis la console Amazon RDS, l'API Amazon RDS, la CLI Amazon RDS ou les kits SDK AWS, définissez le paramètre `log_output` sur FILE. A l'instar du journal des erreurs Aurora MySQL, ces fichiers journaux font l'objet d'une rotation horaire. Les fichiers journaux qui ont été générés au cours des dernières 24 heures sont conservés. Veuillez noter que la période de rétention est différente pour Amazon RDS et pour Aurora.

Publication de journaux Aurora MySQL dans Amazon CloudWatch Logs

Vous pouvez configurer votre cluster de bases de données Aurora MySQL de sorte à publier des données de journaux dans un groupe de journaux dans Amazon CloudWatch Logs. CloudWatch Logs vous permet d'effectuer une analyse en temps réel des données de journaux et d'utiliser CloudWatch pour créer des alarmes et afficher des métriques. CloudWatch Logs permet de conserver les enregistrements des journaux dans une solution de stockage hautement durable. Pour plus d'informations, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs.

Envoi de la sortie du journal Aurora MySQL à des tables

Vous pouvez diriger le journal des requêtes lentes et le journal général vers des tables sur l'instance de base de données en créant un groupe de paramètres DB et en définissant le paramètre du serveur `log_output` sur `TABLE`. Les requêtes générales sont ensuite enregistrées dans la table `mysql.general_log` et les requêtes lentes dans la table `mysql.slow_log`. Vous pouvez interroger les tables pour accéder aux informations des journaux. L'activation de cette journalisation augmente le volume de données écrites dans la base de données, ce qui peut dégrader les performances.

Par défaut, le journal général et le journal des requêtes lentes sont désactivés. Afin d'activer la journalisation dans les tables, vous devez également définir les paramètres `general_log` et `slow_query_log` sur `1`.

Les tables de journaux continuent de grossir jusqu'à ce que les activités de journalisation correspondantes soient désactivées en redéfinissant le paramètre approprié sur `0`. Avec le temps, une grande quantité de données s'accumule et risque d'utiliser une part considérable de l'espace de stockage alloué. Amazon Aurora ne vous permet pas de tronquer les tables de journaux, mais vous pouvez déplacer leurs contenus. Lorsque vous procédez à la rotation d'une table, son contenu est enregistré dans une table de sauvegarde et une nouvelle table de journal vide est créée. Vous pouvez effectuer une rotation manuelle des tables de journaux avec les procédures de ligne de commande suivantes, dans lesquelles l'invite de commande est indiquée par `PROMPT>` :

```
PROMPT> CALL mysql.rds_rotate_slow_log;
PROMPT> CALL mysql.rds_rotate_general_log;
```

Pour supprimer totalement les anciennes données et récupérer l'espace de disque, appelez deux fois à la suite la procédure appropriée.

Configuration d'Aurora MySQL pour la journalisation des bases de données mono-AZ

Le journal binaire est un jeu de fichiers journaux contenant des informations sur les modifications de données apportées à une instance de serveur Aurora MySQL. Le journal binaire contient des informations telles que les suivantes :

- Événements décrivant les modifications apportées à la base de données telles que la création de tables ou les modifications de lignes
- Informations sur la durée de chaque instruction qui a mis à jour les données
- Événements pour des instructions pouvant mettre à jour des données mais ne l'ayant pas fait

Le journal binaire enregistre les instructions envoyées pendant la réplication. Il est également requis pour certaines opérations de récupération. Pour plus d'informations, consultez The Binary Log dans la documentation MySQL.

Les journaux binaires sont accessibles uniquement à partir de l'instance de base de données principale, et non à partir des réplicas.

MySQL on Amazon Aurora prend en charge les formats de journalisation binaire basés sur les lignes, basés sur les instructions et mixtes. Nous recommandons le format mixte, sauf si vous avez besoin d'un format binlog spécifique. Pour plus de détails sur les différents formats de journalisation binaire Aurora MySQL, consultez Formats de journalisation binaire dans la documentation MySQL.

Si vous prévoyez d'utiliser la réplication, le format de journalisation binaire est important, car il détermine le dossier de modifications de données qui est enregistré dans la source et envoyés aux cibles de réplication. Pour plus d'informations sur les avantages et les désavantages des différents formats de journalisation binaire pour la réplication, consultez Advantages and Disadvantages of Statement-Based and Row-Based Replication dans la documentation MySQL.

> **Important**
>
> Avec MySQL 8.0.34, MySQL a rendu le paramètre `binlog_format` obsolète. Dans les versions ultérieures de MySQL, MySQL prévoit de supprimer le paramètre et de ne prendre

en charge que la réplication basée sur les lignes. Par conséquent, nous recommandons d'utiliser la journalisation basée sur les lignes pour les nouvelles configurations de réplication MySQL. Pour plus d'informations, consultez binlog_format dans la documentation MySQL.
Les versions 8.0 et 8.4 de MySQL acceptent le paramètre `binlog_format`. Lors de l'utilisation de ce paramètre, MySQL émet un avertissement d'obsolescence. Dans une future version majeure, MySQL supprimera le paramètre `binlog_format`.
La réplication basée sur les instructions peut provoquer des incohérences entre le cluster de bases de données source et un réplica en lecture. Pour plus d'informations, consultez Determination of Safe and Unsafe Statements in Binary Logging dans la documentation MySQL.
L'activation de la journalisation binaire augmente le nombre d'opérations d'I/O d'écriture disque sur le cluster de bases de données. Vous pouvez surveiller l'utilisation des IOPS à l'aide de la métrique CloudWatch `VolumeWriteIOPs`.

Pour définir le format de journalisation binaire MySQL

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Choisissez le groupe de paramètres de cluster de bases de données, associé au cluster de bases de données, que vous voulez modifier.

   Vous ne pouvez pas modifier un groupe de paramètres par défaut. Si le cluster de bases de données utilise un groupe de paramètres par défaut, créez un nouveau groupe et associez-le à au cluster.

   Pour plus d'informations sur les groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora.
4. Pour Actions, choisissez Modifier.
5. Définissez le paramètre `binlog_format` au format de journalisation binaire de votre choix (`ROW`, `STATEMENT` ou `MIXED`). Vous pouvez également utiliser la valeur `OFF` pour désactiver la journalisation binaire.

   > **Note**
   >
   > Le réglage de `binlog_format` sur `OFF` dans le groupe de paramètres du cluster de bases de données désactive la variable de session `log_bin`. Cela désactive la

journalisation binaire sur le cluster de bases de données Aurora MySQL, lequel à son tour réinitialise la variable de session `binlog_format` à la valeur par défaut `ROW` dans la base de données.

6. Choisissez Save changes (Enregistrer les modifications)pour enregistrer les mises à jour apportées au groupe de paramètres de cluster de bases de données.

Après avoir effectué ces étapes, vous devez redémarrer l'instance d'enregistreur dans le cluster de bases de données pour que vos modifications s'appliquent. Dans Aurora MySQL version 2.09 et inférieures, lorsque vous redémarrez l'instance d'enregistreur, toutes les instances de lecteur du cluster de bases de données sont également redémarrées. Dans Aurora MySQL version 2.10 et ultérieures, vous devez redémarrer toutes les instances de lecteur manuellement. Pour plus d'informations, consultez Redémarrage d'un cluster de bases de données Amazon Aurora ou d'une instance de base de données Amazon Aurora.

Important

La modification d'un groupe de paramètres de cluster de bases de données affecte tous les clusters de bases de données qui utilisent ce dernier. Si vous souhaitez spécifier des formats de journalisation binaire différents pour différents clusters de bases de données Aurora MySQL dans une région AWS, les clusters de bases de données doivent utiliser différents groupes de paramètres de cluster de bases de données. Ces groupes de paramètres identifient différents formats de journalisation. Affectez le groupe de paramètres de cluster de bases de données approprié à chaque cluster de bases de données. Pour plus d'informations sur les paramètres Aurora MySQL, consultez Paramètres de configuration d'Aurora MySQL.

Accès aux journaux binaires MySQL

Vous pouvez utiliser l'utilitaire mysqlbinlog pour télécharger ou diffuser des journaux binaires à partir des instances de base de données RDS for MySQL. Le journal binaire est téléchargé dans votre ordinateur local et vous pouvez effectuer des actions comme relire le journal à l'aide de l'utilitaire mysql. Pour plus d'informations sur l'utilisation de l'utilitaire mysqlbinlog, consultez Using mysqlbinlog to back up binary log files (Utilisation de mysqlbinlog pour sauvegarder les fichiers journaux binaires) dans la documentation MySQL.

Pour exécuter à nouveau l'utilitaire mysqlbinlog sur une instance Amazon RDS, utilisez les options suivantes :

- `--read-from-remote-server` : obligatoire.
- `--host` : le nom DNS du point de terminaison de l'instance.
- `--port` : le port utilisé par l'instance.
- `--user` : un utilisateur MySQL ayant l'autorisation `REPLICATION SLAVE`.
- `--password` : le mot de passe de l'utilisateur MySQL ou omettez la valeur de mot de passe pour que l'utilitaire vous invite à saisir un mot de passe.
- `--raw` : téléchargez le fichier au format binaire.
- `--result-file` : le fichier local qui recevra la sortie brute.
- `--stop-never` : diffusez les fichiers journaux binaires.
- `--verbose` : lorsque vous utilisez le format binlog `ROW`, incluez cette option pour afficher les événements de ligne sous forme d'instructions pseudo-SQL. Pour plus d'informations sur l'option `--verbose`, consultez mysqlbinlog row event display (Affichage d'événements de ligne mysqlbinlog) dans la documentation MySQL.
- Spécifiez les noms pour un ou plusieurs fichiers journaux binaires. Pour obtenir la liste des journaux disponibles, utilisez la commande SQL `SHOW BINARY LOGS`.

Pour plus d'informations sur les options mysqlbinlog, consultez mysqlbinlog — Utility for processing binary log files (mysqlbinlog : utilitaire de traitement des fichiers journaux binaires) dans la documentation MySQL.

Les exemples suivants montrent comment utiliser l'utilitaire mysqlbinlog.

Pour Linux, macOS ou Unix :

```
mysqlbinlog \
    --read-from-remote-server \
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com \
    --port=3306  \
    --user ReplUser \
    --password \
    --raw \
    --verbose \
    --result-file=/tmp/ \
    binlog.00098
```

Pour Windows :

```
mysqlbinlog ^
    --read-from-remote-server ^
    --host=MySQLInstance1.cg034hpkmmjt.region.rds.amazonaws.com ^
    --port=3306  ^
    --user ReplUser ^
    --password ^
    --raw ^
    --verbose ^
    --result-file=/tmp/ ^
    binlog.00098
```

Les journaux binaires doivent rester disponibles sur l'instance de base de données pour que l'utilitaire mysqlbinlog puisse y accéder. Pour garantir leur disponibilité, utilisez la procédure mysql.rds_set_configuration stockée et spécifiez une période suffisamment longue pour télécharger les journaux. Si cette configuration n'est pas définie, Amazon RDS purge les journaux binaires dès que possible, ce qui entraîne des lacunes dans les journaux binaires récupérés par l'utilitaire mysqlbinlog.

L'exemple suivant définit la période de conservation sur 1 jour.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

Pour afficher les paramètres actuels, utilisez la procédure stockée mysql.rds_show_configuration.

```
call mysql.rds_show_configuration;
```

Fichiers journaux de base de données Aurora PostgreSQL

Vous pouvez surveiller les types de fichiers journaux Aurora PostgreSQL suivants :

- Journal PostgreSQL
- Journal d'instance
- Journal des erreurs d'authentification de base de données IAM

> **Note**
>
> Pour activer les journaux d'erreurs d'authentification de base de données IAM, vous devez d'abord activer l'authentification de base de données IAM pour votre cluster de bases de données Aurora PostgreSQL. Pour plus d'informations sur l'authentification de la base de données IAM, consultez Activation et désactivation de l'authentification de base de données IAM.

Aurora PostgreSQL consigne les activités de base de données dans le fichier journal PostgreSQL par défaut. Pour une instance de base de données PostgreSQL sur site, ces messages sont stockés localement dans `log/postgresql.log`. Pour un cluster de bases de données Aurora PostgreSQL, le fichier journal est disponible sur le cluster Aurora. Ces journaux sont également accessibles via la AWS Management Console, où vous pouvez les consulter ou les télécharger. Le niveau de journalisation par défaut capture les échecs de connexion, les erreurs de serveur fatales, les blocages et les échecs de requête.

Pour plus d'informations sur l'affichage, le téléchargement et la consultation des journaux de base de données basés sur des fichiers, consultez Surveillance des fichiers journaux Amazon Aurora. Pour en savoir plus sur les journaux PostgreSQL, consultez la section Working with Amazon RDS and Aurora PostgreSQL logs: Part 1 (Utilisation des journaux Amazon RDS et Aurora PostgreSQL : Partie 1) et Working with Amazon RDS and Aurora PostgreSQL logs: Part 2 (Utilisation des journaux Amazon RDS et Aurora PostgreSQL : Partie 2).

Outre les journaux PostgreSQL standard décrits dans cette rubrique, Aurora PostgreSQL prend également en charge l'extension PostgreSQL Audit (`pgAudit`). La plupart des secteurs réglementés et des agences gouvernementales doivent conserver un journal d'audit ou une piste d'audit des modifications apportées aux données afin de se conformer aux exigences légales. Pour plus d'informations sur l'installation et l'utilisation de pgAudit, consultez Utilisation de pgAudit pour journaliser l'activité de la base de données.

Aurora crée un fichier journal distinct pour les instances de base de données pour lesquelles la pause automatique est activée. le fichier instance.log enregistre toutes les raisons pour lesquelles ces instances de base de données n'ont pas pu être mises en pause comme prévu. Pour plus d'informations sur le comportement des fichiers journaux d'instance et la fonctionnalité de pause automatique d'Aurora, consultez Surveillance des activités de Aurora sans serveur v2 pause et de reprise.

Rubriques

- Paramètres de journalisation dans Aurora PostgreSQL
- Activer la journalisation des requêtes pour votre cluster de bases de données Aurora PostgreSQL.

Paramètres de journalisation dans Aurora PostgreSQL

Vous pouvez personnaliser le comportement de journalisation de votre cluster de bases de données Aurora PostgreSQL en modifiant divers paramètres. Dans le tableau suivant, vous trouverez les paramètres qui affectent combien de temps les journaux sont stockés, quand effectuer la rotation du journal et s'il convient de fournir en sortie le journal au format CSV (valeurs séparées par des virgules). Vous pouvez également trouver le texte de sortie envoyé à STDERR, entre autres paramètres. Pour modifier les valeurs des paramètres modifiables, utilisez un groupe de paramètres de cluster de bases de données pour votre cluster de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez Groupes de paramètres pour Amazon Aurora.

Paramètre	Par défaut	Description
log_destination	stderr	Définit le format de sortie pour le journal. La valeur par défaut est `stderr`, mais vous pouvez également spécifier une valeur séparée par des virgules (CSV) en ajoutant `csvlog` au paramètre. Pour plus d'informations, consultez Définition de la destination du journal (`stderr`, `csvlog`).
log_filename	postgresql.log.%Y-%m-%d-%H%M	Spécifie le modèle du nom de fichier journal. Outre la valeur par défaut, ce paramètre prend en charge `postgresql.log.%Y-%m-%d` et `postgresql.log.%Y-%m-%d-%H` pour le modèle de nom de fichier. Pour Aurora

Paramètre	Par défaut	Description
		PostgreSQL version 17.4 et versions ultérieur es, vous ne pouvez pas modifier ce paramètre.
log_line_prefix	%t:%r:%u@%d:[%p]:	Définit le préfixe pour chaque ligne de journal qui est écrite sur `stderr`, afin de noter l'heure (%t), l'hôte distant (%r), l'utilisateur (%u), la base de données (%d) et l'ID du processus (%p).
log_rotation_age	60	Minutes après lesquelles la rotation automatiq ue du fichier journal est effectuée. Vous pouvez remplacer cette valeur par toute valeur comprise entre 1 et 1 440 minutes. Pour plus d'informations, consultez Définition de la rotation des fichiers journaux.
log_rotation_size	–	Taille (en Ko) à laquelle la rotation automatiq ue du fichier journal est effectuée. Vous pouvez modifier cette valeur dans une plage comprise entre 50 000 et 1 000 000 kilo-octets. Pour en savoir plus, consultez Définition de la rotation des fichiers journaux.
rds.log_retention_ period	4 320	Les journaux PostgreSQL plus anciens que le nombre de minutes spécifié sont supprimés. La valeur par défaut de 4 320 minutes supprime les fichiers journaux après trois jours. Pour plus d'informations, consultez Définition de la période de conservation des journaux.

Pour identifier les problèmes d'application, vous pouvez rechercher dans le journal les échecs de requête, les échecs de connexion, les interblocages et les erreurs fatales du serveur. Par exemple, supposons que vous avez converti une application héritée d'Oracle vers Aurora, mais que certaines requêtes n'ont pas été converties correctement. Ces requêtes mal formatées génèrent des messages d'erreur que vous pouvez trouver dans les journaux pour aider à identifier les problèmes. Pour plus

d'informations sur la journalisation des requêtes, consultez Activer la journalisation des requêtes pour votre cluster de bases de données Aurora PostgreSQL..

Dans les rubriques suivantes, vous pouvez trouver des informations sur la manière de définir les différents paramètres qui contrôlent les détails de base de vos journaux PostgreSQL.

Rubriques

- Définition de la période de conservation des journaux
- Définition de la rotation des fichiers journaux
- Définition de la destination du journal (stderr, csvlog)
- Compréhension du paramètre log_line_prefix

Définition de la période de conservation des journaux

Le paramètre `rds.log_retention_period` indique la durée pendant laquelle votre cluster de bases de données Aurora PostgreSQL conserve ses fichiers journaux. La valeur par défaut est de 3 jours (4 320 minutes), mais vous pouvez définir une valeur comprise entre 1 jour (1 440 minutes) et 7 jours (10 080 minutes). Assurez-vous que votre instance de base de données Aurora PostgreSQL dispose d'un espace de stockage suffisant pour contenir les fichiers journaux pendant cette période.

Nous vous recommandons de publier régulièrement vos journaux dans Amazon CloudWatch Logs, afin de pouvoir visualiser et analyser les données système longtemps après que les journaux ont été supprimés de votre cluster de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez Publication de journaux Aurora PostgreSQL sur Amazon CloudWatch Logs. Une fois que vous avez configuré la publication CloudWatch, Aurora ne supprime pas un journal tant qu'il n'a pas été publié dans CloudWatch Logs.

Quand la capacité de stockage de l'instance de base de données atteint un seuil, Amazon Aurora compresse les journaux PostgreSQL plus anciens. Aurora compresse les fichiers en utilisant l'utilitaire de compression gzip. Pour plus d'informations, consultez le site Web de gzip.

Lorsque le stockage de l'instance de base de données est faible et que tous les journaux disponibles sont compressés, vous obtenez un avertissement qui ressemble à l'exemple suivant :

```
Warning: local storage for PostgreSQL log files is critically low for
this Aurora PostgreSQL instance, and could lead to a database outage.
```

S'il n'y a pas assez de stockage, Aurora peut supprimer les journaux PostgreSQL compressés avant la fin de la période de conservation spécifiée. Si c'est le cas, vous verrez apparaître un message similaire au suivant :

```
The oldest PostgreSQL log files were deleted due to local storage constraints.
```

Définition de la rotation des fichiers journaux

Aurora crée de nouveaux fichiers journaux toutes les heures, par défaut. Le timing est contrôlé par le paramètre `log_rotation_age`. Ce paramètre a une valeur par défaut de 60 (minutes), mais vous pouvez le régler sur une valeur comprise entre 1 minute et 24 heures (1 440 minutes). Au moment de la rotation, un fichier journal distinct est créé. Le fichier est nommé selon le modèle spécifié par le paramètre `log_filename`.

Les fichiers journaux peuvent également faire l'objet d'une rotation en fonction de leur taille, comme indiqué dans le paramètre `log_rotation_size`. Ce paramètre indique que le journal doit faire l'objet d'une rotation lorsqu'il atteint la taille spécifiée (en kilo-octets). La valeur `log_rotation_size` par défaut est de 100 000 Ko (kilo-octets) pour un cluster de bases de données Aurora PostgreSQL, mais vous pouvez la définir entre 50 000 et 1 000 000 de kilo-octets.

Les noms de fichier journal sont basés sur le modèle de nom de fichier spécifié dans le paramètre `log_filename`. Les valeurs disponibles pour ce paramètre sont les suivantes :

- `postgresql.log.%Y-%m-%d` : format par défaut du nom de fichier journal. Inclut l'année, le mois et la date dans le nom du fichier journal.
- `postgresql.log.%Y-%m-%d-%H` – Inclut l'heure dans le format du nom de fichier journal.
- `postgresql.log.%Y-%m-%d-%H%M` – Inclut l'heure:minute dans le format du nom de fichier journal.

Si vous définissez le paramètre `log_rotation_age` sur une valeur inférieure à 60 minutes, définissez également le paramètre `log_filename` au format minute.

Pour plus d'informations, consultez `log_rotation_age` et `log_rotation_size` dans la documentation de PostgreSQL.

Définition de la destination du journal (**`stderr`**, **`csvlog`**)

Par défaut, Aurora PostgreSQL génère des journaux au format d'erreur standard (stderr). Ce format correspond au réglage par défaut du paramètre `log_destination`. Chaque message est préfixé

selon le modèle spécifié dans le paramètre `log_line_prefix`. Pour plus d'informations, consultez Compréhension du paramètre log_line_prefix.

Aurora PostgreSQL peut également générer les journaux au format `csvlog`. La valeur `csvlog` permet d'analyser les données du journal en tant que données CSV (valeurs séparées par des virgules). Par exemple, supposons que vous utilisez l'extension `log_fdw` pour travailler avec vos journaux en tant que tables externes. La table externe créée sur les fichiers journaux `stderr` contient une seule colonne avec les données des événements de journal. En ajoutant `csvlog` au paramètre `log_destination`, vous obtenez le fichier journal au format CSV avec des démarcations pour les différentes colonnes de la table externe. Vous pouvez désormais trier et analyser vos journaux plus facilement.

Si vous spécifiez `csvlog` pour ce paramètre, sachez que les deux fichiers `stderr` et `csvlog` sont générés. Assurez-vous de surveiller le stockage consommé par les journaux, en tenant compte de `rds.log_retention_period` et des autres paramètres qui affectent le stockage et la rotation des journaux. Utiliser `stderr` et `csvlog` fait plus que doubler le stockage consommé par les journaux.

Si vous ajoutez `csvlog` à `log_destination` et que vous souhaitez revenir au paramètre `stderr` seul, vous devez réinitialiser le paramètre. Pour ce faire, ouvrez la console Amazon RDS, puis ouvrez le groupe de paramètres personnalisé du cluster de bases de données pour votre instance. Choisissez le paramètre `log_destination`, choisissez Edit parameter (Modifier le paramètre), puis Reset (Réinitialiser).

Pour plus d'informations sur la configuration de la journalisation, consultez Working with Amazon RDS and Aurora PostgreSQL logs: Part 1 (Utiliser les journaux d'Amazon RDS et Aurora PostgreSQL : partie 1).

Compréhension du paramètre log_line_prefix

Le format du journal `stderr` précède chaque message du journal des détails spécifiés par le paramètre `log_line_prefix`. La valeur par défaut est :

```
%t:%r:%u@%d:[%p]:t
```

À partir de la version 16 d'Aurora PostgreSQL, vous pouvez également choisir :

```
%m:%r:%u@%d:[%p]:%l:%e:%s:%v:%x:%c:%q%a
```

Chaque entrée de journal envoyée à stderr inclut les informations suivantes en fonction de la valeur sélectionnée :

- `%t` : heure de l'entrée du journal sans millisecondes
- `%m` : heure de l'entrée du journal, en millisecondes
- `%r` : adresse de l'hôte distant
- `%u@%d` : nom d'utilisateur @ nom de base de données
- `[%p]` : ID de processus si disponible
- `%l` : numéro de ligne de journal par session
- `%e` : code d'erreur SQL
- `%s` : horodatage de début du processus
- `%v` : identifiant de transaction virtuel
- `%x` : ID de transaction
- `%c` : ID de session
- `%q` : délimiteur non session
- `%a` : nom de l'application

Activer la journalisation des requêtes pour votre cluster de bases de données Aurora PostgreSQL.

Vous pouvez collecter des informations plus détaillées sur les activités de votre base de données, notamment les requêtes, les requêtes en attente de verrouillage, les points de contrôle et de nombreux autres détails en définissant certains des paramètres répertoriés dans le tableau suivant. Cette rubrique se concentre sur la journalisation des requêtes.

Paramètre	Par défaut	Description
log_connections	–	Enregistre toutes les connexions réussies. Pour savoir comment utiliser ce paramètre avec `log_disconnections` pour détecter les pertes de connexion, consultez Gestion de l'abandon des connexions d'Aurora PostgreSQL avec regroupement des connexions .
log_disconnections	–	Journalise la fin de chaque session et sa durée. Pour savoir comment utiliser ce paramètre avec `log_connections` pour détecter les pertes

Paramètre	Par défaut	Description
		de connexion, consultez Gestion de l'abandon des connexions d'Aurora PostgreSQL avec regroupement des connexions .
log_checkpoints	–	Non applicable à Aurora PostgreSQL
log_lock_waits	–	Enregistre les longs temps d'attente pour l'acquisition d'un verrou. Ce paramètre n'est pas défini par défaut.
log_min_duration_sample	–	(ms) Définit la durée minimum d'exécution au-delà de laquelle les instructions sont journalisées. La taille de l'échantillon est définie à l'aide du paramètre `log_statement_sample_rate` .
log_min_duration_statement	–	Toute instruction SQL exécutée au moins pendant la durée spécifiée ou plus est journalisée. Ce paramètre n'est pas défini par défaut. L'activation de ce paramètre peut vous aider à identifier les requêtes non optimisées.
log_statement	–	Définit le type d'instructions enregistrées. Par défaut, ce paramètre n'est pas défini, mais vous pouvez le modifier pour `all`, `ddl` ou `mod` afin de spécifier les types d'instructions SQL que vous souhaitez journaliser. Si vous spécifiez autre chose que `none` pour ce paramètre, vous devez également prendre des mesures supplémentaires pour empêcher l'exposition des mots de passe dans les fichiers journaux. Pour plus d'informations, consultez Atténuation du risque d'exposition des mots de passe lors de l'utilisation de la journalisation de requêtes.

Paramètre	Par défaut	Description
log_statement_sample_rate	–	Le pourcentage d'instructions dépassant la durée spécifiée dans `log_min_duration_sample` pour être journalisées, exprimé sous la forme d'une valeur à virgule flottante comprise entre 0,0 et 1,0.
log_statement_stats	–	Ecrit les statistiques de performance cumulées dans le journal du serveur.

Utilisation de la journalisation pour détecter les requêtes lentes

Vous pouvez journaliser les instructions et les requêtes SQL pour favoriser la recherche des requêtes lentes. Vous pouvez activer cette fonctionnalité en modifiant les valeurs des paramètres `log_statement` et `log_min_duration`, comme indiqué dans cette section. Avant d'activer la journalisation des requêtes pour votre cluster de bases de données Aurora PostgreSQL, vous devez être conscient de l'exposition possible à des mots de passe dans les journaux et de la manière d'atténuer les risques. Pour plus d'informations, consultez Atténuation du risque d'exposition des mots de passe lors de l'utilisation de la journalisation de requêtes.

Vous trouverez ci-dessous des informations de référence sur les paramètres `log_statement` et `log_min_duration`.

log_statement

Ce paramètre indique le type d'instructions SQL qui doivent être envoyées au journal. La valeur par défaut est `none`. Si vous remplacez ce paramètre par `all`, `ddl` ou `mod`, veillez à prendre les mesures recommandées pour réduire le risque d'exposition des mots de passe dans les journaux. Pour plus d'informations, consultez Atténuation du risque d'exposition des mots de passe lors de l'utilisation de la journalisation de requêtes.

Tout

Journalise toutes les instructions. Ce paramètre est recommandé à des fins de débogage.

ddl

Journalise toutes les instructions DDL (Data Definition Language), telles que CREATE, ALTER, DROP, etc.

mod

Journalise toutes les instructions DDL et les instructions de langage de manipulation des données (DML) telles que INSERT, UPDATE et DELETE, qui modifient les données.

none

Aucune instruction SQL n'est journalisée. Nous recommandons ce paramètre pour éviter le risque d'exposer des mots de passe dans les journaux.

log_min_duration_statement

Toute instruction SQL exécutée au moins pendant la durée spécifiée ou plus est journalisée. Ce paramètre n'est pas défini par défaut. L'activation de ce paramètre peut vous aider à identifier les requêtes non optimisées.

–1–2147483647

Le nombre de millisecondes (ms) d'exécution pendant lequel une instruction est journalisée.

Configurer la journalisation des requêtes

Ces étapes supposent que votre cluster de bases de données Aurora PostgreSQL utilise un groupe de paramètres de cluster de bases de données personnalisé.

1. Définissez le paramètre `log_statement` sur `all`. L'exemple suivant illustre les informations écrites dans le fichier `postgresql.log` avec cette définition de paramètre.

```
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: statement:
 SELECT feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:LOG: QUERY
 STATISTICS
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:DETAIL: ! system
 usage stats:
! 0.017355 s user, 0.000000 s system, 0.168593 s elapsed
! [0.025146 s user, 0.000000 s system total]
! 36644 kB max resident size
! 0/8 [0/8] filesystem blocks in/out
! 0/733 [0/1364] page faults/reclaims, 0 [0] swaps
```

```
! 0 [0] signals rcvd, 0/0 [0/0] messages rcvd/sent
! 19/0 [27/0] voluntary/involuntary context switches
2022-10-05 22:05:52 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: SELECT
 feedback, s.sentiment,s.confidence
FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
ORDER BY s.confidence DESC;
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:ERROR: syntax error
 at or near "ORDER" at character 1
2022-10-05 22:05:56 UTC:52.95.4.1(11335):postgres@labdb:[3639]:STATEMENT: ORDER BY
 s.confidence DESC;
---------------------- END OF LOG ----------------------
```

2. Définissez le paramètre `log_min_duration_statement`. L'exemple suivant illustre les informations écrites dans le fichier `postgresql.log` lorsque le paramètre est défini sur 1.

   Les requêtes qui dépassent la durée spécifiée dans le paramètre `log_min_duration_statement` sont enregistrées. Vous en trouverez un exemple ci-dessous. Vous pouvez consulter le fichier journal de votre cluster de bases de données Aurora PostgreSQL dans la console Amazon RDS.

```
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: statement: DROP
 table comments;
2022-10-05 19:05:19 UTC:52.95.4.1(6461):postgres@labdb:[6144]:LOG: duration:
 167.754 ms
2022-10-05 19:08:07 UTC::@:[355]:LOG: checkpoint starting: time
2022-10-05 19:08:08 UTC::@:[355]:LOG: checkpoint complete: wrote 11 buffers
 (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=1.013 s, sync=0.006 s,
 total=1.033 s; sync files=8, longest=0.004 s, average=0.001 s; distance=131028 kB,
 estimate=131028 kB
---------------------- END OF LOG ----------------------
```

Atténuation du risque d'exposition des mots de passe lors de l'utilisation de la journalisation de requêtes

Nous vous recommandons de garder `log_statement` sur `none` pour éviter de dévoiler les mots de passe. Si vous avez réglé `log_statement` sur `all`, `ddl` ou `mod`, nous vous recommandons de suivre une ou plusieurs des étapes suivantes.

- Pour le client, chiffrez les informations sensibles. Pour plus d'informations, consultez Options de chiffrement dans la documentation PostgreSQL. Utilisez les options `ENCRYPTED` (et

UNENCRYPTED) des instructions CREATE et ALTER. Pour plus d'informations, consultez CREATE USER dans la documentation PostgreSQL.

- Pour votre cluster de bases de données Aurora PostgreSQL, configurez et utilisez l'extension PostgreSQL Auditing (pgAudit). Cette extension supprime les informations sensibles dans les instructions CREATE et ALTER envoyées au journal. Pour plus d'informations, consultez Utilisation de pgAudit pour journaliser l'activité de la base de données.
- Limitez l'accès aux journaux CloudWatch.
- Utilisez des mécanismes d'authentification plus forts tels que IAM.

Surveillance des appels d'API Amazon Aurora dansAWS CloudTrail

AWS CloudTrail est un service AWS qui vous aide à auditer votre compte AWS. AWS CloudTrail est activé sur votre compte AWS lorsque vous le créez. Pour de plus amples informations sur CloudTrail, consultez le AWS CloudTrailGuide de l'utilisateur .

Rubriques

- Intégration de CloudTrail à Amazon Aurora
- Entrées de fichier journal Amazon Aurora

Intégration de CloudTrail à Amazon Aurora

Toutes les actions Amazon Aurora sont journalisées par CloudTrail. CloudTrail fournit un registre des actions entreprises par un utilisateur, un rôle ou un service AWS dans Amazon RDS.

Événements CloudTrail

CloudTrail capture tous les appels d'API pour Amazon RDS en tant qu'événements. Un événement représente une demande individuelle émise à partir d'une source quelconque et comprend des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. Les événements incluent les appels de la console Amazon RDS et les appels de code aux opérations de l'API Amazon RDS.

L'activité Amazon Aurora est enregistrée dans un événement CloudTrail dans Event history (Historique des événements). Vous pouvez utiliser la console CloudTrail pour afficher l'activité d'API et les événements enregistrés dans une région AWS au cours des 90 derniers jours. Pour de plus amples informations, veuillez consulter Affichage des événements avec l'historique des événements CloudTrail.

Journaux de suivi CloudTrail

Pour un enregistrement continu des événements dans votre compte AWS, y compris les événements pour Amazon Aurora, créez un journal d'activité. Un journal d'activité est une configuration qui permet la livraison d'événements à un compartiment Amazon S3 spécifié. CloudTrail fournit généralement des fichiers journaux dans les 15 minutes suivant une activité du compte.

> **Note**
>
> Si vous ne configurez pas de journal de suivi, vous pouvez toujours afficher les événements les plus récents dans la console CloudTrail dans Event history (Historique des événements).

Vous pouvez créer deux types de journaux d'activité pour un compte AWS : un journal d'activité qui s'applique à toutes les Régions ou un journal d'activité qui s'applique à une Région. Par défaut, lorsque vous créez un journal de suivi dans la console, il s’applique à toutes les régions .

En outre, vous pouvez configurer d’autres services AWS pour analyser plus en profondeur les données d’événement collectées dans les journaux CloudTrail et agir sur celles-ci. Pour plus d’informations, consultez :

- Présentation de la création d’un journal d’activité
- Intégrations et services pris en charge par CloudTrail
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers journaux CloudTrail de plusieurs régions et Réception de fichiers journaux CloudTrail de plusieurs comptes

Entrées de fichier journal Amazon Aurora

Les fichiers journaux CloudTrail peuvent contenir une ou plusieurs entrées. Les fichiers journaux CloudTrail ne constituent pas une trace de pile ordonnée d'appels d'API publics. Ils ne suivent aucun ordre précis.

L'exemple suivant montre une entrée de journal CloudTrail qui illustre l'action `CreateDBInstance`.

```
{
    "eventVersion": "1.04",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
```

```
    "eventTime": "2018-07-30T22:14:06Z",
    "eventSource": "rds.amazonaws.com",
    "eventName": "CreateDBInstance",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.15.42 Python/3.6.1 Darwin/17.7.0 botocore/1.10.42",
    "requestParameters": {
        "enableCloudwatchLogsExports": [
            "audit",
            "error",
            "general",
            "slowquery"
        ],
        "dBInstanceIdentifier": "test-instance",
        "engine": "mysql",
        "masterUsername": "myawsuser",
        "allocatedStorage": 20,
        "dBInstanceClass": "db.m1.small",
        "masterUserPassword": "****"
    },
    "responseElements": {
        "dBInstanceArn": "arn:aws:rds:us-east-1:123456789012:db:test-instance",
        "storageEncrypted": false,
        "preferredBackupWindow": "10:27-10:57",
        "preferredMaintenanceWindow": "sat:05:47-sat:06:17",
        "backupRetentionPeriod": 1,
        "allocatedStorage": 20,
        "storageType": "standard",
        "engineVersion": "8.0.28",
        "dbInstancePort": 0,
        "optionGroupMemberships": [
            {
                "status": "in-sync",
                "optionGroupName": "default:mysql-8-0"
            }
        ],
        "dBParameterGroups": [
            {
                "dBParameterGroupName": "default.mysql8.0",
                "parameterApplyStatus": "in-sync"
            }
        ],
        "monitoringInterval": 0,
        "dBInstanceClass": "db.m1.small",
```

```
        "readReplicaDBInstanceIdentifiers": [],
        "dBSubnetGroup": {
            "dBSubnetGroupName": "default",
            "dBSubnetGroupDescription": "default",
            "subnets": [
                {
                    "subnetAvailabilityZone": {"name": "us-east-1b"},
                    "subnetIdentifier": "subnet-cbfff283",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1e"},
                    "subnetIdentifier": "subnet-d7c825e8",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1f"},
                    "subnetIdentifier": "subnet-6746046b",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1c"},
                    "subnetIdentifier": "subnet-bac383e0",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1d"},
                    "subnetIdentifier": "subnet-42599426",
                    "subnetStatus": "Active"
                },
                {
                    "subnetAvailabilityZone": {"name": "us-east-1a"},
                    "subnetIdentifier": "subnet-da327bf6",
                    "subnetStatus": "Active"
                }
            ],
            "vpcId": "vpc-136a4c6a",
            "subnetGroupStatus": "Complete"
        },
        "masterUsername": "myawsuser",
        "multiAZ": false,
        "autoMinorVersionUpgrade": true,
        "engine": "mysql",
        "cACertificateIdentifier": "rds-ca-2015",
```

```
        "dbiResourceId": "db-ETDZIIXHEWY5N7GXVC4SH7H5IA",
        "dBSecurityGroups": [],
        "pendingModifiedValues": {
            "masterUserPassword": "****",
            "pendingCloudwatchLogsExports": {
                "logTypesToEnable": [
                    "audit",
                    "error",
                    "general",
                    "slowquery"
                ]
            }
        },
        "dBInstanceStatus": "creating",
        "publiclyAccessible": true,
        "domainMemberships": [],
        "copyTagsToSnapshot": false,
        "dBInstanceIdentifier": "test-instance",
        "licenseModel": "general-public-license",
        "iAMDatabaseAuthenticationEnabled": false,
        "performanceInsightsEnabled": false,
        "vpcSecurityGroups": [
            {
                "status": "active",
                "vpcSecurityGroupId": "sg-f839b688"
            }
        ]
    },
    "requestID": "daf2e3f5-96a3-4df7-a026-863f96db793e",
    "eventID": "797163d3-5726-441d-80a7-6eeb7464acd4",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


Comme indiqué dans l'élément `userIdentity` de l'exemple précédent, chaque événement ou entrée de journal contient des informations sur la personne qui a généré la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été effectuée avec des informations d’identification d’utilisateur root ou IAM.
- Si la demande a été effectuée avec des informations d’identification de sécurité temporaires pour un rôle ou un utilisateur fédéré.
- Si la demande a été effectuée par un autre service AWS.

Pour de plus amples informations sur `userIdentity`, veuillez consulter la section Élément userIdentity CloudTrail. Pour plus d'informations sur `CreateDBInstance` et d'autres actions Amazon Aurora, veuillez consulter la Référence d'API Amazon RDS.

Surveillance d’Amazon Aurora à l’aide des flux d’activité de base de données

En utilisant les flux d’activité de base de données, vous pouvez surveiller en temps quasi réel les flux d’activité de base de données.

Rubriques

Présentation des flux d’activité de base de données

En tant qu’administrateur de base de données Amazon Aurora, vous devez protéger votre base de données et satisfaire aux exigences en matière de conformité et de réglementation. Une politique consiste à intégrer les flux d’activités de base de données avec vos outils de surveillance. De cette façon, vous surveillez l’activité d’audit dans votre cluster Amazon Aurora et définissez des alarmes.

Les menaces de sécurité sont à la fois externes et internes. Pour vous protéger contre des menaces internes, vous pouvez contrôler l’accès administrateur aux flux de données à l’aide de la fonction Database Activity Streams. Les administrateurs de base de données n’ont pas accès à la collecte, à la transmission, au stockage et au traitement des flux.

Table des matières

Fonctionnement des flux d’activité de base de données

Dans Amazon Aurora, vous démarrez un flux d’activité de base de données au niveau du cluster. Toutes les instances de base de données de votre cluster disposent de flux d’activité de base de données activés.

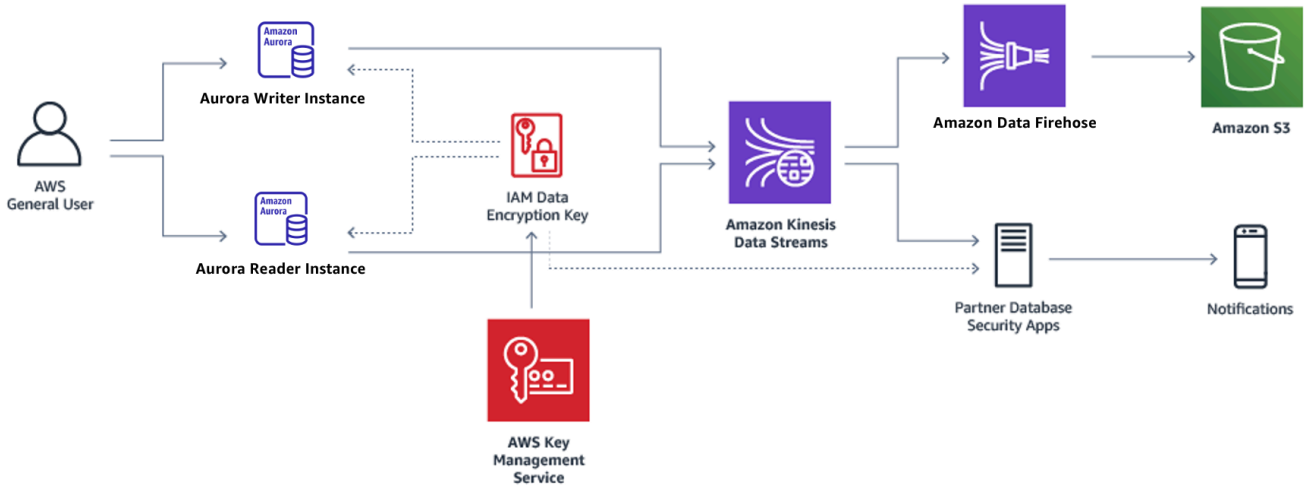
Votre cluster de base de données Aurora envoie (push) les activités vers un flux de données Amazon Kinesis en temps quasi réel. Le flux Kinesis est créé automatiquement. Dans Kinesis, vous pouvez configurer des services AWS tels qu’Amazon Data Firehose et AWS Lambda pour consommer le flux et stocker les données.

> **Important**
>
> L’utilisation de la fonction de flux d’activité de base de données dans Amazon Aurora est gratuite, mais Amazon Kinesis facture un flux de données. Pour plus d’informations, consultez la Tarification d’Amazon Kinesis Data Streams.

Si vous utilisez une base de données globale Aurora, démarrez un flux d’activité de base de données sur chaque cluster de bases de données séparément. Chaque cluster fournit des données d’audit à son propre flux Kinesis au sein de sa propre Région AWS. Les flux d’activité ne fonctionnent pas différemment lors d’un basculement. Ils continuent à auditer votre base de données globale comme d’habitude.

Vous pouvez configurer les applications de gestion de la conformité pour qu’elles consomment les flux d’activité des bases de données. Pour Aurora PostgreSQL, les applications de gestion de la conformité sont Security Guardium d’IBM et SecureSphere Database Audit and Protection d’Imperva. Ces applications peuvent utiliser le flux pour générer des alertes et auditer l’activité sur votre cluster de bases de données Aurora.

Le graphique suivant montre un cluster de bases de données Aurora configuré avec Amazon Data Firehose.



Mode asynchrone et synchrone pour les flux d'activité de base de données

Vous pouvez choisir que la session de base de données gère les événements d'activité de base de données dans l'un des modes suivants :

- Mode asynchrone : quand une session de base de données génère un événement de flux d'activité, la session revient immédiatement aux activités normales. En arrière-plan, l'événement du flux d'activité est converti en enregistrement durable. Si une erreur se produit pendant la tâche en arrière-plan, un événement RDS est envoyé. Cet événement indique le début et la fin de toute fenêtre de temps au cours de laquelle des enregistrements d'événement de flux d'activité ont pu être perdus.

  Le mode asynchrone favorise les performances de la base de données plutôt que la précision du flux d'activité.

  > **Note**
  >
  > Le mode asynchrone est disponible pour Aurora PostgreSQL et Aurora MySQL.

- Mode synchrone : quand une session de base de données génère un événement de flux d'activité, la session bloque d'autres activités jusqu'à ce que l'événement devienne durable. Si l'événement ne peut pas devenir durable pour une raison quelconque, la session de base de données reprend une activité normale. Cependant, un événement RDS est envoyé, indiquant que ces

enregistrements de flux peuvent être perdus pour un certain temps. Un deuxième événement RDS est envoyé après que le système est redevenu sain.

Le mode synchrone favorise la précision du flux d'activité plutôt que les performances de la base de données.

> **Note**
>
> Le mode synchrone est disponible pour Aurora PostgreSQL. Vous ne pouvez pas utiliser le mode synchrone avec Aurora MySQL.

Exigences et limites pour les flux d'activité de base de données

Dans Aurora, les flux d'activité de base de données présentent les limites et les exigences suivantes :

- Amazon Kinesis est nécessaire pour les flux d'activité des bases de données.
- AWS Key Management Service (AWS KMS) est nécessaire pour les flux d'activité de la base de données, car ces éléments sont toujours chiffrés.
- L'application d'un chiffrement supplémentaire à votre flux de données Amazon Kinesis est incompatible avec les flux d'activité de la base de données, qui sont déjà chiffrés avec votre clé AWS KMS.
- Démarrez le flux d'activité de votre base de données au niveau du cluster de bases de données. Si vous ajoutez une instance de base de données à votre cluster, vous n'avez pas besoin de lancer un flux d'activité sur l'instance : elle est automatiquement auditée.
- Dans une base de données globale Aurora, assurez-vous de démarrer un flux d'activité sur chaque cluster de bases de données séparément. Chaque cluster fournit des données d'audit à son propre flux Kinesis au sein de sa propre Région AWS.
- Dans Aurora PostgreSQL, arrêtez le flux d'activité de la base de données avant une mise à niveau majeure de la version. Vous pouvez lancer le flux d'activité de la base de données une fois la mise à niveau terminée.

Disponibilité des régions et des versions

La disponibilité et la prise en charge des fonctions varient selon les versions spécifiques de chaque moteur de base de données Aurora, et selon les Régions AWS. Pour plus d'informations sur la disponibilité des versions et des régions avec Aurora et les flux d'activité des bases de données,

consultez Régions et moteurs de base de données Aurora pris en charge pour les flux d'activité de base de données.

Classes d'instance de base de données prises en charge pour les flux d'activité de base de données

Pour Aurora MySQL, vous pouvez utiliser des flux d'activité de base de données avec les classes d'instance de base de données suivantes :

- db.r8g.*large
- db.r7g.*large
- db.r7i.*large
- db.r6g.*large
- db.r6i.*large
- db.r5.*large
- db.x2g.*

Pour Aurora PostgreSQL, vous pouvez utiliser des flux d'activité de base de données avec les classes d'instance de base de données suivantes :

- db.r8g.*large
- db.r7i.*large
- db.r7g.*large
- db.r6g.*large
- db.r6i.*large
- db.r6id.*large
- db.r5.*large
- db.r4.*large
- db.x2g.*

Prérequis réseau pour les flux d'activité de base de données Aurora MySQL

Dans la section suivante, vous trouverez comment configurer votre cloud privé virtuel (VPC) pour l'utiliser avec des flux d'activité de base de données.

> **Note**
>
> Les prérequis du réseau Aurora MySQL s'appliquent aux versions de moteur suivantes :
>
> - Aurora MySQL versions 2 à 2.11.3
> - Aurora MySQL version 2.12.0
> - Aurora MySQL versions 3 à 3.04.2

Rubriques

Prérequis pour les points de terminaison AWS KMS

Les instances d'un cluster MySQL Aurora qui utilisent des flux d'activité doivent pouvoir accéder aux points de terminaison AWS KMS. Assurez-vous que cette exigence est satisfaite avant d'activer les flux d'activité de base de données pour votre cluster Aurora MySQL. Si le cluster Aurora est accessible au public, cette exigence est remplie automatiquement.

> **Important**
>
> Si le cluster de bases de données Aurora MySQL ne peut pas accéder au point de terminaison AWS KMS, le flux d'activité s'arrête. Dans ce cas, Aurora vous informe de ce problème à l'aide d'événements RDS.

Conditions préalables à la mise à disposition du public

Pour qu'un cluster de bases de données Aurora soit public, il doit répondre aux exigences suivantes :

- Accessible publiquement est Oui sur la page de détails du cluster AWS Management Console.
- Le cluster de base de données se trouve dans un sous-réseau public Amazon VPC. Pour plus d'informations sur les instances de base de données accessibles publiquement, consultez Utilisation d'un cluster de bases de données dans un VPC. Pour plus d'informations sur les sous-réseaux Amazon VPC publics, consultez VPC et sous-réseaux.

Conditions préalables à la disponibilité privée

Si votre cluster de bases de données Aurora se trouve dans un sous-réseau public VPC et n'est pas accessible publiquement, il est privé. Pour conserver votre cluster privé et l'utiliser avec des flux d'activité de base de données, vous disposez des options suivantes :

- Configurez la traduction d'adresses réseau (NAT) dans votre VPC. Pour plus d'informations, consultez Passerelles NAT.
- Créez un point de terminaison AWS KMS dans votre VPC. Cette option est recommandée car elle est plus facile à configurer.

Pour créer un point de terminaison AWS KMS dans votre VPC

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Dans le panneau de navigation, choisissez Points de terminaison.
3. Choisissez Create Endpoint (Créer un point de terminaison).

   La page Créer un point de terminaison s'affiche.
4. Procédez comme suit :
   - Pour Catégorie de service, choisissez Services AWS.
   - Pour Service Name (Nom du service), choisissez com.amazonaws.***region***.kms, où *region* correspond à la Région AWS dans laquelle votre cluster est situé.
   - Pour VPC, choisissez le VPC dans lequel votre cluster est situé.
5. Choisissez Create Endpoint (Créer un point de terminaison).

Pour plus d'informations sur la configuration des points de terminaison d'un VPC, consultez Points de terminaison d'un VPC.

Démarrage d'un flux d'activité de base de données

Pour surveiller l'activité de la base de données pour toutes les instances de votre cluster de bases de données Aurora, démarrez un flux d'activité au niveau du cluster. Les instances de base de données que vous ajoutez au cluster sont automatiquement surveillées. Si vous utilisez une base de données globale Aurora, démarrez un flux d'activité de base de données sur chaque cluster de bases de données séparément. Chaque cluster fournit des données d'audit à son propre flux Kinesis au sein de sa propre Région AWS.

Lorsque vous démarrez un flux d'activité, chaque événement d'activité de base de données que vous avez configuré dans la politique d'audit génère un événement de flux d'activité. Des commandes SQL telles que `CONNECT` et `SELECT` génèrent des événements d'accès. Des commandes SQL telles que `CREATE` et `INSERT` génèrent des événements de modification.

Console

Pour démarrer un flux d'activité de base de données

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Choisissez le cluster de bases de données sur lequel vous souhaitez démarrer un flux d'activité.
4. Pour Actions, choisissez Start activity stream (Démarrer le flux d'activité).

   La fenêtre Démarrer un flux d'activité de base de données : *nom* s'affiche, où *nom* est votre cluster de bases de données.
5. Définissez les paramètres suivants :

   - Pour une AWS KMS key, choisissez une clé dans la liste des AWS KMS keys.

     > **Note**
     >
     > Si votre cluster Aurora MySQL ne parvient pas à accéder aux clés KMS, suivez les instructions indiqué dans Prérequis réseau pour les flux d'activité de base de données Aurora MySQL pour activer cet accès au préalable.

     Aurora utilise la clé KMS pour chiffrer la clé qui va à son tour chiffrer l'activité de base de données. Choisissez une clé KMS différente de la clé par défaut. Pour plus d'informations sur les clés de chiffrement et AWS KMS, consultez Présentation d'AWS Key Management Service dans le Manuel du développeur AWS Key Management Service.
   - Pour le Database activity stream mode (Mode de flux d'activité de base de données), choisissez Asynchronous (Asynchrone) ou Synchronous (Synchrone).

> **Note**
>
> Ce choix s'applique uniquement à Aurora PostgreSQL. Pour Aurora MySQL, vous ne pouvez utiliser que le mode asynchrone.

- Choisissez Immédiatement.

  Lorsque vous choisissez Immédiatement, le cluster de base de donnéesredémarre tout de suite. Si vous choisissez Pendant la prochaine fenêtre de maintenance, le cluster DB ne redémarre pas tout de suite. Dans ce cas, le flux d'activité de base de données ne démarre pas avant la prochaine fenêtre de maintenance.

6. Choisissez Démarrer le flux d'activité de base de données.

   Le statut pour le cluster de bases de données indique que le flux d'activité démarre.

> **Note**
>
> Si l'erreur `You can't start a database activity stream in this configuration` s'affiche, vérifiez Classes d'instance de base de données prises en charge pour les flux d'activité de base de données pour voir si votre cluster de bases de données utilise une classe d'instance prise en charge.

AWS CLI

Pour démarrer des flux d'activité de base de données pour un cluster de bases de données, configurez le cluster de bases de données à l'aide de la commande AWS CLI start-activity-stream.

- `--resource-arn` *`arn`* – Spécifie l'Amazon Resource Name (ARN) du cluster de base de données.
- `--mode` *`sync-or-async`* – Spécifie le mode synchrone (`sync`) ou asynchrone (`async`). Pour Aurora PostgreSQL, vous pouvez choisir l'une ou l'autre valeur. Pour Aurora MySQL, spécifiez `async`.
- `--kms-key-id` *`key`* – Spécifie l'identifiant de clé KMS pour le chiffrement des messages dans le flux d'activité de base de données. L'identifiant de clé KMS AWS est l'ARN de clé, l'ID de clé, l'ARN d'alias ou le nom d'alias pour la AWS KMS key.

L'exemple suivant démarre un flux d'activité de base de données pour un cluster de bases de données en mode asynchrone.

Pour Linux, macOS ou Unix :

```
aws rds start-activity-stream \
    --mode async \
    --kms-key-id my-kms-key-arn \
    --resource-arn my-cluster-arn \
    --apply-immediately
```

Pour Windows :

```
aws rds start-activity-stream ^
    --mode async ^
    --kms-key-id my-kms-key-arn ^
    --resource-arn my-cluster-arn ^
    --apply-immediately
```

Amazon RDS API

Pour démarrer des flux d'activité de base de données pour un cluster de bases de données, configurez le cluster à l'aide de l'opération StartActivityStream.

Appelez l'action avec les paramètres ci-dessous :

- `Region`
- `KmsKeyId`
- `ResourceArn`
- `Mode`

> **Note**
>
> Si un message d'erreur vous informe que la version actuelle d'Aurora PostgreSQL ne permet pas de démarrer un flux d'activité de base de données, appliquez le dernier correctif pour Aurora PostgreSQL avant de démarrer un flux d'activité de base de données. Pour plus d'informations sur la mise à niveau de la base de données Aurora PostgreSQL, consultez Mise à niveau des clusters de bases de données Amazon Aurora.

Voici les versions de correctif minimales pour démarrer les flux d'activité de base de données avec Aurora PostgreSQL.

- 3.4.15 (11.9.15), 11.21.10
- 12.9.15, 12.15.9, 12.16.10, 12.17.7, 12.18.5, 12.19.4, 12.20.3, 12.22.3
- 13.9.12, 13.11.9, 13.12.10, 13.13.7, 13.14.5, 13.15.4, 13.16.3, 13.18.3
- 14.6.12, 14.8.9, 14.9.10, 14.10.7, 14.11.5, 14.12.4, 14.13.3, 14.15.3
- 15.3.9, 15.4.10, 15.5.7, 15.6.5, 15.7.4, 15.8.3, 15.10.3
- 16.1.7, 16.2.5, 16.3.4, 16.4.3, 16.6.3

Obtention de l'état d'un flux d'activité de base de données

Vous pouvez obtenir le statut d'un flux d'activité en utilisant la console ou AWS CLI.

console

Obtention de l'état d'un flux d'activité de base de données

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis le lien du cluster de base de données.
3. Choisissez l'onglet Configuration et cochez l'option Flux d'activité de base de données pour obtenir l'état.

AWS CLI

Vous pouvez obtenir la configuration du flux d'activité pour un cluster de base de données en réponse à une demande CLI describe-db-clusters .

L'exemple suivant décrit *my-cluster*.

```
aws rds --region my-region describe-db-clusters --db-cluster-identifier my-cluster
```

Voici un exemple de réponse JSON. Les champs suivants s'affichent :

- `ActivityStreamKinesisStreamName`
- `ActivityStreamKmsKeyId`

- `ActivityStreamStatus`
- `ActivityStreamMode`
-

Ces champs sont les mêmes pour Aurora PostgreSQL et Aurora MySQL, sauf que `ActivityStreamMode` est toujours `async` pour Aurora MySQL, tandis que pour Aurora PostgreSQL il peut être `sync` ou `async`.

```
{
    "DBClusters": [
        {
      "DBClusterIdentifier": "my-cluster",
            ...
            "ActivityStreamKinesisStreamName": "aws-rds-das-cluster-
A6TSYXITZCZXJHIRVFUBZ5LTWY",
            "ActivityStreamStatus": "starting",
            "ActivityStreamKmsKeyId": "12345678-abcd-efgh-ijkl-bd041f170262",
            "ActivityStreamMode": "async",
            "DbClusterResourceId": "cluster-ABCD123456"
            ...
        }
    ]
}
```

API RDS

Vous pouvez obtenir la configuration du flux d'activité pour un cluster de base de données en réponse à une opération DescribeDBClusters .

Arrêt d’un flux d’activité de base de données

Vous pouvez arrêter un flux d’activité à partir de la console ou d’AWS CLI.

Si vous supprimez votre cluster de bases de données, le flux d’activité est arrêté et le flux Amazon Kinesis sous-jacent est supprimé automatiquement.

Console

Pour désactiver un flux d’activité

1. Ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.

2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez un cluster de bases de données pour lequel vous souhaitez arrêter le flux d'activité de base de données.
4. Pour Actions, choisissez Stop activity stream (Arrêter le flux d'activité). La fenêtre Database Activity Stream (Flux d'activité de base de données) apparaît.

    a. Choisissez Immédiatement.

        Lorsque vous choisissez Immédiatement, le cluster de base de donnéesredémarre tout de suite. Si vous choisissez Pendant la prochaine fenêtre de maintenance, le cluster DB ne redémarre pas tout de suite. Dans ce cas, le flux d'activité de base de données ne s'arrête pas avant la prochaine fenêtre de maintenance.

    b. Choisissez Continuer.

AWS CLI

Pour arrêter les flux d'activité de base de données pour votre cluster de bases de données, configurez le cluster de bases de données à l'aide de la commande AWS CLI stop-activity-stream. Identifiez la région AWS pour le cluster de base de données avec le paramètre `--region`. Le paramètre `--apply-immediately` est facultatif.

Pour Linux, macOS ou Unix :

```
aws rds --region MY_REGION \
    stop-activity-stream \
    --resource-arn MY_CLUSTER_ARN \
    --apply-immediately
```

Pour Windows :

```
aws rds --region MY_REGION ^
    stop-activity-stream ^
    --resource-arn MY_CLUSTER_ARN ^
    --apply-immediately
```

API RDS

Pour arrêter les flux d'activité de base de données pour votre cluster de bases de données, configurez ce cluster à l'aide de l'opération StopActivityStream. Identifiez la région AWS pour le

cluster de base de données avec le paramètre `Region`. Le paramètre `ApplyImmediately` est facultatif.

Surveillance des flux d'activité de base de données

Les flux d'activité de base de données surveillent et rapportent les activités. Le flux d'activité est collecté et transmis à Amazon Kinesis. Depuis Kinesis, vous pouvez surveiller le flux d'activité ou d'autres services et applications peuvent utiliser le flux d'activité pour une analyse plus approfondie. Vous pouvez trouver le nom du flux Kinesis sous-jacent à l'aide de la commande `describe-db-clusters` de AWS CLI ou de l'opération `DescribeDBClusters` de l'API RDS.

Aurora gère le flux Kinesis pour vous comme suit :

- Aurora crée automatiquement le flux Kinesis avec une période de rétention de 24 heures.
- Aurora met à l'échelle le flux Kinesis si nécessaire.
- Si vous arrêtez le flux d'activité de base de données ou supprimez le cluster de bases de données, Aurora supprime le flux Kinesis.

Les catégories d'activité suivantes sont surveillées et incluses dans le journal d'audit de flux d'activité :

- Commandes SQL – Toutes les commandes SQL sont auditées, ainsi que les instructions préparées, les fonctions intégrées et les fonctions en PL/SQL. Les appels aux procédures stockées sont vérifiés. Toutes les instructions SQL émises dans des procédures ou fonctions stockées sont également vérifiées.
- Autres informations de bases de données – L'activité surveillée inclut l'instruction SQL complète, le nombre des lignes affectées par les commandes DML, les objets consultés et le nom unique de base de données. Pour Aurora PostgreSQL, les flux d'activité de base de données surveillent également les variables de liaison et les paramètres de procédure stockée.

> **Important**
>
> Le texte SQL complet de chaque instruction est visible dans le journal d'audit du flux d'activité, y compris les données sensibles. Cependant, les mots de passe des utilisateurs de base de données sont expurgés si Aurora peut les déterminer d'après le contexte, comme dans l'instruction SQL suivante.

```
ALTER ROLE role-name WITH password
```

- Informations de connexion – L'activité surveillée inclut les informations de session et de réseau, l'ID de processus serveur et les codes de sortie.

Si un flux d'activité rencontre un échec pendant la surveillance de votre instance de base de données, vous en êtes informé via des événements RDS.

Les sections ci-après vous permettent d'accéder aux flux d'activité des bases de données, de les auditer et de les traiter.

Rubriques

- Accès à un flux d'activité depuis Amazon Kinesis
- Contenu du journal d'audit et exemples pour les flux d'activité de base de données
- Tableau JSON databaseActivityEventList pour les flux d'activité de base de données
- Traitement d'un flux d'activité de base de données à l'aide du kit AWS SDK

Accès à un flux d'activité depuis Amazon Kinesis

Lorsque vous activez un flux d'activité pour un cluster de bases de données, un flux Kinesis est créé pour vous. Depuis Kinesis, vous pouvez surveiller l'activité de votre base de données en temps réel. Pour effectuer des analyses plus poussées de l'activité de base de données, vous pouvez connecter votre flux Kinesis à des applications grand public. Vous pouvez également connecter le flux à des applications de gestion de la conformité telles que Security Guardium d'IBM ou SecureSphere Database Audit and Protection d'Imperva.

Vous pouvez accéder à votre flux Kinesis à partir de la console RDS ou de la console Kinesis.

Pour accéder à un flux d'activité depuis Kinesis avec la console RDS

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Choisissez le cluster de bases de données où vous souhaitez démarrer un flux d'activité.
4. Choisissez Configuration.

5. Sous Database activity stream (Flux d'activité de la base de données), choisissez le lien sous Kinesis stream (Flux Kinesis).
6. Dans la console Kinesis, choisissez Monitoring (Surveillance) pour commencer à observer l'activité de la base de données.

Pour accéder à un flux d'activité depuis Kinesis avec la console Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Choisissez votre flux d'activité dans la liste des flux Kinesis.

   Le nom d'un flux d'activité comprend le préfixe `aws-rds-das-cluster-` suivi de l'ID de ressource du cluster de bases de données. Voici un exemple.

   ```
   aws-rds-das-cluster-NHVOV4PCLWHGF52NP
   ```

   Pour utiliser la console Amazon RDS afin de trouver l'ID de ressource pour le cluster de bases de données, choisissez votre cluster de bases de données dans la liste des bases de données, puis choisissez l'onglet Configuration.

   Pour trouver le nom complet de flux Kinesis pour un flux d'activité à l'aide de AWS CLI, utilisez une demande de CLI describe-db-clusters et notez la valeur de `ActivityStreamKinesisStreamName` dans la réponse.
3. Choisissez Surveillance pour commencer à observer l'activité de base de données.

Pour plus d'informations sur l'utilisation d'Amazon Kinesis, consultez En quoi consiste le service Amazon Kinesis Data Streams ?.

Contenu du journal d'audit et exemples pour les flux d'activité de base de données

Les événements surveillés sont représentés dans le flux d'activité de base de données sous la forme de chaînes JSON. La structure se compose d'un objet JSON contenant un `DatabaseActivityMonitoringRecord`, qui contient lui-même un tableau des événements d'activité `databaseActivityEventList`.

> **Note**
>
> Pour les flux d'activité de base de données, le tableau JSON `paramList` n'inclut pas les valeurs null des applications Hibernate.

Rubriques

Exemples de journaux d'audit de flux d'activité

Vous trouverez ci-après des exemples de journaux d'audits JSON déchiffrés d'enregistrements d'événements d'activité.

Example Enregistrement d'événement d'activité d'une instruction Aurora PostgreSQL CONNECT SQL

L'enregistrement d'événement d'activité suivant indique une connexion à l'aide d'une instruction SQL `CONNECT` (`command`) par un client psql (`clientApplication`).


```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-10-30 00:39:49.940668+00",
          "logTime": "2019-10-30 00:39:49.990579+00",
          "statementId": 1,
          "substatementId": 1,
          "objectType": null,
          "command": "CONNECT",
          "objectName": null,
          "databaseName": "postgres",
```

```
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "49804",
          "sessionId": "5ce5f7f0.474b",
          "rowCount": null,
          "commandText": null,
          "paramList": [],
          "pid": 18251,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "MISC",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```

Example Enregistrement d'événement d'activité d'une instruction Aurora MySQL CONNECT SQL

L'enregistrement d'événement d'activité suivant indique une connexion à l'aide d'une instruction SQL `CONNECT` (`command`) par un client mysql (`clientApplication`).

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:13.267214+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"rdsadmin",
      "databaseName":"",
      "remoteHost":"localhost",
```

```
      "remotePort":"11053",
      "command":"CONNECT",
      "commandText":"",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"",
      "statementId":0,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"725121",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:13.267207+00",
      "endTime":"2020-05-22 18:07:13.267213+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

Example Registre d'événement d'activité d'une instruction Aurora PostgreSQL CREATE TABLE

L'exemple suivant montre un événement `CREATE TABLE` pour Aurora PostgreSQL.

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-05-24 00:36:54.403455+00",
          "logTime": "2019-05-24 00:36:54.494235+00",
          "statementId": 2,
          "substatementId": 1,
```

```
          "objectType": null,
          "command": "CREATE TABLE",
          "objectName": null,
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "34534",
          "sessionId": "5ce73c6f.7e64",
          "rowCount": null,
          "commandText": "create table my_table (id serial primary key, name
 varchar(32));",
          "paramList": [],
          "pid": 32356,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "DDL",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```

Example Enregistrement d'événement d'activité d'une instruction CREATE TABLE Aurora MySQL

L'exemple suivant montre une instruction `CREATE TABLE` pour Aurora MySQL. L'opération est représentée sous la forme de deux enregistrements d'événements distincts. Un événement a `"class":"MAIN"`. L'autre événement a `"class":"AUX"`. Les messages peuvent arriver dans n'importe quel ordre. Le champ `logTime` de l'événement `MAIN` est toujours antérieur au champ `logTime` des événements `AUX` correspondants.

L'exemple suivant montre l'événement avec une valeur `class` de `MAIN`.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
```

```
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:07:12.250221+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"QUERY",
      "commandText":"CREATE TABLE test1 (id INT)",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65459278,
      "substatementId":1,
      "exitCode":"0",
      "sessionId":"725118",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:12.226384+00",
      "endTime":"2020-05-22 18:07:12.250222+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

L'exemple suivant montre l'événement correspondant avec une valeur `class` de AUX.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
```

```
    {
      "logTime":"2020-05-22 18:07:12.247182+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"CREATE",
      "commandText":"test1",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65459278,
      "substatementId":2,
      "exitCode":"",
      "sessionId":"725118",
      "rowCount":0,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:07:12.226384+00",
      "endTime":"2020-05-22 18:07:12.247182+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"AUX"
    }
  ]
}
```

Example Registre d’événement d’activité d’une instruction Aurora PostgreSQL SELECT

L’exemple suivant montre un événement `SELECT` .

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":
    {
      "type":"DatabaseActivityMonitoringRecord",
```

```
      "clusterId":"cluster-4HNY5V4RRNPKKYB7ICFKE5JBQQ",
      "instanceId":"db-FZJTMYKCXQBUUZ6VLU7NW3ITCM",
      "databaseActivityEventList":[
        {
          "startTime": "2019-05-24 00:39:49.920564+00",
          "logTime": "2019-05-24 00:39:49.940668+00",
          "statementId": 6,
          "substatementId": 1,
          "objectType": "TABLE",
          "command": "SELECT",
          "objectName": "public.my_table",
          "databaseName": "postgres",
          "dbUserName": "rdsadmin",
          "remoteHost": "172.31.3.195",
          "remotePort": "34534",
          "sessionId": "5ce73c6f.7e64",
          "rowCount": 10,
          "commandText": "select * from my_table;",
          "paramList": [],
          "pid": 32356,
          "clientApplication": "psql",
          "exitCode": null,
          "class": "READ",
          "serverVersion": "2.3.1",
          "serverType": "PostgreSQL",
          "serviceName": "Amazon Aurora PostgreSQL-Compatible edition",
          "serverHost": "172.31.3.192",
          "netProtocol": "TCP",
          "dbProtocol": "Postgres 3.0",
          "type": "record",
          "errorMessage": null
        }
      ]
    },
   "key":"decryption-key"
}
```



```
{
    "type": "DatabaseActivityMonitoringRecord",
    "clusterId": "",
    "instanceId": "db-4JCWQLUZVFYP7DIWP6JVQ77O3Q",
    "databaseActivityEventList": [
        {
```

```
            "class": "TABLE",
            "clientApplication": "Microsoft SQL Server Management Studio - Query",
            "command": "SELECT",
            "commandText": "select * from [testDB].[dbo].[TestTable]",
            "databaseName": "testDB",
            "dbProtocol": "SQLSERVER",
            "dbUserName": "test",
            "endTime": null,
            "errorMessage": null,
            "exitCode": 1,
            "logTime": "2022-10-06 21:24:59.9422268+00",
            "netProtocol": null,
            "objectName": "TestTable",
            "objectType": "TABLE",
            "paramList": null,
            "pid": null,
            "remoteHost": "local machine",
            "remotePort": null,
            "rowCount": 0,
            "serverHost": "172.31.30.159",
            "serverType": "SQLSERVER",
            "serverVersion": "15.00.4073.23.v1.R1",
            "serviceName": "sqlserver-ee",
            "sessionId": 62,
            "startTime": null,
            "statementId": "0x03baed90412f564fad640ebe51f89b99",
            "substatementId": 1,
            "transactionId": "4532935",
            "type": "record",
            "engineNativeAuditFields": {
                "target_database_principal_id": 0,
                "target_server_principal_id": 0,
                "target_database_principal_name": "",
                "server_principal_id": 2,
                "user_defined_information": "",
                "response_rows": 0,
                "database_principal_name": "dbo",
                "target_server_principal_name": "",
                "schema_name": "dbo",
                "is_column_permission": true,
                "object_id": 581577110,
                "server_instance_name": "EC2AMAZ-NFUJJNO",
                "target_server_principal_sid": null,
                "additional_information": "",
```

```
                "duration_milliseconds": 0,
                "permission_bitmask": "0x00000000000000000000000000000001",
                "data_sensitivity_information": "",
                "session_server_principal_name": "test",
                "connection_id": "AD3A5084-FB83-45C1-8334-E923459A8109",
                "audit_schema_version": 1,
                "database_principal_id": 1,
                "server_principal_sid":
 "0x0105000000000005150000000bdc2795e2d0717901ba6998cf4010000",
                "user_defined_event_id": 0,
                "host_name": "EC2AMAZ-NFUJJNO"
            }
        }
    ]
}
```

Example Enregistrement d'événement d'activité d'une instruction SELECT Aurora MySQL

L'exemple suivant montre un événement `SELECT`.

L'exemple suivant montre l'événement avec une valeur `class` de `MAIN`.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "clusterId":"cluster-some_id",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:29:57.986467+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"QUERY",
      "commandText":"SELECT * FROM test1 WHERE id < 28",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65469218,
      "substatementId":1,
      "exitCode":"0",
```

```
      "sessionId":"726571",
      "rowCount":2,
      "serverHost":"master",
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:29:57.986364+00",
      "endTime":"2020-05-22 18:29:57.986467+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"MAIN"
    }
  ]
}
```

L'exemple suivant montre l'événement correspondant avec une valeur `class` de AUX.

```
{
  "type":"DatabaseActivityMonitoringRecord",
  "instanceId":"db-some_id",
  "databaseActivityEventList":[
    {
      "logTime":"2020-05-22 18:29:57.986399+00",
      "type":"record",
      "clientApplication":null,
      "pid":2830,
      "dbUserName":"master",
      "databaseName":"test",
      "remoteHost":"localhost",
      "remotePort":"11054",
      "command":"READ",
      "commandText":"test1",
      "paramList":null,
      "objectType":"TABLE",
      "objectName":"test1",
      "statementId":65469218,
      "substatementId":2,
      "exitCode":"",
      "sessionId":"726571",
      "rowCount":0,
      "serverHost":"master",
```

```
      "serverType":"MySQL",
      "serviceName":"Amazon Aurora MySQL",
      "serverVersion":"MySQL 5.7.12",
      "startTime":"2020-05-22 18:29:57.986364+00",
      "endTime":"2020-05-22 18:29:57.986399+00",
      "transactionId":"0",
      "dbProtocol":"MySQL",
      "netProtocol":"TCP",
      "errorMessage":"",
      "class":"AUX"
    }
  ]
}
```

Objet JSON DatabaseActivityMonitoringRecords

Les enregistrements d'événement d'activité de base de données se trouvent dans un objet JSON qui contient les informations suivantes.

Champ JSON	Type de données	Description
`type`	chaîne	Type de l'enregistrement JSON. La valeur est `DatabaseActivityMonitoringRecords` .
`version`	chaîne	Version des enregistrements de surveillance d'activité de base de données. La version des enregistrements d'activité de base de données générés dépend de la version du moteur du cluster de bases de données. • Les enregistrements d'activité de base de données version 1.1 sont générés pour les clusters de bases de données Aurora PostgreSQL exécutant un moteur version 10.10 et versions mineures

Champ JSON	Type de données	Description
		ultérieures ou un moteur versions 11.5 et ultérieures. • Les enregistrements d'activité de base de données version 1.0 sont générés pour des clusters de bases de données Aurora PostgreSQL exécutant un moteur version 10.7 ou 11.4. Tous les champs suivants sont à la fois dans la version 1.0 et dans la version 1.1, sauf indication spécifique.
databaseActivityEvents	chaîne	Objet JSON qui contient les événements d'activité.
key	chaîne	Clé de chiffrement que vous utilisez pour déchiffrer Tableau JSON databaseActivityEventList

Objet JSON databaseActivityEvents

L'objet JSON `databaseActivityEvents` contient les informations suivantes.

Champs de niveau supérieur dans l'enregistrement JSON

Chaque événement du journal d'audit est encapsulé dans un enregistrement au format JSON. Cet enregistrement contient les champs suivants.

type

Ce champ a toujours la valeur `DatabaseActivityMonitoringRecords`.

version ;

Ce champ représente la version du contrat ou du protocole de données de flux d'activité de base de données. Il définit les champs disponibles.

La version 1.0 représente la prise en charge des flux d'activité de données d'origine pour Aurora PostgreSQL versions 10.7 et 11.4. La version 1.1 représente la prise en charge des flux d'activité de données pour Aurora PostgreSQL versions 10.10 et ultérieures et Aurora PostgreSQL version 11.5 et ultérieures. La version 1.1 inclut les champs supplémentaires `errorMessage` et `startTime`. La version 1.2 représente la prise en charge des flux d'activité de données pour Aurora MySQL version 2.08 et ultérieures. La version 1.2 inclut les champs supplémentaires `endTime` et `transactionId`.

databaseActivityEvents

Chaîne chiffrée représentant un ou plusieurs événements d'activité. Elle est représentée sou la forme d'un tableau base64 octets. Lorsque vous déchiffrez la chaîne, le résultat est un enregistrement au format JSON avec des champs comme ceux des exemples de cette section.

key

Clé de données chiffrée utilisée pour chiffrer la chaîne `databaseActivityEvents`. Il s'agit de la même clé AWS KMS key que celle que vous avez fournie lorsque vous avez démarré le flux d'activité de base de données.

L'exemple suivant illustre le format de cet enregistrement.

```
{
  "type":"DatabaseActivityMonitoringRecords",
  "version":"1.1",
  "databaseActivityEvents":"encrypted audit records",
  "key":"encrypted key"
}
```

Pour déchiffrer le contenu du champ `databaseActivityEvents`, procédez comme suit :

1. Déchiffrez la valeur dans le champ JSON `key` à l'aide de la clé KMS que vous avez fournie lors du démarrage du flux d'activité de base de données. Cette opération renvoie la clé de chiffrement des données en texte clair.
2. Décodez en base64 la valeur dans le champ JSON `databaseActivityEvents` pour obtenir le texte chiffré, au format binaire, de la charge utile d'audit.
3. Déchiffrez le chiffrement binaire avec la clé de chiffrement de données que vous avez décodée au cours de la première étape.
4. Décompressez la charge utile déchiffrée.

- La charge utile chiffrée se trouve dans le champ `databaseActivityEvents`.
- Le champ `databaseActivityEventList` contient un tableau d'enregistrements d'audits. Les champs `type` du tableau peuvent être `record` ou `heartbeat`.

L'enregistrement d'événement d'activité du journal d'audit est un objet JSON qui contient les informations suivantes.

Champ JSON	Type de données	Description
`type`	chaîne	Type de l'enregistrement JSON. La valeur est `DatabaseActivityMonitoringRecord` .
`clusterId`	chaîne	Identifiant de ressource de cluster de base de données. Il correspond à l'attribut de cluster de bases de données`DbClusterResourceId` .
`instanceId`	chaîne	Identifiant de ressource d'instance de base de données. Il correspond à l'attribut d'instance de base de données `DbiResourceId` .
Tableau JSON databaseActivityEventList	chaîne	Tableau d'enregistrements d'audits d'activité ou de messages de pulsations.

Tableau JSON databaseActivityEventList pour les flux d'activité de base de données

Les données utiles du journal d'audit sont un tableau JSON `databaseActivityEventList` chiffré. Ci-dessous, les tableaux répertorient par ordre alphabétique les champs de chaque événement d'activité dans le tableau `DatabaseActivityEventList` déchiffré d'un journal d'audit. Les champs diffèrent selon que vous utilisez Aurora PostgreSQL ou Aurora MySQL. Consultez la table qui s'applique à votre moteur de base de données.

> **Important**
>
> Il se peut que la structure d'événement change. Il se peut qu'Aurora ajoute de nouveaux champs aux événements d'activité à l'avenir. Dans les applications qui analysent les données

JSON, assurez-vous que votre code peut ignorer ou prendre les mesures appropriées pour les noms de champs inconnus.

Champs databaseActivityEventList pour Aurora PostgreSQL

Voici les champs `databaseActivityEventList` pour Aurora PostgreSQL.

Champ	Type de données	Description
`class`	chaîne	La classe d'un événement d'activité. Les valeurs possibles pour Aurora PostgreSQL sont les suivantes : • `ALL` • `CONNECT` – Événement de connexion ou déconnexion. • `DDL` – Instruction DDL non incluse dans la liste des instructions pour la classe `ROLE`. • `FUNCTION` – Appel de fonction ou un bloc `DO`. • `MISC` – Commande diverse telle que `DISCARD`, `FETCH`, `CHECKPOINT` ou `VACUUM`. • `NONE` • `READ` – Instruction `SELECT` ou `COPY` lorsque la source est une relation ou une requête. • `ROLE` – Instruction liée aux rôles et aux privilèges, incluant `GRANT`, `REVOKE` et `CREATE/ALTER/DROP ROLE`. • `WRITE` – Instruction `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` ou `COPY` lorsque la destination est une relation.
`clientApplication`	chaîne	Application utilisée par le client pour se connecter, telle que signalée par le client. Le client n'a pas à fournir cette information, la valeur peut être « null ».
`command`	chaîne	Nom de la commande SQL sans aucun détail sur la commande

Champ	Type de données	Description
`commandText`	chaîne	Instruction SQL réelle transmise par l'utilisateur. Pour Aurora PostgreSQL, la valeur est identique à l'instruction SQL d'origine. Ce champ est utilisé pour tous les types d'enregis trements, excepté pour les enregistrements de connexion ou de déconnexion, auxquels cas la valeur est « null ». **Important** Le texte SQL complet de chaque instruction est visible dans le journal d'audit du flux d'activité, y compris les données sensibles. Cependant, les mots de passe des utilisateurs de la base de données sont masqués si Aurora peut les deviner suivant le contexte, comme le montre l'exemple suivant. `ALTER ROLE role-name WITH password`
`databaseName`	chaîne	Base de données à laquelle l'utilisateur s'est connecté.
`dbProtocol`	chaîne	Le protocole de base de données, par exemple `Postgres 3.0`.
`dbUserName`	chaîne	L'utilisateur de la base de données avec lequel le client s'est authentifié.

Champ	Type de données	Description
`errorMessage` (enregistrements d'activité de base de données version 1.1 uniquement)	chaîne	En cas d'erreur, ce champ contient le message d'erreur qui aurait été généré par le serveur de base de données. La valeur `errorMessage` est nulle pour les instructions normales qui n'ont pas donné lieu à une erreur. Une erreur est définie comme étant une activité quelconque qui produirait un événement de journal d'erreurs PostgreSQL visible par le client avec un niveau de gravité égal ou supérieur à `ERROR`. Pour plus d'informations, consultez Niveaux de gravité des messages PostgreSQL. Par exemple, les erreurs de syntaxe et les annulations de requête génèrent un message d'erreur. Les erreurs internes du serveur PostgreSQL, telles que les erreurs de processus du pointeur de contrôle en arrière-plan, ne génèrent pas de message d'erreur. Cependant, des enregistrements sont toujours émis pour des événements de ce type, quel que soit le paramètre du niveau de gravité du journal. Cela empêche les pirates informatiques de désactiver la journalisation pour tenter d'éviter la détection. Consultez également le champ `exitCode`.
`exitCode`	int	Valeur utilisée pour l'enregistrement en sortie de session. En cas de sortie sans problème, elle contient le code de sortie. Un code de sortie ne peut pas toujours être obtenu dans certains scénarios d'échec. Par exemple, si PostgreSQL effectue un `exit()` ou si un opérateur exécute une commande telle que `kill -9`. Si une erreur s'est produite, le champ `exitCode` affiche le code d'erreur SQL `SQLSTATE`, comme indiqué dans les codes d'erreur PostgreSQL. Consultez également le champ `errorMessage` .

Champ	Type de données	Description
`logTime`	chaîne	Horodatage, tel qu'il est enregistré dans l'audit du chemin du code. Cela représente l'heure de fin d'exécution de l'instruction SQL. Consultez également le champ `startTime` .
`netProtocol`	chaîne	Protocole de communication réseau.
`objectName`	chaîne	Nom de l'objet de la base de données si l'instruction SQL agit sur l'un d'eux. Ce champ n'est utilisé que lorsque l'instruction SQL agit sur un objet de base de données. Si l'instruction SQL n'agit pas sur un objet, la valeur est « null ».
`objectType`	chaîne	Type de l'objet de base de données, par exemple, table, index, vue, etc. Ce champ n'est utilisé que lorsque l'instruction SQL agit sur un objet de base de données. Si l'instruction SQL n'agit pas sur un objet, la valeur est « null ». Les valeurs valides sont notamment les suivantes : • `COMPOSITE TYPE` • `FOREIGN TABLE` • `FUNCTION` • `INDEX` • `MATERIALIZED VIEW` • `SEQUENCE` • `TABLE` • `TOAST TABLE` • `VIEW` • `UNKNOWN`
`paramList`	chaîne	Tableau de paramètres séparés par des virgules, transmis à l'instruction SQL. Si l'instruction SQL n'a pas de paramètres, la valeur est un tableau vide.

Champ	Type de données	Description
`pid`	int	ID du processus de backend qui est dédié au service de la connexion du client.
`remoteHost`	chaîne	L'adresse IP du client ou le nom d'hôte. Pour Aurora PostgreSQL, le paramètre `log_hostname` de la base de données détermine lequel est utilisé. La valeur `remoteHost` inclut également `[local]` et `localhost` qui indiquent l'activité de l'utilisateur `rdsadmin`.
`remotePort`	chaîne	Numéro de port du client.
`rowCount`	int	Numéro des lignes de la table affectées ou récupérées par l'instruction SQL. Ce champ n'est utilisé que pour les instructions SQL qui sont des instructions en langage de manipulation de données (DML). Si l'instruction SQL n'est pas une instruction DML, la valeur est « null ».
`serverHost`	chaîne	Adresse IP de l'hôte du serveur de base de données. La valeur `serverHost` inclut également `[local]` et `localhost` qui indiquent l'activité de l'utilisateur `rdsadmin`.
`serverType`	chaîne	Type du serveur de base de données, par exemple `PostgreSQL` .
`serverVersion`	chaîne	Version du serveur de base de données, par exemple `2.3.1` pour Aurora PostgreSQL.
`serviceName`	chaîne	Nom du service, par exemple `Amazon Aurora PostgreSQL-Compatible edition` .
`sessionId`	int	Identifiant de session à pseudo unique.
`sessionId`	int	Identifiant de session à pseudo unique.

Champ	Type de données	Description
`startTime` (enregistrements d'activité de base de données version 1.1 uniquement)	chaîne	Heure à laquelle l'exécution a commencé pour l'instruction SQL. Pour calculer le temps d'exécution approximatif de l'instruction SQL, utilisez `logTime - startTime` . Consultez également le champ `logTime`.
`statementId`	int	Identifiant de l'instruction SQL du client. Le compteur se trouve au niveau de la session et s'incrémente avec chaque instruction SQL entrée par le client.
`substatementId`	int	Identifiant d'une sous-instruction SQL. Cette valeur compte le nombre de sous-instructions pour chaque instruction identifiée par le champ `statementId` .
`type`	chaîne	Type d'événement. Les valeurs valides sont `record` ou `heartbeat` .

Champs databaseActivityEventList pour Aurora MySQL

Voici les champs `databaseActivityEventList` pour Aurora MySQL.

Champ	Type de données	Description
`class`	chaîne	La classe d'un événement d'activité. Les valeurs possibles pour Aurora MySQL sont les suivantes : • `MAIN` – Événement principal représentant une instruction SQL. • `AUX` – Événement supplémentaire contenant des détails supplémentaires. Par exemple, une instruction qui

Champ	Type de données	Description
		renomme un objet peut avoir un événement d'une classe AUX qui reflète le nouveau nom. Pour rechercher les événements MAIN et AUX correspon dant à la même instruction, vérifiez les événements différents qui ont les mêmes valeurs pour le champ pid et pour le champ statementId .
clientApp lication	chaîne	Application utilisée par le client pour se connecter, telle que signalée par le client. Le client n'a pas à fournir cette informati on, la valeur peut être « null ».

Champ	Type de données	Description
`command`	chaîne	Catégorie générale de l'instruction SQL. Les valeurs de ce champ dépendent de la valeur de `class`. Lorsque `class` est `MAIN` les valeurs sont notamment les suivantes : • `CONNECT` – Lorsqu'une session client est connectée. • `QUERY` – Instruction SQL. Accompagnée d'un ou plusieurs événements dont la valeur `class` est `AUX`. • `DISCONNECT` – Lorsqu' une session client est déconnect ée. • `FAILED_CONNECT` – Lorsqu'un client tente de se connecter mais n'y parvient pas. • `CHANGEUSER` – Changement d'état qui fait partie du protocole réseau MySQL, et non d'une instruction que vous émettez. Lorsque `class` est `AUX` les valeurs sont notamment les suivantes : • `READ` – Instruction `SELECT` ou `COPY` lorsque la source est une relation ou une requête. • `WRITE` – Instruction `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` ou `COPY` lorsque la destination est une relation. • `DROP` – Suppression d'un objet. • `CREATE` – Création d'un objet. • `RENAME` – Renommage d'un objet. • `ALTER` – Pour changer les propriétés d'un objet.

Champ	Type de données	Description
`commandText`	chaîne	Pour les événements dont la valeur `class` est `MAIN`, ce champ représente l'instruction SQL réelle transmise par l'utilisateur. Ce champ est utilisé pour tous les types d'enregistrements, excepté pour les enregistrements de connexion ou de déconnexion, auxquels cas la valeur est « null ». Pour les événements dont la valeur `class` est `AUX`, ce champ contient des informations supplémentaires sur les objets impliqués dans l'événement. Pour Aurora MySQL, les caractères tels que les guillemets sont précédés d'une barre oblique inverse, représentant un caractère d'échappement. **Important** Le texte SQL complet de chaque instruction est visible dans le journal d'audit, y compris les données sensibles. Cependant, les mots de passe des utilisateurs de la base de données sont masqués si Aurora peut les deviner suivant le contexte, comme le montre l'exemple suivant. `mysql> SET PASSWORD = 'my-password ';` **Note** Spécifiez un mot de passe autre que celui indiqué ici, en tant que bonne pratique de sécurité.
`databaseName`	chaîne	Base de données à laquelle l'utilisateur s'est connecté.

Champ	Type de données	Description
`dbProtocol`	chaîne	Protocole de la base de données. Actuellement, cette valeur est toujours `MySQL` pour Aurora MySQL.
`dbUserName`	chaîne	L'utilisateur de la base de données avec lequel le client s'est authentifié.
`endTime` (enregistrements d'activité de base de données version 1.2 uniquement)	chaîne	Heure à laquelle l'exécution a fini pour l'instruction SQL. Il est représenté au format UTC (temps universel coordonné). Pour calculer le temps d'exécution de l'instruction SQL, utilisez `endTime - startTime` . Consultez également le champ `startTime` .

Champ	Type de données	Description
`errorMessage` (enregistrements d'activité de base de données version 1.1 uniquement)	chaîne	En cas d'erreur, ce champ contient le message d'erreur qui aurait été généré par le serveur de base de données. La valeur `errorMessage` est nulle pour les instructions normales qui n'ont pas donné lieu à une erreur. Une erreur est définie comme étant une activité quelconque qui produirait un événement de journal d'erreurs MySQL visible par le client avec un niveau de gravité égal ou supérieur à `ERROR`. Pour plus d'informations, consultez Le journal d'erreurs dans le Manuel de référence MySQL. Par exemple, les erreurs de syntaxe et les annulations de requête génèrent un message d'erreur. Les erreurs internes du serveur MySQL, telles que les erreurs de processus du pointeur de contrôle en arrière-plan, ne génèrent pas de message d'erreur. Cependant, des enregistrements sont toujours émis pour des événements de ce type, quel que soit le paramètre du niveau de gravité du journal. Cela empêche les pirates informatiques de désactiver la journalisation pour tenter d'éviter la détection. Consultez également le champ `exitCode`.
`exitCode`	int	Valeur utilisée pour l'enregistrement en sortie de session. En cas de sortie sans problème, elle contient le code de sortie. Un code de sortie ne peut pas toujours être obtenu dans certains scénarios d'échec. Dans de tels cas, cette valeur peut être nulle ou vide.
`logTime`	chaîne	Horodatage, tel qu'il est enregistré dans l'audit du chemin du code. Il est représenté au format UTC (temps universel coordonné). Pour connaître la méthode la plus précise de calculer la durée de l'instruction, consultez les champs `startTime` et `endTime`.

Champ	Type de données	Description
`netProtocol`	chaîne	Protocole de communication réseau. Actuellement, cette valeur est toujours `TCP` pour Aurora MySQL.
`objectName`	chaîne	Nom de l'objet de la base de données si l'instruction SQL agit sur l'un d'eux. Ce champ n'est utilisé que lorsque l'instruction SQL agit sur un objet de base de données. Si l'instruction SQL n'agit pas sur un objet, cette valeur est vide. Pour construire le nom complet de l'objet, combinez `databaseName` et `objectName`. Si la requête comprend plusieurs objets, ce champ peut être une liste de noms séparés par des virgules.
`objectType`	chaîne	Type de l'objet de base de données, par exemple, table, index, etc. Ce champ n'est utilisé que lorsque l'instruction SQL agit sur un objet de base de données. Si l'instruction SQL n'agit pas sur un objet, la valeur est « null ». Les valeurs valides pour Aurora MySQL sont notamment les suivantes : • `INDEX` • `TABLE` • `UNKNOWN`
`paramList`	chaîne	Ce champ n'est pas utilisé pour Aurora MySQL et est toujours « null ».
`pid`	int	ID du processus de backend qui est dédié au service de la connexion du client. Lorsque le serveur de base de données est redémarré, les modifications `pid` et le compteur du champ `statementId` redémarrent.

Champ	Type de données	Description
`remoteHost`	chaîne	L'adresse IP ou le nom d'hôte du client qui a émis l'instruction SQL. Pour Aurora MySQL, le paramètre `skip_name_resolve` de la base de données détermine lequel est utilisé. La valeur `localhost` indique l'activité de l'utilisateur spécial `rdsadmin`.
`remotePort`	chaîne	Numéro de port du client.
`rowCount`	int	Nombre de lignes renvoyées par l'instruction SQL. Par exemple, si une instruction SELECT renvoie 10 lignes, rowCount est égal à 10. Pour les instructions INSERT ou UPDATE, rowCount est égal 0.
`serverHost`	chaîne	Identifiant d'instance du serveur de base de données.
`serverType`	chaîne	Type du serveur de base de données, par exemple `MySQL`.
`serverVersion`	chaîne	Version du serveur de base de données. Actuellement, cette valeur est toujours `MySQL 5.7.12` pour Aurora MySQL.
`serviceName`	chaîne	Nom du service. Actuellement, cette valeur est toujours `Amazon Aurora MySQL` pour Aurora MySQL.
`sessionId`	int	Identifiant de session à pseudo unique.
`startTime` (enregistrements d'activité de base de données version 1.1 uniquement)	chaîne	Heure à laquelle l'exécution a commencé pour l'instruction SQL. Il est représenté au format UTC (temps universel coordonné). Pour calculer le temps d'exécution de l'instruction SQL, utilisez `endTime - startTime`. Consultez également le champ `endTime`.

Champ	Type de données	Description
`statementId`	int	Identifiant de l'instruction SQL du client. Le compteur s'incrémente avec chaque instruction SQL saisie par le client. Le compteur est réinitialisé lorsque l'instance de base de données est redémarrée.
`substatementId`	int	Identifiant d'une sous-instruction SQL. Cette valeur est 1 pour les événements de classe `MAIN` et 2 pour les événements de classe `AUX`. Utilisez le champ `statementId` pour identifier tous les événements générés par la même instruction.
`transactionId` (enregistrements d'activité de base de données version 1.2 uniquement)	int	Identifiant d'une transaction.
`type`	chaîne	Type d'événement. Les valeurs valides sont `record` ou `heartbeat` .

Traitement d'un flux d'activité de base de données à l'aide du kit AWS SDK

Vous pouvez traiter par programmation un flux d'activité à l'aide du kit AWS SDK. Les exemples suivants sont des exemples Java et Python entièrement fonctionnels de traitement du flux de données Kinesis.

Java

```
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.nio.ByteBuffer;
import java.nio.charset.StandardCharsets;
```

```
import java.security.NoSuchAlgorithmException;
import java.security.NoSuchProviderException;
import java.security.Security;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import java.util.zip.GZIPInputStream;

import javax.crypto.Cipher;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.spec.SecretKeySpec;

import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.encryptionsdk.AwsCrypto;
import com.amazonaws.encryptionsdk.CryptoInputStream;
import com.amazonaws.encryptionsdk.jce.JceMasterKey;
import
 com.amazonaws.services.kinesis.clientlibrary.exceptions.InvalidStateException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ShutdownException;
import com.amazonaws.services.kinesis.clientlibrary.exceptions.ThrottlingException;
import com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessor;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorCheckpointer;
import
 com.amazonaws.services.kinesis.clientlibrary.interfaces.IRecordProcessorFactory;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.InitialPositionInStream;
import
 com.amazonaws.services.kinesis.clientlibrary.lib.worker.KinesisClientLibConfiguration;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.ShutdownReason;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker;
import com.amazonaws.services.kinesis.clientlibrary.lib.worker.Worker.Builder;
import com.amazonaws.services.kinesis.model.Record;
import com.amazonaws.services.kms.AWSKMS;
import com.amazonaws.services.kms.AWSKMSClientBuilder;
import com.amazonaws.services.kms.model.DecryptRequest;
import com.amazonaws.services.kms.model.DecryptResult;
import com.amazonaws.util.Base64;
import com.amazonaws.util.IOUtils;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.annotations.SerializedName;
```

```
import org.bouncycastle.jce.provider.BouncyCastleProvider;

public class DemoConsumer {

    private static final String STREAM_NAME = "aws-rds-das-[cluster-external-
resource-id]";
    private static final String APPLICATION_NAME = "AnyApplication"; //unique
 application name for dynamo table generation that holds kinesis shard tracking
    private static final String AWS_ACCESS_KEY =
 "[AWS_ACCESS_KEY_TO_ACCESS_KINESIS]";
    private static final String AWS_SECRET_KEY =
 "[AWS_SECRET_KEY_TO_ACCESS_KINESIS]";
    private static final String DBC_RESOURCE_ID = "[cluster-external-resource-id]";
    private static final String REGION_NAME = "[region-name]"; //us-east-1, us-
east-2...
    private static final BasicAWSCredentials CREDENTIALS = new
 BasicAWSCredentials(AWS_ACCESS_KEY, AWS_SECRET_KEY);
    private static final AWSStaticCredentialsProvider CREDENTIALS_PROVIDER = new
 AWSStaticCredentialsProvider(CREDENTIALS);

    private static final AwsCrypto CRYPTO = new AwsCrypto();
    private static final AWSKMS KMS = AWSKMSClientBuilder.standard()
            .withRegion(REGION_NAME)
            .withCredentials(CREDENTIALS_PROVIDER).build();

    class Activity {
        String type;
        String version;
        String databaseActivityEvents;
        String key;
    }

    class ActivityEvent {
        @SerializedName("class") String _class;
        String clientApplication;
        String command;
        String commandText;
        String databaseName;
        String dbProtocol;
        String dbUserName;
        String endTime;
        String errorMessage;
        String exitCode;
        String logTime;
```

```
        String netProtocol;
        String objectName;
        String objectType;
        List<String> paramList;
        String pid;
        String remoteHost;
        String remotePort;
        String rowCount;
        String serverHost;
        String serverType;
        String serverVersion;
        String serviceName;
        String sessionId;
        String startTime;
        String statementId;
        String substatementId;
        String transactionId;
        String type;
    }

    class ActivityRecords {
        String type;
        String clusterId;
        String instanceId;
        List<ActivityEvent> databaseActivityEventList;
    }

    static class RecordProcessorFactory implements IRecordProcessorFactory {
        @Override
        public IRecordProcessor createProcessor() {
            return new RecordProcessor();
        }
    }

    static class RecordProcessor implements IRecordProcessor {

        private static final long BACKOFF_TIME_IN_MILLIS = 3000L;
        private static final int PROCESSING_RETRIES_MAX = 10;
        private static final long CHECKPOINT_INTERVAL_MILLIS = 60000L;
        private static final Gson GSON = new
GsonBuilder().serializeNulls().create();

        private static final Cipher CIPHER;
        static {
```

```
            Security.insertProviderAt(new BouncyCastleProvider(), 1);
            try {
                CIPHER = Cipher.getInstance("AES/GCM/NoPadding", "BC");
            } catch (NoSuchAlgorithmException | NoSuchPaddingException |
NoSuchProviderException e) {
                throw new ExceptionInInitializerError(e);
            }
        }

        private long nextCheckpointTimeInMillis;

        @Override
        public void initialize(String shardId) {
        }

        @Override
        public void processRecords(final List<Record> records, final
IRecordProcessorCheckpointer checkpointer) {
            for (final Record record : records) {
                processSingleBlob(record.getData());
            }

            if (System.currentTimeMillis() > nextCheckpointTimeInMillis) {
                checkpoint(checkpointer);
                nextCheckpointTimeInMillis = System.currentTimeMillis() +
CHECKPOINT_INTERVAL_MILLIS;
            }
        }

        @Override
        public void shutdown(IRecordProcessorCheckpointer checkpointer,
ShutdownReason reason) {
            if (reason == ShutdownReason.TERMINATE) {
                checkpoint(checkpointer);
            }
        }

        private void processSingleBlob(final ByteBuffer bytes) {
            try {
                // JSON $Activity
                final Activity activity = GSON.fromJson(new String(bytes.array(),
StandardCharsets.UTF_8), Activity.class);

                // Base64.Decode
```

```
                final byte[] decoded =
Base64.decode(activity.databaseActivityEvents);
                final byte[] decodedDataKey = Base64.decode(activity.key);

                Map<String, String> context = new HashMap<>();
                context.put("aws:rds:dbc-id", DBC_RESOURCE_ID);

                // Decrypt
                final DecryptRequest decryptRequest = new DecryptRequest()

.withCiphertextBlob(ByteBuffer.wrap(decodedDataKey)).withEncryptionContext(context);
                final DecryptResult decryptResult = KMS.decrypt(decryptRequest);
                final byte[] decrypted = decrypt(decoded,
getByteArray(decryptResult.getPlaintext()));

                // GZip Decompress
                final byte[] decompressed = decompress(decrypted);
                // JSON $ActivityRecords
                final ActivityRecords activityRecords = GSON.fromJson(new
String(decompressed, StandardCharsets.UTF_8), ActivityRecords.class);

                // Iterate throught $ActivityEvents
                for (final ActivityEvent event :
activityRecords.databaseActivityEventList) {
                    System.out.println(GSON.toJson(event));
                }
            } catch (Exception e) {
                // Handle error.
                e.printStackTrace();
            }
        }

        private static byte[] decompress(final byte[] src) throws IOException {
            ByteArrayInputStream byteArrayInputStream = new
ByteArrayInputStream(src);
            GZIPInputStream gzipInputStream = new
GZIPInputStream(byteArrayInputStream);
            return IOUtils.toByteArray(gzipInputStream);
        }

        private void checkpoint(IRecordProcessorCheckpointer checkpointer) {
            for (int i = 0; i < PROCESSING_RETRIES_MAX; i++) {
                try {
                    checkpointer.checkpoint();
```

```
                    break;
                } catch (ShutdownException se) {
                    // Ignore checkpoint if the processor instance has been shutdown
(fail over).
                    System.out.println("Caught shutdown exception, skipping
checkpoint." + se);
                    break;
                } catch (ThrottlingException e) {
                    // Backoff and re-attempt checkpoint upon transient failures
                    if (i >= (PROCESSING_RETRIES_MAX - 1)) {
                        System.out.println("Checkpoint failed after " + (i + 1) +
"attempts." + e);
                        break;
                    } else {
                        System.out.println("Transient issue when checkpointing -
attempt " + (i + 1) + " of " + PROCESSING_RETRIES_MAX + e);
                    }
                } catch (InvalidStateException e) {
                    // This indicates an issue with the DynamoDB table (check for
table, provisioned IOPS).
                    System.out.println("Cannot save checkpoint to the DynamoDB table
used by the Amazon Kinesis Client Library." + e);
                    break;
                }
                try {
                    Thread.sleep(BACKOFF_TIME_IN_MILLIS);
                } catch (InterruptedException e) {
                    System.out.println("Interrupted sleep" + e);
                }
            }
        }
    }

    private static byte[] decrypt(final byte[] decoded, final byte[] decodedDataKey)
throws IOException {
        // Create a JCE master key provider using the random key and an AES-GCM
encryption algorithm
        final JceMasterKey masterKey = JceMasterKey.getInstance(new
SecretKeySpec(decodedDataKey, "AES"),
                "BC", "DataKey", "AES/GCM/NoPadding");
        try (final CryptoInputStream<JceMasterKey> decryptingStream =
CRYPTO.createDecryptingStream(masterKey, new ByteArrayInputStream(decoded));
             final ByteArrayOutputStream out = new ByteArrayOutputStream()) {
            IOUtils.copy(decryptingStream, out);
```

```
            return out.toByteArray();
        }
    }

    public static void main(String[] args) throws Exception {
        final String workerId = InetAddress.getLocalHost().getCanonicalHostName() +
 ":" + UUID.randomUUID();
        final KinesisClientLibConfiguration kinesisClientLibConfiguration =
                new KinesisClientLibConfiguration(APPLICATION_NAME, STREAM_NAME,
 CREDENTIALS_PROVIDER, workerId);

 kinesisClientLibConfiguration.withInitialPositionInStream(InitialPositionInStream.LATEST);
        kinesisClientLibConfiguration.withRegionName(REGION_NAME);
        final Worker worker = new Builder()
                .recordProcessorFactory(new RecordProcessorFactory())
                .config(kinesisClientLibConfiguration)
                .build();

        System.out.printf("Running %s to process stream %s as worker %s...\n",
 APPLICATION_NAME, STREAM_NAME, workerId);

        try {
            worker.run();
        } catch (Throwable t) {
            System.err.println("Caught throwable while processing data.");
            t.printStackTrace();
            System.exit(1);
        }
        System.exit(0);
    }

    private static byte[] getByteArray(final ByteBuffer b) {
        byte[] byteArray = new byte[b.remaining()];
        b.get(byteArray);
        return byteArray;
    }
}
```

Python

```
import base64
import json
import zlib
```

```
import aws_encryption_sdk
from aws_encryption_sdk import CommitmentPolicy
from aws_encryption_sdk.internal.crypto import WrappingKey
from aws_encryption_sdk.key_providers.raw import RawMasterKeyProvider
from aws_encryption_sdk.identifiers import WrappingAlgorithm, EncryptionKeyType
import boto3

REGION_NAME = '<region>'                    # us-east-1
RESOURCE_ID = '<external-resource-id>'      # cluster-ABCD123456
STREAM_NAME = 'aws-rds-das-' + RESOURCE_ID  # aws-rds-das-cluster-ABCD123456

enc_client =
 aws_encryption_sdk.EncryptionSDKClient(commitment_policy=CommitmentPolicy.FORBID_ENCRYPT_AL

class MyRawMasterKeyProvider(RawMasterKeyProvider):
    provider_id = "BC"

    def __new__(cls, *args, **kwargs):
        obj = super(RawMasterKeyProvider, cls).__new__(cls)
        return obj

    def __init__(self, plain_key):
        RawMasterKeyProvider.__init__(self)
        self.wrapping_key =
 WrappingKey(wrapping_algorithm=WrappingAlgorithm.AES_256_GCM_IV12_TAG16_NO_PADDING,
                                        wrapping_key=plain_key,
 wrapping_key_type=EncryptionKeyType.SYMMETRIC)

    def _get_raw_key(self, key_id):
        return self.wrapping_key


def decrypt_payload(payload, data_key):
    my_key_provider = MyRawMasterKeyProvider(data_key)
    my_key_provider.add_master_key("DataKey")
    decrypted_plaintext, header = enc_client.decrypt(
        source=payload,

 materials_manager=aws_encryption_sdk.materials_managers.default.DefaultCryptoMaterialsManag
    return decrypted_plaintext


def decrypt_decompress(payload, key):
    decrypted = decrypt_payload(payload, key)
```

```
    return zlib.decompress(decrypted, zlib.MAX_WBITS + 16)


def main():
    session = boto3.session.Session()
    kms = session.client('kms', region_name=REGION_NAME)
    kinesis = session.client('kinesis', region_name=REGION_NAME)

    response = kinesis.describe_stream(StreamName=STREAM_NAME)
    shard_iters = []
    for shard in response['StreamDescription']['Shards']:
        shard_iter_response = kinesis.get_shard_iterator(StreamName=STREAM_NAME,
 ShardId=shard['ShardId'],

 ShardIteratorType='LATEST')
        shard_iters.append(shard_iter_response['ShardIterator'])

    while len(shard_iters) > 0:
        next_shard_iters = []
        for shard_iter in shard_iters:
            response = kinesis.get_records(ShardIterator=shard_iter, Limit=10000)
            for record in response['Records']:
                record_data = record['Data']
                record_data = json.loads(record_data)
                payload_decoded =
 base64.b64decode(record_data['databaseActivityEvents'])
                data_key_decoded = base64.b64decode(record_data['key'])
                data_key_decrypt_result =
 kms.decrypt(CiphertextBlob=data_key_decoded,

 EncryptionContext={'aws:rds:dbc-id': RESOURCE_ID})
                print (decrypt_decompress(payload_decoded,
 data_key_decrypt_result['Plaintext']))
            if 'NextShardIterator' in response:
                next_shard_iters.append(response['NextShardIterator'])
        shard_iters = next_shard_iters


if __name__ == '__main__':
    main()
```

Exemples de politique IAM pour les flux d’activité de base de données

Tout utilisateur disposant des privilèges de rôle Gestion des identités et des accès AWS (IAM) appropriés pour les flux d'activité de base de données peut créer, démarrer, arrêter et modifier les paramètres des flux d'activité pour une de base de données. Ces actions sont consignées dans le journal d’audit du flux. Pour des raisons de conformité optimales, nous vous recommandons de ne pas octroyer ces privilèges à DBAs.

Vous devrez paramétrer les accès aux flux d’activité de base de données à l’aide de politiques IAM. Pour plus d’informations sur l’authentification Aurora, consultez Identity and Access Management pour Amazon Aurora. Pour plus d’informations sur la création des stratégies IAM, consultez Création et utilisation d'une politique IAM pour l'accès à une base de données IAM.

Example politique pour autoriser la configuration de Database Activity Streams

Pour donner aux utilisateurs des accès précis en vue de modifier les flux d’activité, utilisez les clés de contexte d’opération spécifiques au service `rds:StartActivityStream` et `rds:StopActivityStream` dans une stratégie IAM. L’exemple de politique IAM suivant autorise un utilisateur ou un rôle à configurer des flux d’activité.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ConfigureActivityStreams",
            "Effect": "Allow",
            "Action": [
                "rds:StartActivityStream",
                "rds:StopActivityStream"
            ],
            "Resource": "*"
        }
    ]
}
```

Example politique pour autoriser le démarrage de Database Activity Streams

L’exemple de politique IAM suivant autorise un utilisateur ou un rôle à démarrer des flux d’activité.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStartActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
    ]
}
```

Example politique pour autoriser l’arrêt de Database Activity Streams

L’exemple de politique IAM suivant autorise un utilisateur ou un rôle à arrêter des flux d’activité.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowStopActivityStreams",
            "Effect":"Allow",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example politique pour refuser le démarrage de Database Activity Streams

La politique IAM suivante empêche un utilisateur ou un rôle de démarrer des flux d’activité.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStartActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StartActivityStream",
            "Resource":"*"
        }
     ]
}
```

Example politique pour refuser l’arrêt de Database Activity Streams

La politique IAM suivante empêche un utilisateur ou un rôle d’arrêter des flux d’activité.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"DenyStopActivityStreams",
            "Effect":"Deny",
            "Action":"rds:StopActivityStream",
            "Resource":"*"
        }
    ]
}
```

Surveillance des menaces avec Amazon GuardDuty RDS Protection pour Amazon Aurora

Amazon GuardDuty est un service de détection des menaces qui vous aide à protéger vos comptes, vos conteneurs, vos charges de travail et les données de votre environnement AWS. Grâce à des modèles de machine learning (ML) et à des fonctions de détection des anomalies et des menaces, GuardDuty surveille en continu diverses sources de journaux et les activités d'exécution afin d'identifier et de hiérarchiser les risques de sécurité potentiels ainsi que les activités malveillantes dans votre environnement.

GuardDuty RDS Protection analyse et établit un profil pour les événements de connexion afin de détecter les menaces d'accès potentielles à vos bases de données Amazon Aurora. Lorsque vous activez la protection RDS, GuardDuty consomme les événements de connexion RDS de vos bases de données Aurora. RDS Protection surveille ces événements et en établit le profil pour détecter les menaces potentielles de l'intérieur ou des acteurs externes.

Pour plus d'informations sur l'activation de GuardDuty RDS Protection, consultez GuardDuty RDS Protection dans le Guide de l'utilisateur Amazon GuardDuty.

Lorsque RDS Protection détecte une menace potentielle, telle qu'un modèle inhabituel de tentatives de connexion réussies ou échouées, GuardDuty génère une nouvelle constatation avec des détails sur la base de données potentiellement compromise. Vous pouvez consulter les détails des résultats dans la section récapitulative des résultats de la console Amazon GuardDuty. Les détails des résultats varient en fonction du type de découverte. Les principaux détails, le type de ressource et le rôle de la ressource, déterminent le type d'informations disponibles pour toute recherche. Pour plus d'informations sur les informations couramment disponibles sur les résultats et les types de résultats, consultez Finding details (Détails des résultats) et GuardDuty RDS Protection finding types (Types de recherche GuardDuty RDS Protection), respectivement, dans le Guide de l'utilisateur Amazon GuardDuty.

Vous pouvez activer ou désactiver la fonction de protection RDS dans tous les Compte AWS de n'importe quelle Région AWS où cette fonction est disponible. Lorsque la protection RDS n'est pas activée, GuardDuty ne détecte pas les bases de données Aurora potentiellement compromises et ne fournit pas de détails sur la compromission.

Un compte GuardDuty existant peut activer RDS Protection avec une période d'essai de 30 jours. Pour un nouveau compte GuardDuty, RDS Protection est déjà activée et incluse dans la période

d’essai gratuite de 30 jours. Pour plus d’informations, consultez Estimating GuardDuty cost (Estimation des coûts GuardDuty) dans le Guide de l’utilisateur Amazon GuardDuty.

Pour plus d’informations sur les Régions AWS où GuardDuty ne prend pas encore en charge RDS Protection, consultez Disponibilité des fonctionnalités spécifiques à la région dans le Guide de l’utilisateur Amazon GuardDuty.

Pour plus d’informations sur les versions de base de données Aurora prises en charge par GuardDuty RDS Protection, consultez Bases de données Amazon Aurora, Amazon RDS et Aurora Limitless prises en charge dans le guide de l’utilisateur Amazon GuardDuty.

Utilisation de Amazon Aurora MySQL

Amazon Aurora est un moteur de base de données relationnelle entièrement gérée compatible avec MySQL, qui associe la vitesse et la disponibilité des bases de données commerciales haut de gamme à la simplicité et à la rentabilité des bases de données open source. Aurora MySQL est une solution alternative pour MySQL, qui vous permet de configurer, gérer et mettre à l'échelle de façon simple et économique vos déploiements MySQL existants et nouveaux, de façon à ce que vous puissiez vous concentrer sur votre activité et vos applications. Amazon RDS assure l'administration d'Aurora en gérant des tâches de base de données routinières, telles que l'approvisionnement, l'application de correctifs, la sauvegarde, la récupération, la détection d'échecs et la réparation. Amazon RDS fournit également des outils de migration à l'aide de boutons de commande pour convertir vos applications Amazon RDS for MySQL en applications Aurora MySQL.

Rubriques

- Mises à jour du moteur de base de données pour Amazon Aurora MySQL

Présentation d'Amazon Aurora MySQL

Les sections suivantes fournissent une présentation d'Amazon Aurora MySQL.

Rubriques

- Améliorations des performances Amazon Aurora MySQL
- Amazon Aurora MySQL et données spatiales
- Aurora MySQL version 3 compatible avec MySQL 8.0
- Aurora MySQL version 2 compatible avec MySQL 5.7

Améliorations des performances Amazon Aurora MySQL

Amazon Aurora inclut les améliorations des performances pour prendre en charge les différents besoins des bases de données commerciales haut de gamme.

Fast Insert

Fast Insert accélère les insertions en parallèle triées sur la clé primaire et s'applique spécifiquement aux instructions `LOAD DATA` et `INSERT INTO ... SELECT ...`. Fast Insert met en cache l'emplacement d'un curseur dans une traversée d'index tout en exécutant l'instruction. Cela évite d'avoir à traverser à nouveau l'index inutilement.

L'insertion rapide n'est activée que pour les tables InnoDB classiques dans Aurora MySQL 3.03.2 et versions ultérieures. Cette optimisation ne fonctionne pas pour les tables temporaires InnoDB. Elle est désactivée dans Aurora MySQL version 2 pour toutes les versions 2.11 et 2.12. L'optimisation de l'insertion rapide ne fonctionne que si l'optimisation de l'indice de hachage adaptatif est désactivée.

Vous pouvez surveiller les métriques suivantes pour déterminer l'efficacité de Fast Insert pour votre cluster de bases de données :

- `aurora_fast_insert_cache_hits` : compteur incrémenté quand le curseur en cache est extrait et contrôlé avec succès.
- `aurora_fast_insert_cache_misses` : compteur incrémenté quand le curseur mis en cache n'est plus valide et qu'Aurora exécute une traversée d'index normale.

Vous pouvez extraire la valeur actuelle de la métrique Fast Insert à l'aide de la commande suivante :

```
mysql> show global status like 'Aurora_fast_insert%';
```

Vous obtenez une sortie similaire à ce qui suit :

```
+---------------------------------+-----------+
| Variable_name                   | Value     |
+---------------------------------+-----------+
| Aurora_fast_insert_cache_hits   | 3598300   |
| Aurora_fast_insert_cache_misses | 436401336 |
+---------------------------------+-----------+
```

Amazon Aurora MySQL et données spatiales

La liste suivante résume les fonctionnalités spatiales Aurora MySQL principales et explique comment elles correspondent aux fonctionnalités spatiales dans MySQL :

- Aurora MySQL version 2 prend en charge les mêmes types de données spatiales et fonctions de relations spatiales que MySQL 5.7. Pour plus d'informations sur ces types de données et fonctions, consultez Types de données spatiales et Fonctions de relation spatiale dans la documentation MySQL 5.7.
- Aurora MySQL version 3 prend en charge les mêmes types de données spatiales et fonctions de relations spatiales que MySQL 8.0. Pour plus d'informations sur ces types de données et fonctions, consultez Types de données spatiales et Fonctions de relation spatiale dans la documentation MySQL 8.0.
- Aurora MySQL prend en charge l'indexation spatiale sur les tables InnoDB. L'indexation spatiale améliore les performances des requêtes sur des jeux de données volumineux pour les requêtes sur des données spatiales. Dans MySQL, l'indexation spatiale pour les tables InnoDB est disponible dans MySQL 5.7 et 8.0.

  Aurora MySQL utilise une stratégie d'indexation spatiale différente de celle utilisée par MySQL pour des performances élevées avec des requêtes spatiales. L'implémentation des index spatiaux Aurora utilise une courbe remplissant l'espace sur un arbre B, qui est destiné à fournir des performances plus élevées pour les analyses de plages spatiales qu'un arbre R.

> **Note**
>
> Dans Aurora MySQL, une transaction sur une table avec un index spatial défini sur une colonne comportant un identifiant de référence spatiale (SRID) ne peut pas insérer dans une zone sélectionnée pour la mise à jour par une autre transaction.

Les instructions suivantes en langage de définition de données (DDL) sont prises en charge pour la création d'index sur les colonnes qui utilisent des types de données spatiales.

CREATE TABLE

Vous pouvez utiliser les mots clés `SPATIAL INDEX` dans une instruction `CREATE TABLE` pour ajouter un index spatial à une colonne dans une nouvelle table. Voici un exemple.

```
CREATE TABLE test (shape POLYGON NOT NULL, SPATIAL INDEX(shape));
```

ALTER TABLE

Vous pouvez utiliser les mots clés `SPATIAL INDEX` dans une instruction `ALTER TABLE` pour ajouter un index spatial à une colonne dans une table existante. Voici un exemple.

```
ALTER TABLE test ADD SPATIAL INDEX(shape);
```

CREATE INDEX

Vous pouvez utiliser le mot clé `SPATIAL` dans une instruction `CREATE INDEX` pour ajouter un index spatial à une colonne dans une table existante. Voici un exemple.

```
CREATE SPATIAL INDEX shape_index ON test (shape);
```

Aurora MySQL version 3 compatible avec MySQL 8.0

Vous pouvez utiliser Aurora MySQL version 3 pour obtenir les dernières fonctionnalités compatibles avec MySQL, des améliorations de performances et des corrections de bugs. Vous trouverez ci-après des informations sur Aurora MySQL version 3 avec compatibilité MySQL 8.0. Vous pouvez apprendre à mettre à niveau les clusters et applications vers Aurora MySQL version 3.

Certaines fonctionnalités d'Aurora, comme Aurora Serverless v2, nécessitent la version 3 d'Aurora MySQL.

Rubriques

Fonctions de MySQL 8.0 Community Edition

La version initiale d'Aurora MySQL version 3 est compatible avec MySQL 8.0.23 Community Edition. MySQL 8.0 introduit plusieurs nouvelles fonctions, dont les suivantes :

- Prise en charge du langage de définition des données (DDL) atomiques. Pour plus d'informations, consultez Prise en charge du langage de définition des données (DDL) atomiques.
- Fonctions JSON Pour plus d'informations, consultez Fonctions JSON dans le manuel de référence MySQL.
- Fonctions de fenêtrage. Pour plus d'informations, consultez Fonctions de fenêtrage dans le manuel de référence MySQL.
- Expressions de table communes (CTE), à l'aide de la clause `WITH`. Pour plus d'informations, consultez WITH (expressions de table communes) dans le manuel de référence MySQL.
- Clauses `ADD COLUMN` et `RENAME COLUMN` optimisées pour l'instruction `ALTER TABLE`. Ces optimisations sont appelées « DDL instantané ». Aurora MySQL version 3 est compatible avec la fonctionnalité DDL instantané de MySQL version communautaire. L'ancienne fonction Aurora Fast DDL n'est pas utilisée. Pour plus d'informations sur l'utilisation du DDL instantané, consultez Instant DDL (Aurora MySQL version 3).
- Index décroissants, fonctionnels et invisibles. Pour plus d'informations, consultez Index Invisibles, Index décroissants et Instruction CREATE INDEX dans le manuel de référence MySQL.

- Privilèges basés sur des rôles contrôlés par des instructions SQL. Pour plus d'informations sur les modifications apportées au modèle de privilèges, consultez Modèle de privilège basé sur les rôles.
- Clauses `NOWAIT` et `SKIP LOCKED` avec l'instruction `SELECT ... FOR SHARE`. Ces clauses évitent d'attendre que d'autres transactions libèrent les verrous de ligne. Pour plus d'informations, consultez Lectures de verrouillage dans le manuel de référence MySQL.
- Améliorations apportées à la réplication des journaux binaires (binlog). Pour plus d'informations sur Aurora MySQL, consultez Réplication de journaux binaires. Vous pouvez notamment effectuer une réplication filtrée. Pour plus d'informations sur l'utilisation de la réplication filtrée, consultez Comment les serveurs évaluent les règles de filtrage de réplication dans le manuel de référence MySQL.
- Indicateurs. Certains indicateurs compatibles avec MySQL 8.0 ont déjà été rétroportés vers Aurora MySQL version 2. Pour obtenir des informations sur l'utilisation des indicateurs avec Aurora MySQL, consultez Indicateurs Aurora MySQL. Pour obtenir la liste complète des indicateurs dans MySQL 8.0 version communautaire, consultez Indicateurs de l'optimiseur dans le manuel de référence MySQL.

Pour obtenir la liste complète des fonctions ajoutées à MySQL 8.0 Community Edition, consultez l'article de blog Liste complète des nouvelles fonctions de MySQL 8.0.

Aurora MySQL version 3 inclut également des modifications apportées aux mots-clés pour un langage inclusif, rétroportés depuis MySQL 8.0.26 version communautaire. Pour plus d'informations sur ces modifications, consultez Changements linguistiques inclusifs pour Aurora MySQL version 3.

Prérequis Aurora MySQL version 3 pour Aurora MySQL sans serveur v2

Aurora MySQL version 3 est un prérequis pour toutes les instances de base de données dans un cluster Aurora MySQL sans serveur v2. Aurora MySQL sans serveur v2 prend en charge les instances de lecture dans un cluster de bases de données, ainsi que d'autres fonctionnalités Aurora qui ne sont pas disponibles pour Aurora MySQL sans serveur v1. Il offre également une mise à l'échelle plus rapide et plus granulaire qu'Aurora MySQL sans serveur v1.

Notes de mise à jour d'Aurora MySQL version 3

Pour les notes de mise à jour de toutes les versions d'Aurora MySQL version 3, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 3 dans Notes de mise à jour d'Aurora MySQL.

Nouvelles optimisations des requêtes parallèles

L'optimisation des requêtes parallèles Aurora s'applique désormais à d'autres opérations SQL :

- La requête parallèle s'applique désormais aux tables contenant les types de données `TEXT`, `BLOB`, `JSON`, `GEOMETRY` et `VARCHAR`, et `CHAR` de plus de 768 octets.
- Les requêtes parallèles peuvent optimiser les requêtes impliquant des tables partitionnées.
- Une requête parallèle permet d'optimiser les requêtes impliquant des appels de fonction agrégés dans la liste de sélection et la clause `HAVING`.

Pour plus d'informations sur ces améliorations, consultez Mise à niveau des clusters de requêtes parallèles vers Aurora MySQL version 3. Pour obtenir des informations générales sur la requête parallèle Aurora, consultez Requêtes parallèles pour Amazon Aurora MySQL.

Optimisations destinées à réduire le temps de redémarrage de la base de données

Votre cluster de bases de données Aurora MySQL doit être hautement disponible pendant les interruptions planifiées et imprévues.

Les administrateurs de la base de données doivent effectuer une maintenance occasionnelle de celle-ci. Cette maintenance inclut l'application de correctifs, les mises à niveau, les modifications des paramètres de la base de données nécessitant un redémarrage manuel, le basculement pour réduire le temps nécessaire pour modifier la classe d'instance, etc. Ces actions planifiées nécessitent une durée d'indisponibilité.

Cependant, les durées d'indisponibilité peuvent également être provoquées par des actions imprévues, telles qu'un basculement inattendu dû à une défaillance matérielle sous-jacente ou à une limitation des ressources de la base de données. Toutes ces actions planifiées et imprévues entraînent le redémarrage de la base de données.

Dans Aurora MySQL 3.05 et versions ultérieures, nous avons introduit des optimisations qui réduisent le temps de redémarrage de la base de données. Ces optimisations permettent de réduire la durée d'indisponibilité de 65 % et atténuent les perturbations engendrées sur les charges de travail de votre base de données après un redémarrage.

Lors du démarrage de la base de données, de nombreux composants de la mémoire interne sont initialisés. Le plus important d'entre eux est le pool de mémoire tampon InnoDB, qui, dans Aurora MySQL, représente 75 % de la taille de la mémoire de l'instance par défaut. Nos tests ont révélé

que le temps d'initialisation est proportionnel à la taille du pool de mémoire tampon InnoDB et qu'il évolue donc en fonction de la taille de la classe d'instance de la base de données. Lors de cette phase d'initialisation, la base de données ne peut pas accepter de connexions, ce qui entraîne une durée d'indisponibilité plus longue lors des redémarrages. La première phase du redémarrage rapide d'Aurora MySQL optimise l'initialisation du pool de mémoire tampon, ce qui réduit le temps d'initialisation de la base de données et donc le temps de redémarrage global.

Pour plus d'informations, consultez le blog Reduce downtime with Amazon Aurora MySQL database restart time optimizations.

Nouveau comportement de table temporaire dans Aurora MySQL version 3

Aurora MySQL version 3 gère les tables temporaires différemment des versions précédentes d'Aurora MySQL. Ce nouveau comportement est hérité de MySQL 8.0 Community Edition. Il existe deux types de tables temporaires qui peuvent être créées avec Aurora MySQL version 3 :

- Tables temporaires internes (ou implicites) : créées par le moteur Aurora MySQL pour gérer les opérations telles que le tri, l'agrégation, les tables dérivées ou les expressions de table communes (CTE).
- Tables temporaires créées par l'utilisateur (ou explicites) : créées par le moteur Aurora MySQL lorsque vous utilisez l'instruction `CREATE TEMPORARY TABLE`.

Il existe des considérations supplémentaires pour les tables temporaires internes et créées par l'utilisateur sur les instances de base de données de lecteur Aurora. Ces modifications sont présentées dans les sections suivantes.

Rubriques

- Moteur de stockage pour tables temporaires internes (implicites)
- Limitation de la taille des tables temporaires internes en mémoire
- Atténuation des problèmes de remplissage pour les tables temporaires internes sur les réplicas Aurora
- Optimisation du paramètre temptable_max_mmap sur les instances de base de données Aurora MySQL
- Tables temporaires (explicites) créées par l'utilisateur sur les instances de base de données de lecteur
- Erreurs de création d'une table temporaire et atténuation

Moteur de stockage pour tables temporaires internes (implicites)

Lors de la génération de jeux de résultats intermédiaires, Aurora MySQL tente initialement d'écrire dans des tables temporaires en mémoire. Cela peut échouer, en raison de types de données incompatibles ou de limites configurées. Si c'est le cas, la table temporaire est convertie en table temporaire sur disque plutôt que d'être conservée en mémoire. Pour plus d'informations, consultez Internal Temporary Table Use in MySQL dans la documentation MySQL.

Dans Aurora MySQL version 3, le fonctionnement des tables temporaires internes est différent des versions précédentes d'Aurora MySQL. Pour ces tables temporaires, au lieu de choisir entre les moteurs de stockage InnoDB et MyISAM, vous avez maintenant le choix entre les moteurs de stockage `TempTable` et `MEMORY`.

Avec le moteur de stockage `TempTable`, vous disposez d'un choix supplémentaire pour gérer certaines données. Les données affectées dépassent la capacité du pool de mémoire qui contient toutes les tables temporaires internes de l'instance de base de données.

Ces choix peuvent influencer les performances des requêtes qui génèrent des volumes élevés de données temporaires, par exemple lors de l'exécution d'agrégations telles que `GROUP BY` sur des tables volumineuses.

> **Tip**
>
> Si votre charge de travail inclut des requêtes qui génèrent des tables temporaires internes, confirmez les performances de votre application avec cette modification en exécutant des définitions de points de référence et en surveillant les métriques liées aux performances. Dans certains cas, la quantité de données temporaires correspond à la capacité du pool de mémoire `TempTable` ou est légèrement supérieure à cette dernière. Le cas échéant, nous vous recommandons d'utiliser le paramètre `TempTable` pour les tables temporaires internes et les fichiers mappés en mémoire pour conserver toutes les données excédentaires. Il s'agit de la valeur par défaut.

Le moteur de stockage `TempTable` est le moteur par défaut. `TempTable` utilise un pool de mémoire commun pour toutes les tables temporaires utilisant ce moteur, au lieu d'une limite de mémoire maximale par table. La taille de ce pool de mémoire est spécifiée par le paramètre temptable_max_ram. Elle est par défaut de 1 Gio sur les instances de base de données de 16 Gio de mémoire ou plus, et de 16 Mo sur les instances de base de données de moins de 16 Gio de

mémoire. La taille du pool de mémoire influe sur la consommation de mémoire au niveau de la session.

Dans certains cas, lorsque vous utilisez le moteur de stockage `TempTable`, les données temporaires peuvent dépasser la taille du pool de mémoire. Si tel est le cas, Aurora MySQL stocke les données de débordement à l'aide d'un mécanisme secondaire.

Vous pouvez définir le paramètre temptable_max_mmap pour choisir si les données dépassent la taille des fichiers temporaires mappés en mémoire ou des tables temporaires internes InnoDB sur le disque. Les différents formats de données et les critères de dépassement de ces mécanismes de dépassement peuvent affecter les performances des requêtes. Pour ce faire, Ils influencent la quantité de données écrites sur disque et la demande de débit de stockage sur le disque.

Aurora MySQL version 3 stocke les données de débordement de la manière suivante :

- Sur l'instance de base de données d'enregistreur, les données qui débordent vers des tables temporaires internes InnoDB ou des fichiers temporaires mappés en mémoire se trouvent sur le stockage local de l'instance.
- Sur les instances de base de données de lecteur, les données excédentaires se trouvent toujours dans des fichiers temporaires mappés en mémoire sur le stockage local.

  Les instances en lecture seule ne peuvent stocker aucune donnée sur le volume de cluster Aurora.

Les paramètres de configuration associés aux tables temporaires internes s'appliquent différemment aux instances de lecteur et d'enregistreur sur votre cluster.

- Sur les instances de lecteur, Aurora MySQL utilise toujours le moteur de stockage `TempTable`.
- La taille par défaut de `temptable_max_mmap` est de 1 Gio, pour les instances de lecteur et d'enregistreur, quelle que soit la taille de la mémoire de l'instance de base de données. Vous pouvez ajuster cette valeur à la fois sur les instances d'enregistreur et de lecteur.
- Définir `temptable_max_mmap` sur `0` désactive l'utilisation des fichiers temporaires mappés en mémoire sur les instances d'enregistreur.
- Vous ne pouvez pas définir `temptable_max_mmap` sur `0` sur les instances de lecteur.

> **Note**
>
> Nous déconseillons l'utilisation du paramètre temptable_use_mmap. Il est devenu obsolète et sa prise en charge devrait être supprimée dans une future version de MySQL.

Limitation de la taille des tables temporaires internes en mémoire

Comme indiqué dans Moteur de stockage pour tables temporaires internes (implicites), vous pouvez contrôler les ressources de tables temporaires de manière globale en utilisant les paramètres temptable_max_ram et temptable_max_mmap.

Vous pouvez également limiter la taille de n'importe quelle table temporaire interne en mémoire en utilisant le paramètre de base de données tmp_table_size. Cette limite vise à empêcher les requêtes individuelles de consommer une quantité excessive de ressources de tables temporaires globales, ce qui peut affecter les performances de requêtes simultanées nécessitant ces ressources.

Le paramètre `tmp_table_size` définit la taille maximale des tables temporaires créées par le moteur de stockage `MEMORY` dans Aurora MySQL version 3.

Dans Aurora MySQL version 3.04 ou ultérieure, `tmp_table_size` définit également la taille maximale des tables temporaires créées par le moteur de stockage `TempTable` quand le paramètre de base de données `aurora_tmptable_enable_per_table_limit` a pour valeur `ON`. Ce comportement est désactivé par défaut (`OFF`), ce qui constitue le même comportement que dans Aurora MySQL version 3.03 et versions antérieures.

- Quand `aurora_tmptable_enable_per_table_limit` a pour valeur `OFF`, `tmp_table_size` n'est pas pris en compte pour les tables temporaires internes en mémoire créées par le moteur de stockage `TempTable`.

  Cependant, la limite globale des ressources `TempTable` s'applique toujours. Aurora MySQL a le comportement suivant lorsque la limite globale des ressources `TempTable` est atteinte :

  - Instances de base de données d'enregistreur : Aurora MySQL convertit automatiquement la table temporaire en mémoire en une table temporaire InnoDB sur disque.
  - Instances de base de données de lecteur : la requête se termine par une erreur.

    ```
    ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlxx_xxx' is full
    ```

- Quand `aurora_tmptable_enable_per_table_limit` a pour valeur `ON`, Aurora MySQL a le comportement suivant lorsque la limite `tmp_table_size` est atteinte :
  - Instances de base de données d'enregistreur : Aurora MySQL convertit automatiquement la table temporaire en mémoire en une table temporaire InnoDB sur disque.
  - Instances de base de données de lecteur : la requête se termine par une erreur.

    ```
    ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlxx_xxx' is full
    ```

    La limite globale des ressources `TempTable` et la limite par table s'appliquent toutes les deux dans ce cas.

> **Note**
>
> Le paramètre `aurora_tmptable_enable_per_table_limit` n'a aucun effet quand internal_tmp_mem_storage_engine a pour valeur `MEMORY`. Dans ce cas, la taille maximale d'une table temporaire en mémoire est définie par la valeur tmp_table_size ou max_heap_table_size, la plus petite de ces deux valeurs étant retenue.

Les exemples suivants montrent le comportement du paramètre `aurora_tmptable_enable_per_table_limit` pour les instances de base de données d'enregistreur et de lecteur.

Example d'une instance de base de données d'enregistreur avec **`aurora_tmptable_enable_per_table_limit`** défini sur **`OFF`**

La table temporaire en mémoire n'est pas convertie en table temporaire InnoDB sur disque.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+--------------------+------------------+----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
```

```
+--------------------+------------------+-----------------------------------------
+---------------------+----------------------+
|                   0 | 3.04.0           |                                       0 |
       1073741824 |           1073741824 |
+--------------------+------------------+-----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> show status like '%created_tmp_disk%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| Created_tmp_disk_tables | 0     |
+--------------------------+-------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 60000000) SELECT max(n) FROM cte;
+----------+
| max(n)   |
+----------+
| 60000000 |
+----------+
1 row in set (13.99 sec)

mysql> show status like '%created_tmp_disk%';
+--------------------------+-------+
| Variable_name            | Value |
+--------------------------+-------+
| Created_tmp_disk_tables | 0     |
+--------------------------+-------+
1 row in set (0.00 sec)
```

Example d'une instance de base de données d'enregistreur avec **`aurora_tmptable_enable_per_table_limit`** défini sur **`ON`**

La table temporaire en mémoire est convertie en table temporaire InnoDB sur disque.

```
mysql> set aurora_tmptable_enable_per_table_limit=1;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@tmp_table_size;
+--------------------+-----------------+-----------------------------------------
+------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@tmp_table_size |
+--------------------+-----------------+-----------------------------------------
+------------------+
|                  0 | 3.04.0          |                                        1 |
      16777216 |
+--------------------+-----------------+-----------------------------------------
+------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 0     |
+-------------------------+-------+
1 row in set (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 6000000) SELECT max(n) FROM cte;
+---------+
| max(n)  |
+---------+
| 6000000 |
+---------+
1 row in set (4.10 sec)

mysql> show status like '%created_tmp_disk%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 1     |
+-------------------------+-------+
1 row in set (0.00 sec)
```

Example d'une instance de base de données de lecteur avec **`aurora_tmptable_enable_per_table_limit`** défini sur **`OFF`**

La requête se termine sans erreur car `tmp_table_size` ne s'applique pas, et la limite globale des ressources `TempTable` n'a pas été atteinte.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+-------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+-------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
|                  1 | 3.04.0           |                                       0 |
       1073741824 |           1073741824 |
+-------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 60000000) SELECT max(n) FROM cte;
+----------+
| max(n)   |
+----------+
| 60000000 |
+----------+
1 row in set (14.05 sec)
```

Example d'une instance de base de données de lecteur avec **`aurora_tmptable_enable_per_table_limit`** défini sur **`OFF`**

Cette requête atteint la limite globale des ressources TempTable avec `aurora_tmptable_enable_per_table_limit` défini sur OFF. La requête se termine avec une erreur sur les instances de lecteur.

```
mysql> set aurora_tmptable_enable_per_table_limit=0;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@temptable_max_r
+-------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@temptable_max_ram | @@temptable_max_mmap |
+-------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
|                  1 | 3.04.0           |                                        0 |
       1073741824 |           1073741824 |
+-------------------+-----------------+-----------------------------------------
+---------------------+----------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.01 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 120000000) SELECT max(n) FROM cte;
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlfd_1586_2' is full
```

Example d'une instance de base de données de lecteur avec **`aurora_tmptable_enable_per_table_limit`** défini sur **`ON`**

La requête se termine avec une erreur lorsque la limite `tmp_table_size` est atteinte.

```
mysql> set aurora_tmptable_enable_per_table_limit=1;
Query OK, 0 rows affected (0.00 sec)

mysql> select
 @@innodb_read_only,@@aurora_version,@@aurora_tmptable_enable_per_table_limit,@@tmp_table_size;
+-------------------+-----------------+-----------------------------------------
+------------------+
| @@innodb_read_only | @@aurora_version | @@aurora_tmptable_enable_per_table_limit |
 @@tmp_table_size |
+-------------------+-----------------+-----------------------------------------
+------------------+
|                  1 | 3.04.0           |                                        1 |
      16777216 |
```

```
+--------------------+-----------------+----------------------------------------
+------------------+
1 row in set (0.00 sec)

mysql> set cte_max_recursion_depth=4294967295;
Query OK, 0 rows affected (0.00 sec)

mysql> WITH RECURSIVE cte (n) AS (SELECT 1 UNION ALL SELECT n + 1 FROM cte WHERE n <
 6000000) SELECT max(n) FROM cte;
ERROR 1114 (HY000): The table '/rdsdbdata/tmp/#sqlfd_8_2' is full
```

Atténuation des problèmes de remplissage pour les tables temporaires internes sur les réplicas Aurora

Pour éviter les problèmes de limite de taille pour les tables temporaires, définissez les paramètres `temptable_max_ram` et `temptable_max_mmap` sur une valeur combinée, adaptée aux exigences de votre charge de travail.

Soyez vigilant lorsque vous définissez la valeur du paramètre `temptable_max_ram`. La définition d'une valeur trop élevée réduit la mémoire disponible sur l'instance de base de données, ce qui peut entraîner une condition de mémoire insuffisante. Surveillez la quantité moyenne de mémoire libérable sur l'instance de base de données. Déterminez ensuite une valeur appropriée pour `temptable_max_ram`, de sorte qu'il vous reste une quantité raisonnable de mémoire libre sur l'instance. Pour plus d'informations, consultez Problèmes liés à la mémoire libérable dans Amazon Aurora.

Il est également important de surveiller la taille du stockage local et la consommation d'espace des tables temporaires. Vous pouvez surveiller le stockage temporaire disponible pour une instance de base de données spécifique à l'aide de la métrique Amazon CloudWatch `FreeLocalStorage` décrite dans CloudWatch Métriques Amazon pour Amazon Aurora.

> **Note**
>
> Cette procédure ne fonctionne pas quand le paramètre `aurora_tmptable_enable_per_table_limit` est défini sur `ON`. Pour plus d'informations, consultez Limitation de la taille des tables temporaires internes en mémoire.

Example 1

Vous savez que vos tables temporaires atteignent une taille cumulée de 20 Gio. Vous souhaitez définir les tables temporaires en mémoire sur 2 Gio et atteindre une taille maximale de 20 Gio sur disque.

Définissez `temptable_max_ram` sur **`2,147,483,648`** et `temptable_max_mmap` sur **`21,474,836,480`**. Ces valeurs sont exprimées en octets.

Ces valeurs de paramètre garantissent que vos tables temporaires peuvent atteindre un total cumulé de 22 Gio.

Example 2

La taille actuelle de votre instance est 16xlarge ou supérieure. Vous ne connaissez pas la taille totale des tables temporaires dont vous pourriez avoir besoin. Vous souhaitez pouvoir utiliser jusqu'à 4 Gio en mémoire et jusqu'à la taille de stockage maximale disponible sur disque.

Définissez `temptable_max_ram` sur **`4,294,967,296`** et `temptable_max_mmap` sur **`1,099,511,627,776`**. Ces valeurs sont exprimées en octets.

Ici, vous êtes en train de définir `temptable_max_mmap` sur 1 Tio, ce qui est inférieur au stockage local maximal de 1,2 Tio sur une instance de base de données Aurora 16xlarge.

Sur une taille d'instance plus petite, ajustez la valeur de `temptable_max_mmap` afin qu'elle ne remplisse pas le stockage local disponible. Par exemple, une instance 2xlarge ne dispose que de 160 Gio de stockage local disponible. Par conséquent, nous vous recommandons de définir la valeur sur 160 Gio au maximum. Pour plus d'informations sur le stockage local disponible pour les tailles d'instance de base de données, consultez Limites de stockage temporaires pour Aurora MySQL.

Optimisation du paramètre temptable_max_mmap sur les instances de base de données Aurora MySQL

Le paramètre `temptable_max_mmap` d'Aurora MySQL contrôle la quantité maximale d'espace disque local qui peut être utilisée par les fichiers mappés en mémoire avant de déborder vers les tables temporaires InnoDB sur disque (sur les instances de base de données d'enregistreur) ou de provoquer une erreur (sur les instances de base de données de lecteur). Une configuration appropriée de ce paramètre d'instance de base de données peut aider à optimiser les performances de vos instances de base de données.

Prérequis

1. Assurez-vous que le schéma de performances est activé. Pour effectuer une vérification, exécutez la commande suivante.

   ```
   SELECT @@performance_schema;
   ```

   La valeur de sortie 1 indique qu'il est activé.

2. Vérifiez que l'instrumentation de mémoire de table temporaire est activée. Pour effectuer une vérification, exécutez la commande suivante.

   ```
   SELECT name, enabled FROM performance_schema.setup_instruments WHERE name LIKE
    '%memory%temptable%';
   ```

   La colonne `enabled` indique `YES` pour les entrées d'instrumentation de mémoire de table temporaire pertinentes.

Surveillance de l'utilisation des tables temporaires

Lorsque vous définissez la valeur initiale pour `temptable_max_mmap`, nous vous recommandons de commencer par 80 % de la taille de stockage local pour la classe d'instance de base de données que vous utilisez. Les tables temporaires disposeront ainsi de suffisamment d'espace disque pour fonctionner efficacement, tout en laissant de la place à d'autres utilisations du disque sur l'instance.

Pour connaître la taille de stockage local de votre classe d'instance de base de données, consultez Limites de stockage temporaires pour Aurora MySQL.

Par exemple, si vous utilisez la classe d'instance de base de données db.r5.large, la taille du stockage local est de 32 Gio. Dans ce cas, vous devez initialement définir le paramètre `temptable_max_mmap` pour qu'il corresponde à 80 % de 32 Gio, soit 25,6 Gio.

Après avoir défini la valeur `temptable_max_mmap` initiale, exécutez votre charge de travail maximale sur les instances Aurora MySQL. Surveillez l'utilisation actuelle et l'utilisation élevée du disque des tables temporaires à l'aide de la requête SQL suivante :

```
SELECT event_name, current_count, current_alloc, current_avg_alloc, high_count,
 high_alloc, high_avg_alloc
FROM sys.memory_global_by_current_bytes WHERE event_name LIKE 'memory/temptable/%';
```

Cette requête permet de récupérer les informations suivantes :

- `event_name` — Nom de l'événement d'utilisation du disque ou de la mémoire de table temporaire.
- `current_count` — Nombre actuel de blocs de disque ou de mémoire de table temporaire alloués.
- `current_alloc` — Quantité actuelle de mémoire ou de disque allouée aux tables temporaires.
- `current_avg_alloc` — Taille moyenne actuelle des blocs de disque ou de mémoire de table temporaire.
- `high_count` — Nombre maximal de blocs disque ou de mémoire de table temporaire alloués.
- `high_alloc` — Quantité maximale de mémoire ou de disque allouée aux tables temporaires.
- `high_avg_alloc` — Taille moyenne maximale des blocs de disque ou de mémoire de table temporaire.

Si vos requêtes échouent avec une erreur de type La table est pleine avec ce paramètre, cela indique que votre charge de travail nécessite plus d'espace disque pour les opérations des tables temporaires. Dans ce cas, envisagez d'augmenter la taille de votre instance de base de données pour qu'elle dispose d'un espace de stockage local plus important.

Configuration de la valeur **`temptable_max_mmap`** optimale

Utilisez la procédure suivante pour surveiller et définir la taille appropriée pour le paramètre `temptable_max_mmap`.

1. Passez en revue le résultat de la requête précédente et identifiez le pic d'utilisation du disque de table temporaire, comme indiqué dans la colonne `high_alloc`.
2. En fonction du pic d'utilisation du disque de table temporaire, ajustez le paramètre `temptable_max_mmap` dans le groupe de paramètres de base de données de vos instances de base de données Aurora MySQL.

   Définissez cette valeur pour qu'elle soit légèrement supérieure au pic d'utilisation du disque de table temporaire afin de tenir compte de la croissance future.
3. Appliquez les modifications du groupe de paramètres à vos instances de base de données.
4. Surveillez à nouveau l'utilisation du disque de table temporaire pendant votre charge de travail maximale pour vous assurer que la nouvelle valeur `temptable_max_mmap` est appropriée.
5. Répétez les étapes précédentes autant de fois que nécessaire pour optimiser le paramètre `temptable_max_mmap`.

Tables temporaires (explicites) créées par l'utilisateur sur les instances de base de données de lecteur

Vous pouvez créer des tables temporaires explicites en utilisant le mot-clé `TEMPORARY` dans votre instruction `CREATE TABLE`. Les tables temporaires explicites sont prises en charge sur l'instance de base de données d'enregistreur dans un cluster de bases de données Aurora. Vous pouvez également utiliser des tables temporaires explicites sur les instances de base de données de lecteur, mais les tables ne peuvent pas imposer l'utilisation du moteur de stockage InnoDB.

Pour éviter les erreurs lors de la création de tables temporaires explicites sur les instances de base de données de lecture Aurora MySQL, assurez-vous d'exécuter toutes les instructions `CREATE TEMPORARY TABLE` de l'une ou l'autre des manières suivantes, ou des deux :

- Ne spécifiez pas la clause `ENGINE=InnoDB`.
- Ne définissez pas le mode SQL sur `NO_ENGINE_SUBSTITUTION`.

Erreurs de création d'une table temporaire et atténuation

L'erreur que vous recevez est différente selon que vous utilisez ou non une instruction `CREATE TEMPORARY TABLE` simple ou la variante `CREATE TEMPORARY TABLE AS SELECT`. Les exemples suivants montrent les différents types d'erreurs.

Ce comportement de table temporaire s'applique uniquement aux instances en lecture seule. Ce premier exemple confirme que c'est le type d'instance à laquelle la session est connectée.

```
mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+
```

Les instructions `CREATE TEMPORARY TABLE` simples échouent lorsque le mode SQL `NO_ENGINE_SUBSTITUTION` est activé. Lorsque `NO_ENGINE_SUBSTITUTION` est désactivé (par défaut), la substitution du moteur approprié est effectuée et la création de la table temporaire aboutit.

```
mysql> set sql_mode = 'NO_ENGINE_SUBSTITUTION';

mysql>  CREATE TEMPORARY TABLE tt2 (id int) ENGINE=InnoDB;
```

```
ERROR 3161 (HY000): Storage engine InnoDB is disabled (Table creation is disallowed).

mysql> SET sql_mode = '';

mysql> CREATE TEMPORARY TABLE tt4 (id int) ENGINE=InnoDB;

mysql> SHOW CREATE TABLE tt4\G
*************************** 1. row ***************************
       Table: tt4
Create Table: CREATE TEMPORARY TABLE `tt4` (
  `id` int DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Les instructions `CREATE TEMPORARY TABLE AS SELECT` échouent quand le mode SQL `NO_ENGINE_SUBSTITUTION` est activé. Lorsque `NO_ENGINE_SUBSTITUTION` est désactivé (par défaut), la substitution du moteur approprié est effectuée et la création de la table temporaire aboutit.

```
mysql> set sql_mode = 'NO_ENGINE_SUBSTITUTION';

mysql> CREATE TEMPORARY TABLE tt1 ENGINE=InnoDB AS SELECT * FROM t1;
ERROR 3161 (HY000): Storage engine InnoDB is disabled (Table creation is disallowed).

mysql> SET sql_mode = '';

mysql> show create table tt3;
+-------+-----------------------------------------------------------+
| Table | Create Table                                              |
+-------+-----------------------------------------------------------+
| tt3   | CREATE TEMPORARY TABLE `tt3` (
  `id` int DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci |
+-------+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

Pour plus d'informations sur les aspects du stockage et les implications en termes de performances des tables temporaires dans la version 3 d'Aurora MySQL, consultez l'article de blog Use the TempTable storage engine on Amazon RDS for MySQL and Amazon Aurora MySQL.

Comparaison d'Aurora MySQL version 2 et Aurora MySQL version 3

Utilisez les éléments suivants pour en savoir plus sur les modifications à prendre en compte lorsque vous mettez à niveau le cluster Aurora MySQL version 2 vers la version 3.

Rubriques

Prise en charge du langage de définition des données (DDL) atomiques

L'un des changements les plus importants entre MySQL 5.7 et 8.0 est l'introduction de l'Atomic Data Dictionary. Avant MySQL 8.0, le dictionnaire de données MySQL utilisait une approche basée sur des fichiers pour stocker les métadonnées telles que les définitions de tables (.frm), les déclencheurs (.trg) et les fonctions séparément des métadonnées du moteur de stockage (telles que celles d'InnoDB). Cela posait certains problèmes, y compris le risque que les tables deviennent « orphelines » si un événement inattendu se produisait au cours d'une opération DDL, entraînant une désynchronisation des métadonnées basées sur les fichiers et des métadonnées du moteur de stockage.

Pour résoudre ce problème, MySQL 8.0 inclut l'Atomic Data Dictionary, qui stocke toutes les métadonnées dans un ensemble de tables InnoDB internes dans le schéma `mysql`. Cette nouvelle architecture fournit une méthode transactionnelle conforme à l'ACID pour gérer les métadonnées des bases de données, résolvant ainsi le problème de « DDL atomique » rencontré par l'ancienne approche basée sur les fichiers. Pour plus d'informations sur l'Atomic Data Dictionary, consultez Suppression du stockage de métadonnées basé sur des fichiers et Prise en charge des instructions de définition de données atomiques dans le manuel de référence MySQL.

En raison de cette modification d'architecture, vous devez tenir compte des points suivants lors de la mise à niveau d'Aurora MySQL version 2 vers la version 3 :

- Les métadonnées basées sur les fichiers dans la version 2 doivent être migrées vers les nouvelles tables du dictionnaire de données lors du processus de mise à niveau vers la version 3. Selon le nombre d'objets de base de données migrés, cela peut prendre un certain temps.

- Les modifications ont également introduit de nouvelles incompatibilités qui devront peut-être être corrigées avant de pouvoir passer de MySQL 5.7 à 8.0. Par exemple, la version 8.0 contient de nouveaux mots clés réservés susceptibles d'entrer en conflit avec les noms d'objets de base de données existants.

Pour vous aider à identifier ces incompatibilités avant de mettre à niveau le moteur, Aurora MySQL exécute une série de vérification de la compatibilité des mises à niveau (vérifications préalables) afin de déterminer s'il existe des objets incompatibles dans votre dictionnaire de base de données, avant d'effectuer la mise à niveau du dictionnaire de données. Pour plus d'informations sur ces vérifications préalables, consultez Vérifications préalables aux mises à niveau de version majeure pour Aurora MySQL.

Différences de fonctions entre Aurora MySQL version 2 et 3

Les fonctions Amazon Aurora MySQL suivantes sont prises en charge dans Aurora MySQL pour MySQL 5.7, mais pas dans Aurora MySQL pour MySQL 8.0 :

- Vous ne pouvez pas utiliser Aurora MySQL version 3 pour les clusters Aurora Serverless v1. Aurora MySQL version 3 fonctionne avec Aurora Serverless v2.
- Le mode laboratoire ne s'applique pas à Aurora MySQL version 3. Il n'existe aucune fonctionnalité de mode laboratoire dans Aurora MySQL version 3. Le DDL instantané remplace la fonction DDL en ligne rapide qui était précédemment disponible en mode laboratoire. Pour obtenir un exemple, consultez Instant DDL (Aurora MySQL version 3).
- Le cache de requêtes est supprimé de MySQL 8.0 version communautaire et d'Aurora MySQL version 3.
- Aurora MySQL version 3 est compatible avec la fonction Joindre par hachage de MySQL version communautaire. L'implémentation spécifique à Aurora des jointures de hachage dans Aurora MySQL version 2 n'est pas utilisée. Pour plus d'informations sur l'utilisation des jointures de hachage avec une requête parallèle Aurora, consultez Activation de la jointure par hachage pour les clusters de requête parallèle et Indicateurs Aurora MySQL. Pour obtenir des informations générales sur l'utilisation des jointures de hachage, consultez Optimisation des jointures de hachage dans le manuel de référence MySQL.
- La procédure `mysql.lambda_async` stockée rendue obsolète dans Aurora MySQL version 2 est supprimée dans la version 3. Pour la version 3, utilisez la fonction asynchrone `lambda_async` à la place.

- Le jeu de caractères par défaut dans Aurora MySQL version 3 est `utf8mb4`. Dans Aurora MySQL version 2, le jeu de caractères par défaut était `latin1`. Pour plus d'informations sur ce jeu de caractères, consultez Jeu de caractères utf8mb4 (encodage Unicode 4 octets en UTF-8) dans le manuel de référence MySQL.

Certaines fonctions Aurora MySQL sont disponibles pour certaines combinaisons de région AWS et de version du moteur de base de données. Pour en savoir plus, consultez Fonctionnalités prises en charge dans Amazon Aurora by Région AWS et dans le moteur de base de données Aurora.

Assistance pour les classes d'instance

Aurora MySQL version 3 prend en charge un ensemble de classes d'instance différent de celui d'Aurora MySQL version 2 :

- Pour les instances plus volumineuses, vous pouvez utiliser les classes d'instance modernes telles que `db.r5`, `db.r6g` et `db.x2g`.
- Pour les instances plus petites, vous pouvez utiliser les classes d'instance modernes telles que `db.t3` et `db.t4g`.

> **Note**
>
> Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à la production. Pour plus de détails sur les classes d'instance T, consultez Utilisation de classes d'instance T pour le développement et les tests.

Les classes d'instance suivantes d'Aurora MySQL version 2 ne sont pas disponibles pour Aurora MySQL version 3 :

- `db.r4`
- `db.r3`
- `db.t3.small`
- `db.t2`

Dans vos scripts d'administration, vérifiez les instructions CLI qui créent des instances de base de données Aurora MySQL. Codez en dur les noms de classes d'instance non disponibles pour Aurora

MySQL version 3. Si nécessaire, modifiez les noms de classes d'instance par ceux pris en charge par Aurora MySQL version 3.

> **Tip**
>
> Pour vérifier les classes d'instance que vous pouvez utiliser pour une combinaison spécifique de version Aurora MySQL et de région AWS, utilisez la commande `describe-orderable-db-instance-options` de l'AWS CLI.

Pour plus d'informations sur les classes d'instance Aurora, consultez Classes d'instance de base de données Amazon Aurora.

Modifications des paramètres d'Aurora MySQL version 3

Aurora MySQL version 3 inclut de nouveaux paramètres de configuration au niveau du cluster et de l'instance. Aurora MySQL version 3 supprime également certains paramètres présents dans Aurora MySQL version 2. Certains noms de paramètres sont modifiés suite à l'initiative visant un langage inclusif. Pour des raisons de compatibilité ascendante, vous pouvez toujours récupérer les valeurs des paramètres à l'aide des anciens noms ou des nouveaux noms. Toutefois, vous devez utiliser les nouveaux noms pour spécifier des valeurs de paramètres dans un groupe de paramètres personnalisés.

Dans Aurora MySQL version 3, la valeur du paramètre `lower_case_table_names` est définie de façon permanente au moment de la création du cluster. Si vous utilisez une autre valeur que la valeur par défaut pour cette option, configurez votre groupe de paramètres personnalisés Aurora MySQL version 3 avant la mise à niveau. Spécifiez ensuite le groupe de paramètres pendant l'opération de création de cluster ou de restauration d'instantanés.

> **Note**
>
> Avec une base de données globale Aurora basée sur Aurora MySQL, vous ne pouvez pas effectuer une mise à niveau sur place d'Aurora MySQL version 2 vers la version 3 si le paramètre `lower_case_table_names` est activé. Utilisez plutôt la méthode de restauration des instantanés.

Dans Aurora MySQL version 3, les paramètres `init_connect` et `read_only` ne s'appliquent pas aux utilisateurs disposant du privilège `CONNECTION_ADMIN`. Cela inclut l'utilisateur principal d'Aurora. Pour plus d'informations, consultez Modèle de privilège basé sur les rôles.

Pour obtenir la liste de tous les paramètres du cluster Aurora MySQL, consultez Paramètres de niveau cluster. Le tableau couvre tous les paramètres d'Aurora MySQL versions 2 et 3. Le tableau contient des notes indiquant les nouveaux paramètres dans Aurora MySQL version 3 ou ceux qui ont été supprimés d'Aurora MySQL version 3.

Pour obtenir la liste complète de tous les paramètres de l'instance Aurora MySQL, consultez Paramètres de niveau instance. Le tableau couvre tous les paramètres d'Aurora MySQL versions 2 et 3. Le tableau contient des notes indiquant les nouveaux paramètres dans Aurora MySQL version 3 et ceux qui ont été supprimés d'Aurora MySQL version 3. Il contient également des notes indiquant les paramètres modifiables dans les versions antérieures, mais pas Aurora MySQL version 3.

Pour plus d'informations sur les noms de paramètres modifiés, consultez Changements linguistiques inclusifs pour Aurora MySQL version 3.

Variables d'état

Pour plus d'informations sur les variables d'état non applicables à Aurora MySQL, consultez Variables d'état MySQL ne s'appliquant pas à Aurora MySQL.

Changements linguistiques inclusifs pour Aurora MySQL version 3

Aurora MySQL version 3 est compatible avec la version 8.0.23 de MySQL Community Edition. Aurora MySQL version 3 inclut également des modifications depuis MySQL 8.0.26 liées aux mots-clés et aux schémas de système pour un langage inclusif. Par exemple, il est préférable d'utiliser la commande `SHOW REPLICA STATUS` plutôt que la commande `SHOW SLAVE STATUS`.

Les métriques Amazon CloudWatch suivantes portent de nouveaux noms dans Aurora MySQL version 3.

Dans Aurora MySQL version 3, seuls les nouveaux noms de métriques sont disponibles. Assurez-vous de mettre à jour toutes les alarmes ou autres automatisations qui reposent sur des noms de métriques lorsque vous effectuez une mise à niveau vers Aurora MySQL version 3.

Ancien nom	Nouveau nom	
ForwardingMasterDMLLatency	ForwardingWriterDMLLatency	
ForwardingMasterOpenSessions	ForwardingWriterOpenSessions	
AuroraDMLRejectedMasterFull	AuroraDMLRejectedWriterFull	
ForwardingMasterDMLThroughput	ForwardingWriterDMLThroughput	

Les variables d'état suivantes portent de nouveaux noms dans Aurora MySQL version 3.

Pour des raisons de compatibilité, vous pouvez utiliser l'un ou l'autre des noms dans la version initiale d'Aurora MySQL version 3. Les anciens noms de variables d'état seront supprimés dans une version ultérieure.

Nom à supprimer	Nom nouveau ou préféré	
Aurora_fwd_master_dml_stmt_duration	Aurora_fwd_writer_dml_stmt_duration	
Aurora_fwd_master_dml_stmt_count	Aurora_fwd_writer_dml_stmt_count	
Aurora_fwd_master_select_stmt_duration	Aurora_fwd_writer_select_stmt_duration	
Aurora_fwd_master_select_stmt_count	Aurora_fwd_writer_select_stmt_count	
Aurora_fwd_master_errors_session_timeout	Aurora_fwd_writer_errors_session_timeout	

Nom à supprimer	Nom nouveau ou préféré
`Aurora_fwd_master_open_sessions`	`Aurora_fwd_writer_open_sessions`
`Aurora_fwd_master_errors_session_limit`	`Aurora_fwd_writer_errors_session_limit`
`Aurora_fwd_master_errors_rpc_timeout`	`Aurora_fwd_writer_errors_rpc_timeout`

Les paramètres de configuration suivants portent de nouveaux noms dans Aurora MySQL version 3.

Pour des raisons de compatibilité, vous pouvez vérifier les valeurs des paramètres dans le client `mysql` en utilisant l'un ou l'autre des noms dans la version initiale d'Aurora MySQL version 3. Vous pouvez uniquement utiliser les nouveaux noms lorsque vous modifiez les valeurs dans un groupe de paramètres personnalisés. Les anciens noms de paramètres seront supprimés dans une version ultérieure.

Nom à supprimer	Nom nouveau ou préféré
`aurora_fwd_master_idle_timeout`	`aurora_fwd_writer_idle_timeout`
`aurora_fwd_master_max_connections_pct`	`aurora_fwd_writer_max_connections_pct`
`master_verify_checksum`	`source_verify_checksum`
`sync_master_info`	`sync_source_info`
`init_slave`	`init_replica`
`rpl_stop_slave_timeout`	`rpl_stop_replica_timeout`

Nom à supprimer	Nom nouveau ou préféré
log_slow_slave_statements	log_slow_replica_statements
slave_max_allowed_packet	replica_max_allowed_packet
slave_compressed_protocol	replica_compressed_protocol
slave_exec_mode	replica_exec_mode
slave_type_conversions	replica_type_conversions
slave_sql_verify_checksum	replica_sql_verify_checksum
slave_parallel_type	replica_parallel_type
slave_preserve_commit_order	replica_preserve_commit_order
log_slave_updates	log_replica_updates
slave_allow_batching	replica_allow_batching
slave_load_tmpdir	replica_load_tmpdir
slave_net_timeout	replica_net_timeout
sql_slave_skip_counter	sql_replica_skip_counter
slave_skip_errors	replica_skip_errors

Nom à supprimer	Nom nouveau ou préféré
slave_checkpoint_period	replica_checkpoint_period
slave_checkpoint_group	replica_checkpoint_group
slave_transaction_retries	replica_transaction_retries
slave_parallel_workers	replica_parallel_workers
slave_pending_jobs_size_max	replica_pending_jobs_size_max
pseudo_slave_mode	pseudo_replica_mode

Les procédures enregistrées suivantes portent de nouveaux noms dans Aurora MySQL version 3.

Pour des raisons de compatibilité, vous pouvez utiliser l'un ou l'autre des noms dans la version initiale d'Aurora MySQL version 3. Les anciens noms de procédures seront supprimés dans une version ultérieure.

Nom à supprimer	Nom nouveau ou préféré
mysql.rds_set_master_auto_position	mysql.rds_set_source_auto_position
mysql.rds_set_external_master	mysql.rds_set_external_source
mysql.rds_set_external_master_with_auto_position	mysql.rds_set_external_source_with_auto_position

Nom à supprimer	Nom nouveau ou préféré
`mysql.rds_reset_external_master`	`mysql.rds_reset_external_source`
`mysql.rds_next_master_log`	`mysql.rds_next_source_log`

Valeurs AUTO_INCREMENT

Dans Aurora MySQL version 3, Aurora conserve la valeur `AUTO_INCREMENT` pour chaque table lorsqu'elle redémarre chaque instance de base de données. Dans Aurora MySQL version 2, la valeur `AUTO_INCREMENT` n'était pas conservée après un redémarrage.

La valeur `AUTO_INCREMENT` n'est pas conservée lorsque vous configurez un nouveau cluster en effectuant une restauration à partir d'un instantané ou une reprise ponctuelle et en clonant un cluster. Le cas échéant, la valeur `AUTO_INCREMENT` est initialisée en fonction de la valeur reposant sur la plus grande valeur de colonne de la table au moment de la création de l'instantané. Ce comportement est différent de celui de RDS pour MySQL 8.0, où la valeur `AUTO_INCREMENT` est conservée pendant ces opérations.

Réplication de journaux binaires

Dans la version 8.0 de MySQL Community Edition, la réplication des journaux binaires est activée par défaut. Dans Aurora MySQL version 3, la réplication des journaux binaires est désactivée par défaut.

> **Tip**
>
> Si vos exigences en matière de haute disponibilité sont satisfaites par les fonctions de réplication intégrées à Aurora, vous pouvez laisser la réplication des journaux binaires désactivée. De cette façon, vous pouvez éviter la surcharge de performances de la réplication des journaux binaires. Vous pouvez également éviter la surveillance et le dépannage associés nécessaires à la gestion de la réplication des journaux binaires.

Aurora prend en charge la réplication de journaux binaires depuis une source compatible MySQL 5.7 vers Aurora MySQL version 3. Le système source peut être un cluster de bases de données Aurora MySQL, une instance de base de données RDS pour MySQL ou une instance MySQL sur site.

Tout comme MySQL version communautaire, Aurora MySQL prend en charge la réplication depuis une source exécutant une version spécifique vers une cible exécutant la même version majeure ou une version majeure supérieure. Par exemple, la réplication depuis un système compatible MySQL 5.6 vers Aurora MySQL version 3 n'est pas prise en charge. La réplication depuis Aurora MySQL version 3 vers un système compatible MySQL 5.7 ou MySQL 5.6 n'est pas prise en charge. Pour plus d'informations sur l'utilisation de la réplication des journaux binaires, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

Aurora MySQL version 3 inclut des améliorations apportées à la réplication des journaux binaires dans MySQL 8.0 version communautaire, comme la réplication filtrée. Pour plus d'informations sur les améliorations apportées à MySQL 8.0 version communautaire, consultez Comment les serveurs évaluent les règles de filtrage de réplication dans le manuel de référence MySQL.

Compression des transactions pour la réplication des journaux binaires

Pour plus d'informations sur l'utilisation de la compression des journaux binaires, consultez Compression des transactions de journaux binairesdans le manuel de référence MySQL.

Les limitations suivantes s'appliquent à la compression des journaux binaires dans Aurora MySQL version 3 :

- Les transactions dont les données de journaux binaires sont supérieures à la taille de paquet maximale autorisée ne sont pas compressées. Ceci est vrai, que le paramètre de compression des journaux binaires Aurora MySQL soit activé ou non. Ces transactions sont répliquées sans être compressées.
- Si vous utilisez un connecteur CDC (Change Data Capture) qui ne prend pas encore en charge MySQL 8.0, vous ne pouvez pas utiliser cette fonction. Nous vous recommandons de tester minutieusement tous les connecteurs tiers avec une compression de journaux binaires. Nous vous recommandons également de le faire avant d'activer la compression des journaux binaires sur les systèmes qui utilisent la réplication des journaux binaires pour CDC.

Comparaison d'Aurora MySQL version 3 et de MySQL 8.0 Community Edition

Vous pouvez utiliser les informations suivantes pour en savoir plus sur les modifications à prendre en compte lorsque vous effectuez une conversion d'un autre système compatible MySQL 8.0 vers Aurora MySQL version 3.

En général, Aurora MySQL version 3 prend en charge l'ensemble de fonctions de MySQL 8.0.23 version communautaire. Certaines nouvelles fonctions de l'édition communautaire MySQL 8.0 ne s'appliquent pas à Aurora MySQL. Certaines de ces fonctions ne sont pas compatibles avec certains aspects d'Aurora, tels que l'architecture de stockage Aurora. Les autres fonctions ne sont pas nécessaires car le service de gestion Amazon RDS offre des fonctions équivalentes. Les fonctions suivantes de MySQL 8.0 version communautaire ne sont pas prises en charge ou fonctionnent différemment dans Aurora MySQL version 3.

Pour les notes de mise à jour de toutes les versions d'Aurora MySQL version 3, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 3 dans Notes de mise à jour d'Aurora MySQL.

Rubriques

Les fonctions MySQL 8.0 ne sont pas disponibles dans Aurora MySQL version 3

Les fonctions suivantes de MySQL 8.0 version communautaire ne sont pas disponibles ou fonctionnent différemment dans Aurora MySQL version 3.

- Les groupes de ressources et les instructions SQL associées ne sont pas pris en charge dans Aurora MySQL.
- Aurora MySQL ne prend pas en charge les espaces de table d'annulation définis par l'utilisateur ni les instructions SQL associées, telles que `CREATE UNDO TABLESPACE`, `ALTER UNDO TABLESPACE ... SET INACTIVE` et `DROP UNDO TABLESPACE`.
- Aurora MySQL ne prend pas en charge la troncature des espaces de table d'annulation pour les versions d'Aurora MySQL inférieures à 3.06. Dans Aurora MySQL 3.06 et versions ultérieures, la troncature automatique des espaces de tables d'annulation est prise en charge.
- Le plug-in de validation de mot de passe est pris en charge.
- Vous ne pouvez pas modifier les paramètres des plug-ins MySQL, y compris les plugins de validation de mot de passe.
- Le plugin X n'est pas pris en charge.
- La réplication multisource n'est pas prise en charge.

Modèle de privilège basé sur les rôles

Avec Aurora MySQL version 3, vous ne pouvez pas modifier les tables dans la base de données `mysql` directement. En particulier, vous ne pouvez pas configurer d'utilisateurs en les insérant dans la table `mysql.user`. Au lieu de cela, vous utilisez des instructions SQL pour accorder des privilèges basés sur des rôles. Vous ne pouvez pas non plus créer d'autres types d'objets tels que des procédures stockées dans la base de données `mysql`. Vous pouvez toujours interroger les tables `mysql`. Si vous utilisez la réplication des journaux binaires, les modifications apportées directement aux tables `mysql` du cluster source ne sont pas répliquées sur le cluster cible.

Dans certains cas, votre application peut utiliser des raccourcis pour créer des utilisateurs ou d'autres objets en les insérant dans les tables `mysql`. Le cas échéant, modifiez le code de votre application pour utiliser les instructions correspondantes telles que `CREATE USER`. Si votre application crée des procédures stockées ou d'autres objets dans la base de données, utilisez plutôt une autre base de données `mysql`.

Pour exporter des métadonnées pour les utilisateurs de bases de données pendant la migration à partir d'une base de données MySQL externe, vous pouvez utiliser une commande MySQL Shell à la place de `mysqldump`. Pour plus d'informations, consultez Utilitaire de vidage d'instance, Utilitaire de vidage de schéma et Utilitaire de vidage de table.

Pour simplifier la gestion des autorisations pour de nombreux utilisateurs ou applications, vous pouvez utiliser l'instruction `CREATE ROLE` pour créer un rôle doté d'un ensemble d'autorisations. Vous pouvez ensuite utiliser les instructions `GRANT` et `SET ROLE`, et la fonction `current_role` pour attribuer des rôles à des utilisateurs ou des applications, changer le rôle actuel et vérifier les rôles en vigueur. Pour plus d'informations sur le système d'autorisations basé sur les rôles dans MySQL 8.0, consultez Utilisation de rôles dans le manuel de référence MySQL.

> **Important**
>
> Nous vous recommandons vivement de ne pas avoir recours au rôle d'utilisateur principal directement dans vos applications. Au lieu de cela, respectez la bonne pratique qui consiste à avoir recours à un utilisateur de base de données doté des privilèges minimum requis pour votre application.

Rubriques

- rds_superuser_role

- Utilisateur de vérification des privilèges pour la réplication des journaux binaires
- Rôles pour accéder à d'autres AWS services

rds_superuser_role

Aurora MySQL version 3 inclut un rôle spécial doté de tous les privilèges suivants. Ce rôle est nommé `rds_superuser_role`. Ce rôle est déjà accordé à l'utilisateur administratif principal de chaque cluster. Le rôle `rds_superuser_role` inclut les privilèges suivants pour tous les objets de base de données :

- `ALTER`
- `APPLICATION_PASSWORD_ADMIN`
- `ALTER ROUTINE`
- `CONNECTION_ADMIN`
- `CREATE`
- `CREATE ROLE`
- `CREATE ROUTINE`
- `CREATE TEMPORARY TABLES`
- `CREATE USER`
- `CREATE VIEW`
- `DELETE`
- `DROP`
- `DROP ROLE`
- `EVENT`
- `EXECUTE`
- `FLUSH_OPTIMIZER_COSTS` (Aurora MySQL versions 3.09 et ultérieures)
- `FLUSH_STATUS` (Aurora MySQL versions 3.09 et ultérieures)
- `FLUSH_TABLES` (Aurora MySQL versions 3.09 et ultérieures)
- `FLUSH_USER_RESOURCES` (Aurora MySQL versions 3.09 et ultérieures)
- `INDEX`
- `INSERT`

- LOCK TABLES
- PROCESS
- REFERENCES
- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- ROLE_ADMIN
- SET_USER_ID
- SELECT
- SHOW DATABASES
- SHOW_ROUTINE (Aurora MySQL versions 3.04 et ultérieures)
- SHOW VIEW
- TRIGGER
- UPDATE
- XA_RECOVER_ADMIN

La définition du rôle inclut également `WITH GRANT OPTION` afin qu'un utilisateur administratif puisse accorder ce rôle à d'autres utilisateurs. En particulier, l'administrateur doit accorder tous les privilèges nécessaires à la réplication des journaux binaires avec le cluster Aurora MySQL comme cible.

> **Tip**
>
> Pour voir tous les détails des autorisations, saisissez les instructions suivantes.
>
> ```
> SHOW GRANTS FOR rds_superuser_role@'%';
> SHOW GRANTS FOR name_of_administrative_user_for_your_cluster@'%';
> ```

Utilisateur de vérification des privilèges pour la réplication des journaux binaires

Aurora MySQL version 3 inclut un utilisateur de vérification des privilèges pour la réplication des journaux binaires (binlog), `rdsrepladmin_priv_checks_user`. Outre les privilèges de ce `rds_superuser_role`, cet utilisateur possède le privilège `replication_applier`.

Lorsque vous activez la réplication des journaux binaires en appelant la procédure `mysql.rds_start_replication` stockée, la réplication `rdsrepladmin_priv_checks_user` est créée.

L'utilisateur `rdsrepladmin_priv_checks_user@localhost` est un utilisateur réservé. Ne le modifiez pas.

Rôles pour accéder à d'autres AWS services

Aurora MySQL version 3 inclut des rôles que vous pouvez utiliser pour accéder à d'autres AWS services. Vous pouvez définir un grand nombre de ces rôles comme alternative à l'octroi de privilèges. Par exemple, vous définissez `GRANT AWS_LAMBDA_ACCESS TO` *`user`* plutôt que `GRANT INVOKE LAMBDA ON *.* TO` *`user`*. Pour les procédures d'accès à d'autres AWS services, voirIntégration d'Amazon Aurora MySQL avec d'autres services AWS. Aurora MySQL version 3 inclut les rôles suivants liés à l'accès à d'autres AWS services :

- `AWS_LAMBDA_ACCESS` : alternative au privilège `INVOKE LAMBDA`. Pour plus d'informations, consultez Appel d'une fonction Lambda à partir d'un cluster de bases de données Amazon Aurora MySQL.
- `AWS_LOAD_S3_ACCESS` : alternative au privilège `LOAD FROM S3`. Pour plus d'informations, consultez Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3.
- `AWS_SELECT_S3_ACCESS` : alternative au privilège `SELECT INTO S3`. Pour plus d'informations, consultez Enregistrement de données d'un cluster de bases de données Amazon Aurora MySQL dans des fichiers texte stockés dans un compartiment Amazon S3.
- `AWS_COMPREHEND_ACCESS` : alternative au privilège `INVOKE COMPREHEND`. Pour plus d'informations, consultez Autorisation d'accès aux utilisateurs de base de données pour le machine learning Aurora.
- `AWS_SAGEMAKER_ACCESS` : alternative au privilège `INVOKE SAGEMAKER`. Pour plus d'informations, consultez Autorisation d'accès aux utilisateurs de base de données pour le machine learning Aurora.
- `AWS_BEDROCK_ACCESS` : il n'existe aucun privilège `INVOKE` analogue pour Amazon Bedrock. Pour plus d'informations, consultez Autorisation d'accès aux utilisateurs de base de données pour le machine learning Aurora.

Lorsque vous accordez l'accès à l'aide de rôles dans Aurora MySQL version 3, vous activez également le rôle à l'aide de l'instruction `SET ROLE` *`role_name`* ou `SET ROLE ALL`.

L'exemple suivant montre comment procéder. Remplacez le nom de rôle approprié par `AWS_SELECT_S3_ACCESS`.

```
# Grant role to user.

mysql> GRANT AWS_SELECT_S3_ACCESS TO 'user'@'domain-or-ip-address'

# Check the current roles for your user. In this case, the AWS_SELECT_S3_ACCESS role
 has not been activated.
# Only the rds_superuser_role is currently in effect.
mysql> SELECT CURRENT_ROLE();
+--------------------------+
| CURRENT_ROLE()           |
+--------------------------+
| `rds_superuser_role`@`%` |
+--------------------------+
1 row in set (0.00 sec)

# Activate all roles associated with this user using SET ROLE.
# You can activate specific roles or all roles.
# In this case, the user only has 2 roles, so we specify ALL.
mysql> SET ROLE ALL;
Query OK, 0 rows affected (0.00 sec)

# Verify role is now active
mysql> SELECT CURRENT_ROLE();
+-----------------------------------------------------+
| CURRENT_ROLE()                                      |
+-----------------------------------------------------+
| `AWS_SELECT_S3_ACCESS`@`%`,`rds_superuser_role`@`%` |
+-----------------------------------------------------+
```

Localisation de l'ID du serveur de base de données

L'ID du serveur de base de données (`server_id`) est requis pour la réplication des journaux binaires (binlog). La méthode permettant de trouver l'ID du serveur est différente dans Aurora MySQL et Community MySQL.

Dans Community MySQL, l'ID du serveur est un nombre, que vous pouvez obtenir en utilisant la syntaxe suivante lorsque vous êtes connecté au serveur :

```
mysql> select @@server_id;
```

```
+-------------+
| @@server_id |
+-------------+
| 2           |
+-------------+
1 row in set (0.00 sec)
```

Dans Aurora MySQL, l'ID du serveur est l'ID de l'instance de base de données, que vous obtenez en utilisant la syntaxe suivante lorsque vous êtes connecté à l'instance de base de données :

```
mysql> select @@aurora_server_id;

+-------------------------+
| @@aurora_server_id      |
+-------------------------+
| mydbcluster-instance-2 |
+-------------------------+
1 row in set (0.00 sec)
```

Pour plus d'informations sur la réplication des journaux binaires, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

Authentification

Dans MySQL 8.0 version communautaire, le plugin d'authentification par défaut est `caching_sha2_password`. Aurora MySQL version 3 utilise toujours le plugin `mysql_native_password`. Vous ne pouvez pas modifier le paramètre `default_authentication_plugin`. Vous pouvez toutefois créer des utilisateurs et modifier les utilisateurs actuels pour que leur mot de passe individuel utilise le nouveau plugin d'authentification. Voici un exemple.

```
mysql> CREATE USER 'testnewsha'@'%' IDENTIFIED WITH caching_sha2_password BY
 'aNewShaPassword';
Query OK, 0 rows affected (0.74 sec)
```

Mise à niveau vers Aurora MySQL version 3

Pour en savoir plus sur la mise à niveau de votre base de données Aurora MySQL version 2 vers la version 3, consultez Mise à niveau de la version majeure d'un cluster de bases de données Amazon Aurora MySQL.

Aurora MySQL version 2 compatible avec MySQL 5.7

Cette rubrique décrit les différences entre Aurora MySQL version 2 et MySQL 5.7 Community Edition.

> **Important**
>
> La version 2 d'Aurora MySQL a atteint sa fin de support standard le 31 octobre 2024. Pour plus d'informations, consultez Préparation à la fin du support standard de l'Édition compatible d'Amazon Aurora MySQL version 2.

Fonctions non prises en charge dans Aurora MySQL version 2

Les fonctions suivantes sont prises en charge dans MySQL 5.7, mais ne le sont actuellement pas dans Aurora MySQL version 2 :

- Instruction SQL `CREATE TABLESPACE`
- plugin de réplication de groupe
- Augmentation de la taille de page
- Chargement du pool de mémoires tampons InnoDB au démarrage
- plugin d'analyse de texte intégral InnoDB
- Réplication multi-source
- Redimensionnement de pool de mémoires tampons en ligne
- Plugin de validation de mot de passe – Vous pouvez installer le plugin, mais il n'est pas pris en charge. Vous ne pouvez pas personnaliser le plugin.
- plugins de réécriture de requête
- Filtrage de réplication
- Protocole X

Pour plus d'informations sur ces fonctions, consultez la documentation MySQL 5.7.

Comportement d'espace de table temporaire dans Aurora MySQL version 2

Dans MySQL 5.7, l'espace de table temporaire s'étend automatiquement et sa taille augmente au besoin pour accueillir les tables temporaires sur disque. Lorsque des tables temporaires sont supprimées, l'espace libéré peut être réutilisé pour de nouvelles tables temporaires, mais l'espace

de table temporaire conserve sa taille étendue et ne diminue pas. L'espace de table temporaire est supprimé et recréé lorsque le moteur est redémarré.

Dans Aurora MySQL version 2, le comportement suivant s'applique :

- Pour les nouveaux clusters de bases de données Aurora MySQL créés avec les versions 2.10 et ultérieures, l'espace de table temporaire est supprimé et recréé lorsque vous redémarrez la base de données. Cela permet à la fonction de redimensionnement dynamique de récupérer l'espace de stockage.
- Pour les clusters de bases de données Aurora MySQL existants mis à niveau vers :
  - Versions 2.10 et ultérieures : l'espace de table temporaire est supprimé et recréé lorsque vous redémarrez la base de données. Cela permet à la fonction de redimensionnement dynamique de récupérer l'espace de stockage.
  - Version 2.09 : l'espace de table temporaire n'est pas supprimé lorsque vous redémarrez la base de données.

Vous pouvez vérifier la taille de l'espace de table temporaire sur votre cluster de bases de données Aurora MySQL version 2 en utilisant la requête suivante :

```
SELECT
    FILE_NAME,
    TABLESPACE_NAME,
    ROUND((TOTAL_EXTENTS * EXTENT_SIZE) / 1024 / 1024 / 1024, 4) AS SIZE
FROM
    INFORMATION_SCHEMA.FILES
WHERE
    TABLESPACE_NAME = 'innodb_temporary';
```

Pour plus d'informations, consultez The Temporary Tablespace (L'espace de table temporaire) dans la documentation MySQL.

Moteur de stockage pour des tables temporaires sur disque

Aurora MySQL version 2 utilise différents moteurs de stockage pour les tables temporaires internes sur disque en fonction du rôle de l'instance.

- Sur l'instance d'enregistreur, les tables temporaires sur disque utilisent le moteur de stockage InnoDB par défaut. Elles sont stockées dans l'espace de table temporaire du volume de cluster Aurora.

Vous pouvez modifier ce comportement sur l'instance d'enregistreur en modifiant la valeur du paramètre de base de données `internal_tmp_disk_storage_engine`. Pour plus d'informations, consultez Paramètres de niveau instance.

- Sur les instances de lecture, les tables temporaires sur disque utilisent le moteur de stockage MyISAM, qui utilise le stockage local. En effet, les instances en lecture seule ne peuvent stocker aucune donnée sur le volume de cluster Aurora.

Sécurité avec Amazon Aurora MySQL

La sécurité d'Amazon Aurora MySQL est gérée à trois niveaux :

- Pour contrôler les personnes autorisées à exécuter des opérations de gestion Amazon RDS sur des clusters et des instances de base de données Aurora MySQL, vous utilisez Gestion des identités et des accès AWS (IAM). Lorsque vous vous connectez à AWS à l'aide des informations d'identification IAM, votre compte AWS doit disposer des politiques IAM qui accordent les autorisations requises pour exécuter les opérations de gestion Amazon RDS. Pour plus d'informations, consultez Identity and Access Management pour Amazon Aurora.

  Si vous utilisez IAM pour accéder à la console Amazon RDS, connectez-vous d'abord à l'AWS Management Console avec vos informations d'identification IAM. Connectez-vous ensuite à la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

- Les clusters de base de données Aurora MySQL doivent être créés dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC. Pour contrôler les appareils et les instances Amazon EC2 qui peuvent ouvrir des connexions au point de terminaison et au port de l'instance de base de données pour les clusters de bases de données Aurora MySQL d'un VPC, utilisez un groupe de sécurité VPC. Vous pouvez établir ces connexions entre les points de terminaison et les ports en utilisant le protocole TLS (Transport Layer Security). En outre, les règles de pare-feu de votre entreprise peuvent contrôler si les appareils en cours d'exécution dans votre entreprise peuvent ouvrir des connexions à une instance de base de données. Pour plus d'informations sur les VPC, consultez Amazon VPC et Amazon Aurora.

  La location de VPC prise en charge dépend de la classe d'instance de base de données utilisée par vos clusters de bases de données Aurora MySQL. Avec la location de VPC `default`, le VPC s'exécute sur du matériel partagé. Avec la location de VPC `dedicated`, le VPC s'exécute sur une instance de matériel dédiée. Les classes d'instance de base de données à performances extensibles prennent uniquement en charge la location de VPC par défaut. Les classes d'instance de base de données à performances extensibles incluent les classes d'instance de base de données db.t2, db.t3 et db.t4g. Toutes les autres classes d'instance de base de données MySQL Aurora prennent en charge à la fois la location de VPC par défaut et dédiée.

  > **Note**
  >
  > Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à

la production. Pour plus de détails sur les classes d'instance T, consultez Utilisation de classes d'instance T pour le développement et les tests.

Pour plus d'informations sur les classes d'instance, consultez Classes d'instance de base de données Amazon Aurora. Pour plus d'informations sur la location de VPC `default` et `dedicated`, consultez Instances dédiées dans le Guide de l'utilisateur Amazon Elastic Compute Cloud.

- Pour authentifier la connexion et les autorisations d'un cluster de bases de données Amazon Aurora MySQL, vous pouvez adopter l'une des approches suivantes, ou les combiner :
  - Vous pouvez adopter la même approche qu'avec une instance autonome de MySQL.

    Les commandes telles que `CREATE USER`, `RENAME USER`, `GRANT`, `REVOKE` et `SET PASSWORD` fonctionnent de la même façon que dans les bases de données sur site, comme le fait la modification directe des tables du schéma de base de données. Pour plus d'informations, consultez Contrôle d'accès et gestion des comptes dans la documentation MySQL.
  - Vous pouvez également utiliser l'authentification de base de données IAM.

    L'authentification de base de données IAM vous permet de vous authentifier sur votre cluster de bases de données à l'aide d'un utilisateur IAM ou d'un rôle IAM et d'un jeton d'authentification. Un jeton d'authentification est une valeur unique qui est générée à l'aide du processus de signature Signature Version 4. L'authentification de base de données IAM vous permet d'utiliser les mêmes informations d'identification pour contrôler l'accès à vos ressources AWS et à vos bases de données. Pour plus d'informations, consultez Authentification de base de données IAM.

> **Note**
>
> Pour plus d'informations, consultez Sécurité dans Amazon Aurora.

Dans les sections suivantes, consultez les informations sur les autorisations utilisateur pour les connexions Aurora MySQL et TLS avec les clusters de bases de données Aurora MySQL.

Rubriques

- Privilèges d'utilisateur principal avec Amazon Aurora MySQL.
- Connexions TLS aux clusters de bases de données Aurora MySQL

Privilèges d’utilisateur principal avec Amazon Aurora MySQL.

Lorsque vous créez une instance de base de données MySQL Amazon Aurora, l’utilisateur principal dispose des privilèges par défaut répertoriés dans Privilèges du compte utilisateur principal.

Pour fournir des services de gestion pour chaque cluster de bases de données, les utilisateurs `admin` et `rdsadmin` sont créés lors de la création du cluster de bases de données. Les tentatives de supprimer, renommer et modifier le mot de passe du compte `rdsadmin`, ou d’en modifier les privilèges, génèrent une erreur.

Dans les clusters de bases de données Aurora MySQL version 2, les utilisateurs `admin` et `rdsadmin` sont créés lors de la création du cluster de bases de données. Dans les clusters de bases de données Aurora MySQL version 3, les utilisateurs `admin`, `rdsadmin` et `rds_superuser_role` sont créés.

> **Important**
>
> Nous vous recommandons vivement de ne pas avoir recours au rôle d’utilisateur principal directement dans vos applications. Au lieu de cela, respectez la bonne pratique qui consiste à avoir recours à un utilisateur de base de données doté des privilèges minimum requis pour votre application.

Pour la gestion du cluster de bases de données Aurora MySQL, les commandes standard `kill` et `kill_query` ont fait l’objet de restrictions. Utilisez à la place les commandes Amazon RDS `rds_kill` et `rds_kill_query` pour arrêter les sessions utilisateur ou les requêtes sur les instances de base de données Aurora MySQL.

> **Note**
>
> Le chiffrement d’une instance de base de données et des instantanés n’est pas pris en charge pour la région Chine (Ningxia).

Connexions TLS aux clusters de bases de données Aurora MySQL

Les clusters de bases de données Amazon Aurora MySQL prennent en charge les connexions TLS (Transport Layer Security) à partir des applications utilisant le même processus et la même clé publique que les instances de base de données RDS for MySQL.

Amazon RDS crée un certificat TLS et l'installe sur l'instance de base de données quand Amazon RDS met en service l'instance. Ces certificats sont signés par une autorité de certification. Le certificat TLS inclut le point de terminaison de l'instance de base de données en tant que nom commun (CN) du certificat TLS pour assurer une protection contre les attaques par usurpation. Par conséquent, vous pouvez utiliser uniquement le point de terminaison de cluster de bases de données pour vous connecter à un cluster de bases de données à l'aide de TLS, si votre client prend en charge les noms SAN (Subject Alternative Names). Sinon, vous devez utilisez le point de terminaison d'instance d'une instance de dispositif d'écriture.

Pour plus d'informations sur le téléchargement de certificats, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.

Nous recommandons d'utiliser le pilote AWS JDBC comme client prenant en charge SAN avec TLS. Pour en savoir plus sur le pilote JDBC AWS et pour obtenir des instructions d'utilisation complètes, consultez le référentiel GitHub du pilote JDBC Amazon Web Services (AWS).

Rubriques

- Exiger une connexion TLS à un cluster de bases de données Aurora MySQL
- Versions TLS pour Aurora MySQL
- Configuration de suites de chiffrement pour les connexions aux clusters de bases de données Aurora MySQL
- Chiffrement des connexions à un cluster de bases de données Aurora MySQL

Exiger une connexion TLS à un cluster de bases de données Aurora MySQL

Vous pouvez exiger que toutes les connexions utilisateur à votre cluster de bases de données Aurora MySQL utilisent TLS à l'aide du paramètre de cluster de bases de données `require_secure_transport`. Par défaut, le paramètre `require_secure_transport` est défini sur `OFF`. Vous pouvez définir le paramètre `require_secure_transport` sur `ON` pour exiger TLS pour les connexions à votre cluster de bases de données.

Vous pouvez définir la valeur du paramètre `require_secure_transport` en mettant à jour le groupe de paramètres pour votre cluster de bases de données. Vous n'avez pas besoin de redémarrer votre cluster de bases de données pour que la modification prenne effet. Pour plus d'informations sur les groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora.

> **Note**
>
> Le paramètre `require_secure_transport` est disponible pour Aurora MySQL versions 2 et 3. Vous pouvez définir ce paramètre dans un groupe de paramètres de cluster de bases de données personnalisé. Le paramètre n'est pas disponible dans les groupes de paramètres d'instance de base de données.

Lorsque le paramètre `require_secure_transport` est défini sur `ON` pour un cluster de bases de données, un client de base de données peut s'y connecter s'il peut établir une connexion chiffrée. Sinon, un message d'erreur similaire au suivant est renvoyé au client :

```
MySQL Error 3159 (HY000): Connections using insecure transport are prohibited while --
require_secure_transport=ON.
```

Versions TLS pour Aurora MySQL

Aurora MySQL prend en charge le protocole TLS (Transport Layer Security) versions 1.0, 1.1, 1.2 et 1.3. À partir d'Aurora MySQL version 3.04.0, vous pouvez utiliser le protocole TLS 1.3 pour sécuriser vos connexions. Le tableau suivant affiche la prise en charge du protocole TLS pour les versions Aurora MySQL.

Aurora MySQL Version	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Par défaut
Aurora MySQL version 2	Obsolète	Obsolète	Pris en charge	Non pris en charge	Toutes les versions TLS prises en charge
Aurora MySQL version 3 (antérieure à 3.04.0)	Obsolète	Obsolète	Pris en charge	Non pris en charge	Toutes les versions TLS prises en charge

Aurora MySQL Version	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3	Par défaut
Aurora MySQL version 3 (3.04.0 et versions ultérieures)	Non pris en charge	Non pris en charge	Pris en charge	Pris en charge	Toutes les versions TLS prises en charge

> **Important**
>
> Si vous utilisez des groupes de paramètres personnalisés pour vos clusters Aurora MySQL de version 2 ou de version antérieure à 3.04.0, nous vous recommandons d'utiliser TLS 1.2, car les protocoles TLS 1.0 et 1.1 sont moins sécurisés. L'édition communautaire de MySQL 8.0.26 et Aurora MySQL 3.03 et ses versions mineures ont rendu obsolète la prise en charge de TLS versions 1.1 et 1.0.
>
> L'édition communautaire de MySQL 8.0.28 et les versions 3.04.0 et ultérieures compatibles d'Aurora MySQL ne prennent pas en charge TLS 1.1 ni TLS 1.0. Si vous utilisez Aurora MySQL 3.04.0 ou version ultérieure, ne définissez pas le protocole TLS sur 1.0 ni 1.1 dans votre groupe de paramètres personnalisés.
>
> Pour Aurora MySQL version 3.04.0 ou ultérieure, les paramètres par défaut sont TLS 1.3 et TLS 1.2.

Vous pouvez utiliser le paramètre de cluster de bases de données `tls_version` pour indiquer les versions de protocole autorisées. Des paramètres client similaires existent pour la plupart des outils client ou des pilotes de base de données. Certains clients plus anciens peuvent ne pas prendre en charge les versions TLS plus récentes. Par défaut, le cluster de bases de données tente d'utiliser la version de protocole TLS la plus élevée autorisée par la configuration du serveur et du client.

Définissez le paramètre de cluster de bases de données `tls_version` sur l'une des valeurs suivantes :

- `TLSv1.3`

- `TLSv1.2`
- `TLSv1.1`
- `TLSv1`

Vous pouvez également définir le paramètre `tls_version` sous forme de chaîne de liste séparée par des virgules. Si vous souhaitez utiliser à la fois les protocoles TLS 1.2 et TLS 1.3, le paramètre `tls_version` doit inclure tous les protocoles, du plus bas au plus élevé. Dans ce cas, `tls_version` est défini comme suit :

```
tls_version=TLSv1.2,TLSv1.3
```

Pour plus d'informations sur la modification de paramètres dans un groupe de paramètres de cluster de bases de données, consultez Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora. Si vous utilisez l'AWS CLI pour modifier le `tls_version` groupe de paramètres de cluster de bases de données, `ApplyMethod` doit avoir la valeur `pending-reboot`. Lorsque la méthode d'application est`pending-reboot`, les modifications des paramètres sont appliquées après l'arrêt et le redémarrage des clusters de bases de données associés au groupe de paramètres.

Configuration de suites de chiffrement pour les connexions aux clusters de bases de données Aurora MySQL

L'utilisation de suites de chiffrement configurables vous permet d'avoir plus de contrôle sur la sécurité des connexions de vos bases de données. Vous pouvez spécifier une liste de suites de chiffrement que vous souhaitez autoriser pour la sécurisation des connexions TLS client à votre base de données. Avec les suites de chiffrement configurables, vous pouvez contrôler le chiffrement de connexion accepté par votre serveur de base de données. Cela permet d'éviter l'utilisation de codes de chiffrement non sécurisés ou rendus obsolètes.

Les suites de chiffrement configurables sont prises en charge dans Aurora MySQL version 3 et Aurora MySQL version 2. Pour spécifier la liste des chiffrements TLS 1.2, TLS 1.1 et TLS 1.0 autorisés pour le chiffrement des connexions, modifiez le paramètre de cluster `ssl_cipher`. Définissez le paramètre `ssl_cipher` dans un groupe de paramètres de cluster utilisant l'AWS Management Console, l'AWS CLI, ou l'API RDS.

Définissez le paramètre `ssl_cipher` comme une chaîne de valeurs de chiffrement séparées par des virgules pour votre version TLS. Pour l'application client, vous pouvez spécifier les chiffrements

à utiliser pour les connexions chiffrées en utilisant l'option `--ssl-cipher` lors de la connexion à la base de données. Pour obtenir plus d'informations sur la connexion à votre base de données, consultez Connexion à un cluster de bases de données Amazon Aurora MySQL.

À partir d'Aurora MySQL version 3.04.0, vous pouvez spécifier des suites de chiffrement TLS 1.3. Pour spécifier les suites de chiffrement TLS 1.3 autorisées, modifiez le paramètre `tls_ciphersuites` de votre groupe de paramètres. TLS 1.3 a réduit le nombre de suites de chiffrement disponibles en raison de modifications apportées à la convention de dénomination qui supprime le mécanisme d'échange de clés et le certificat utilisés. Définissez le paramètre `tls_ciphersuites` comme une chaîne de valeurs de chiffrement séparées par des virgules pour TLS 1.3.

Le tableau suivant présente les chiffrements pris en charge ainsi que le protocole de chiffrement TLS et les versions valides du moteur Aurora MySQL pour chaque chiffrement.

Chiffrement	Protocole de chiffrement	Versions Aurora MySQL prises en charge
`ECDHE-RSA-AES128-SHA`	TLS 1.0	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures
`ECDHE-RSA-AES128-SHA256`	TLS 1.2	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures
`ECDHE-RSA-AES128-GCM-SHA256`	TLS 1.2	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures
`ECDHE-RSA-AES256-SHA`	TLS 1.0	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures
`ECDHE-RSA-AES256-GCM-SHA384`	TLS 1.2	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures
`ECDHE-RSA-CHACHA20-POLY1305`	TLS 1.2	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures
`ECDHE-ECDSA-AES128-SHA`	TLS 1.0	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures

Chiffrement	Protocole de chiffrement	Versions Aurora MySQL prises en charge
`ECDHE-ECDSA-AES256-SHA`	TLS 1.0	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures
`ECDHE-ECDSA-AES128-GCM-SHA256`	TLS 1.2	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures
`ECDHE-ECDSA-AES256-GCM-SHA384`	TLS 1.2	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures
`ECDHE-ECDSA-CHACHA20-POLY1305`	TLS 1.2	3.04.0 et versions ultérieures, 2.11.0 et versions ultérieures
`TLS_AES_128_GCM_SHA256`	TLS 1.3	Versions 3.04.0 et ultérieures
`TLS_AES_256_GCM_SHA384`	TLS 1.3	Versions 3.04.0 et ultérieures
`TLS_CHACHA20_POLY1305_SHA256`	TLS 1.3	Versions 3.04.0 et ultérieures

Pour plus d'informations sur la modification de paramètres dans un groupe de paramètres de cluster de bases de données, consultez Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora. Si vous utilisez l'interface CLI pour modifier le paramètre `ssl_cipher` du cluster de bases de données, veillez à définir le paramètre `ApplyMethod` sur `pending-reboot`. Lorsque la méthode d'application est`pending-reboot`, les modifications des paramètres sont appliquées après l'arrêt et le redémarrage des clusters de bases de données associés au groupe de paramètres.

Vous pouvez également utiliser la commande CLI describe-engine-default-cluster-parameters pour déterminer quelles suites de chiffrement sont actuellement prises en charge pour une famille de groupes de paramètres spécifique. L'exemple suivant montre comment obtenir les valeurs autorisées pour le paramètre de cluster `ssl_cipher` pour Aurora MySQL version 2.

```
aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-
mysql5.7

        ...some output truncated...
 {
        "ParameterName": "ssl_cipher",
        "ParameterValue": "ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-
AES128-GCM-SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-CHACHA20-
POLY1305,ECDHE-ECDSA-AES256-SHA,ECDHE-ECDSA-CHACHA20-POLY1305,ECDHE-ECDSA-AES256-GCM-
SHA384,ECDHE-ECDSA-AES128-GCM-SHA256,ECDHE-ECDSA-AES128-SHA",
        "Description": "The list of permissible ciphers for connection encryption.",
        "Source": "system",
        "ApplyType": "static",
        "DataType": "list",
        "AllowedValues": "ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-
AES128-GCM-SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-GCM-SHA384,ECDHE-RSA-CHACHA20-
POLY1305,ECDHE-ECDSA-AES256-SHA,ECDHE-ECDSA-CHACHA20-POLY1305,ECDHE-ECDSA-AES256-GCM-
SHA384,ECDHE-ECDSA-AES128-GCM-SHA256,ECDHE-ECDSA-AES128-SHA",
        "IsModifiable": true,
        "SupportedEngineModes": [
            "provisioned"
        ]
    },
       ...some output truncated...
```

Pour obtenir plus d'informations sur les chiffrements, consultez la variable ssl_cipher dans la documentation MySQL. Pour plus d'informations sur les formats de suite de chiffrement, consultez les documentations relatives aux format de liste openssl-ciphers et format de chaîne openssl-ciphers sur le site Web d'OpenSSL.

Chiffrement des connexions à un cluster de bases de données Aurora MySQL

Pour chiffrer les connexions à l'aide du client `mysql` par défaut, lancez le client mysql à l'aide du paramètre `--ssl-ca` pour référencer la clé publique. Par exemple :

Pour MySQL 5.7 et 8.0 :

```
mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com
--ssl-ca=full_path_to_CA_certificate --ssl-mode=VERIFY_IDENTITY
```

Pour MySQL 5.6 :

```
mysql -h myinstance.123456789012.rds-us-east-1.amazonaws.com
--ssl-ca=full_path_to_CA_certificate --ssl-verify-server-cert
```

Remplacez *full_path_to_CA_certificate* par le chemin complet vers votre certificat d'une autorité de certification (CA). Pour plus d'informations sur le téléchargement d'un certificat, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.

Vous pouvez exiger des connexions TLS pour des comptes d'utilisateur spécifiques. Par exemple, vous pouvez utiliser l'une des instructions suivantes, selon votre version de MySQL, pour exiger des connexions TLS sur le compte d'utilisateur `encrypted_user`.

Pour MySQL 5.7 et 8.0 :

```
ALTER USER 'encrypted_user'@'%' REQUIRE SSL;
```

Pour MySQL 5.6 :

```
GRANT USAGE ON *.* TO 'encrypted_user'@'%' REQUIRE SSL;
```

Lorsque vous utilisez un proxy RDS, vous vous connectez au point de terminaison du proxy et non au point de terminaison de cluster habituel. Vous pouvez rendre le certificat SSL/TLS obligatoire ou facultatif pour les connexions au proxy, comme pour les connexions directes au cluster de bases de données Aurora. Pour obtenir des informations sur l'utilisation du proxy RDS, consultez Proxy Amazon RDS pour Aurora.

> **Note**
>
> Pour plus d'informations sur les connexions TLS avec MySQL, consultez la documentation MySQL.

Mise à jour des applications pour se connecter aux clusters de bases de données Aurora MySQL à l'aide des nouveaux certificats TLS

Le 13 janvier 2023, Amazon RDS a publié de nouveaux certificats d'autorité de certification (CA) pour une connexion à vos clusters de bases de données Aurora à l'aide du protocole TLS (Transport Layer Security). Vous trouverez ci-après des informations sur la mise à jour de vos applications afin d'utiliser les nouveaux certificats.

Cette rubrique peut vous aider à déterminer si des applications clientes utilisent le protocole TLS pour se connecter à vos clusters de bases de données. Si tel est le cas, il vous est alors possible de vérifier si ces applications nécessitent une vérification du certificat pour se connecter.

> **Note**
>
> Certaines applications sont configurées pour ne se connecter aux clusters de bases de données Aurora MySQL que si la vérification du certificat sur le serveur s'effectue avec succès.
> Pour ces applications, vous devez mettre à jour les magasins d'approbations des applications clientes afin d'inclure les nouveaux certificats de l'autorité de certification.

Une fois que vous avez mis à jour les certificats de l'autorité de certification dans les magasins d'approbations des applications clientes, vous pouvez soumettre les certificats de vos clusters de bases de données à une rotation. Nous vous recommandons vivement de tester ces procédures dans un environnement de développement ou intermédiaire avant de les implémenter dans vos environnements de production.

Pour de plus amples informations sur la rotation de certificats, veuillez consulter Rotation de votre certificat SSL/TLS. Pour en savoir plus sur le téléchargement de certificats, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters. Pour obtenir des informations sur l'utilisation du protocole TLS avec les clusters de bases de données Aurora MySQL, consultez Connexions TLS aux clusters de bases de données Aurora MySQL.

Rubriques

- Contrôle de la nécessité d'une vérification du certificat du client pour qu'il puisse se connecter
- Mise à jour du magasin d'approbations de votre application
- Exemple de code Java pour l'établissement de connexions TLS

Déterminer si des applications se connectent à votre cluster de bases de données Aurora MySQL via le protocole TLS

Si vous utilisez Aurora MySQL version 2 (compatible avec MySQL 5.7) et que le schéma de performance est activé, exécutez la requête suivante pour vérifier si les connexions utilisent le protocole TLS. Pour de plus amples informations sur l'activation du schéma de performance, veuillez consulter Performance Schema Quick Start dans la documentation MySQL.

```
mysql> SELECT id, user, host, connection_type
       FROM performance_schema.threads pst
       INNER JOIN information_schema.processlist isp
       ON pst.processlist_id = isp.id;
```

Dans cet exemple de sortie, vous pouvez voir que votre propre session (`admin`) et une application connectée sous le nom de `webapp1` utilisent toutes deux le protocole TLS.

```
+----+-----------------+------------------+-----------------+
| id | user            | host             | connection_type |
+----+-----------------+------------------+-----------------+
|  8 | admin           | 10.0.4.249:42590 | SSL/TLS         |
|  4 | event_scheduler | localhost        | NULL            |
| 10 | webapp1         | 159.28.1.1:42189 | SSL/TLS         |
+----+-----------------+------------------+-----------------+
3 rows in set (0.00 sec)
```

Contrôle de la nécessité d'une vérification du certificat du client pour qu'il puisse se connecter

Vous pouvez vérifier si les clients JDBC et les clients MySQL requièrent une vérification du certificat pour pouvoir se connecter.

JDBC

L'exemple suivant avec MySQL Connector/J 8.0 illustre une façon de vérifier les propriétés de connexion JDBC d'une application afin de déterminer si les connexions nécessitent un certificat valide pour réussir. Pour de plus amples informations sur l'ensemble des options de connexion JDBC pour MySQL, veuillez consulter Configuration Properties dans la documentation MySQL.

Lorsque vous utilisez MySQL Connector/J 8.0, une connexion TLS nécessite la vérification du certificat de l'autorité de certification sur le serveur si vos propriétés de connexion ont `sslMode` défini sur `VERIFY_CA` ou `VERIFY_IDENTITY`, comme illustré dans l'exemple suivant.

```
Properties properties = new Properties();
properties.setProperty("sslMode", "VERIFY_IDENTITY");
properties.put("user", DB_USER);
properties.put("password", DB_PASSWORD);
```

> **Note**
>
> Si vous utilisez MySQL Java Connector v5.1.38 ou version ultérieure, ou MySQL Java Connector v8.0.9 ou version ultérieure, pour vous connecter à vos bases de données, même si vous n'avez pas explicitement configuré vos applications de manière à utiliser TLS lors de la connexion à vos bases de données, ces pilotes clients utilisent par défaut TLS. En outre, lors de l'utilisation de TLS, ils effectuent une vérification partielle du certificat et ne parviennent pas à se connecter si le certificat du serveur de base de données a expiré.

MySQL

Les exemples suivants avec le client MySQL montrent deux façons de vérifier la connexion MySQL d'un script pour déterminer si les connexions nécessitent un certificat valide pour réussir. Pour de plus amples informations sur l'ensemble des options de connexion avec le client MySQL, veuillez consulter Client-Side Configuration for Encrypted Connections dans la documentation MySQL.

Lorsque vous utilisez le client MySQL 5.7 ou MySQL 8.0, une connexion TLS nécessite la vérification du certificat de l'autorité de certification sur le serveur si, pour l'option `--ssl-mode`, vous spécifiez `VERIFY_CA` ou `VERIFY_IDENTITY`, comme dans l'exemple suivant.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-mode=VERIFY_CA
```

Lorsque vous utilisez le client MySQL 5.6, une connexion SSL nécessite la vérification du certificat de l'autorité de certification sur le serveur si vous spécifiez l'option `--ssl-verify-server-cert`, comme illustré dans l'exemple suivant.

```
mysql -h mysql-database.rds.amazonaws.com -uadmin -ppassword --ssl-ca=/tmp/ssl-cert.pem
 --ssl-verify-server-cert
```

Mise à jour du magasin d'approbations de votre application

Pour de plus amples informations sur la mise à jour du magasin d'approbations des applications MySQL, veuillez consulter Installing SSL Certificates dans la documentation MySQL.

> **Note**
>
> Lors de la mise à jour du magasin d'approbations, vous pouvez conserver les certificats plus anciens en complément de l'ajout des nouveaux certificats.

Mise à jour du magasin d'approbations de votre application pour JDBC

Vous pouvez mettre à jour le magasin d'approbations pour les applications qui utilisent JDBC dans le cadre des connexions TLS.

Pour plus d'informations sur le téléchargement du certificat racine, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.

Pour obtenir des exemples de scripts qui importent des certificats, consultez Exemple de script pour importer les certificats dans votre magasin d'approbations.

Si vous utilisez le pilote JDBC mysql dans une application, définissez les propriétés suivantes dans l'application.

```
System.setProperty("javax.net.ssl.trustStore", certs);
System.setProperty("javax.net.ssl.trustStorePassword", "password");
```

> **Note**
>
> Spécifiez un mot de passe autre que celui indiqué ici, en tant que bonne pratique de sécurité.

Lorsque vous démarrez l'application, définissez les propriétés suivantes.

```
java -Djavax.net.ssl.trustStore=/path_to_truststore/MyTruststore.jks -
Djavax.net.ssl.trustStorePassword=my_truststore_password com.companyName.MyApplication
```

Exemple de code Java pour l'établissement de connexions TLS

L'exemple de code suivant montre comment configurer la connexion SSL qui valide le certificat sur le serveur à l'aide de JDBC.

```
public class MySQLSSLTest {

        private static final String DB_USER = "user name";
        private static final String DB_PASSWORD = "password";
        // This key store has only the prod root ca.
        private static final String KEY_STORE_FILE_PATH = "file-path-to-keystore";
        private static final String KEY_STORE_PASS = "keystore-password";

    public static void test(String[] args) throws Exception {
        Class.forName("com.mysql.jdbc.Driver");


        System.setProperty("javax.net.ssl.trustStore", KEY_STORE_FILE_PATH);
        System.setProperty("javax.net.ssl.trustStorePassword", KEY_STORE_PASS);

        Properties properties = new Properties();
        properties.setProperty("sslMode", "VERIFY_IDENTITY");
        properties.put("user", DB_USER);
        properties.put("password", DB_PASSWORD);

        Connection connection = DriverManager.getConnection("jdbc:mysql://jagdeeps-ssl-
test.cni62e2e7kwh.us-east-1.rds.amazonaws.com:3306",properties);
        Statement stmt=connection.createStatement();

        ResultSet rs=stmt.executeQuery("SELECT 1 from dual");

        return;
    }
}
```

Important

Une fois que vous avez déterminé que vos connexions de base de données utilisent TLS et que vous avez mis à jour le magasin d'approbations de votre application, vous pouvez mettre à jour votre base de données pour utiliser les certificats rds-ca-rsa2048-g1. Pour obtenir des instructions, veuillez consulter l'étape 3 dans Mise à jour de votre certificat CA en modifiant votre instance de base de données .

Utilisation de l'authentification Kerberos pour Aurora MySQL

Vous pouvez utiliser l'authentification Kerberos pour authentifier les utilisateurs lorsqu'ils se connectent à votre cluster de bases de données Aurora MySQL. Pour ce faire, vous configurez votre cluster de bases de données afin qu'il utilise AWS Directory Service for Microsoft Active Directory pour l'authentification Kerberos. AWS Directory Service for Microsoft Active Directory est également appelé AWS Managed Microsoft AD. Cette fonction est disponible avec Directory Service. Pour en savoir plus, consultez Qu'est-ce qu'Directory Service ? dans le Guide d'administration AWS Directory Service.

Pour démarrer, créez un annuaire AWS Managed Microsoft AD pour stocker les informations d'identification utilisateur. Fournissez ensuite à votre cluster de bases de données Aurora MySQL le domaine de l'annuaire Active Directory ainsi que d'autres informations. Lorsque les utilisateurs s'authentifient auprès de l'instance de cluster de bases de données Aurora MySQL, les demandes d'authentification sont transférées vers l'annuaire AWS Managed Microsoft AD.

Vous pouvez gagner du temps et de l'argent en conservant toutes les informations d'identification dans le même annuaire. Cette approche vous permet d'avoir un endroit centralisé de stockage et de gestion des informations d'identification pour plusieurs clusters de bases de données. L'utilisation d'un annuaire peut également améliorer votre profil de sécurité global.

Vous pouvez également accéder aux informations d'identification à partir de votre propre annuaire Microsoft Active Directory sur site. Pour ce faire, vous créez une relation de domaine d'approbation afin que l'annuaire AWS Managed Microsoft AD approuve votre annuaire Microsoft Active Directory sur site. De cette façon, vos utilisateurs peuvent accéder à vos clusters de bases de données Aurora MySQL avec la même expérience d'authentification unique (SSO) Windows que lorsqu'ils accèdent aux charges de travail de votre réseau sur site.

Une base de données peut utiliser Kerberos, Gestion des identités et des accès AWS (IAM), ou à la fois l'authentification Kerberos et IAM. Toutefois, comme les authentifications Kerberos et IAM fournissent des méthodes d'authentification différentes, un utilisateur spécifique peut se connecter à une base de données en utilisant uniquement l'une ou l'autre méthode d'authentification, mais pas les deux. Pour plus d'informations sur l'authentification IAM, veuillez consulter Authentification de base de données IAM.

Table des matières

Présentation de l'authentification Kerberos pour les clusters de bases de données Aurora MySQL

Pour configurer l'authentification Kerberos pour un cluster de bases de données Aurora MySQL, effectuez les étapes générales suivantes. Ces étapes sont décrites plus en détail ci-dessous.

1. Utilisez AWS Managed Microsoft AD pour créer un annuaire AWS Managed Microsoft AD. Vous pouvez utiliser AWS Management Console, AWS CLI ou l'Directory Service pour créer l'annuaire. Pour obtenir des instructions détaillées, consultez Création d'un annuaire AWS Managed Microsoft AD dans le Guide d'administration AWS Directory Service.
2. Créez un rôle Gestion des identités et des accès AWS (IAM) utilisant la politique IAM gérée `AmazonRDSDirectoryServiceAccess`. Le rôle autorise Amazon Aurora à effectuer des appels vers votre annuaire.

Pour que le rôle autorise l'accès, le point de terminaison AWS Security Token Service (AWS STS) doit être activé dans la Région AWS pour votre compte AWS. Les points de terminaison AWS STS sont actifs par défaut dans toutes les Régions AWS et vous pouvez les utiliser sans qu'aucune autre action soit nécessaire. Pour de plus amples informations, veuillez consulter Activation et désactivation d'AWS STS dans une Région AWS dans le Guide de l'utilisateur IAM.

3. Créez et configurez les utilisateurs dans l'annuaire AWS Managed Microsoft AD à l'aide des outils Microsoft Active Directory. Pour plus d'informations sur la création d'utilisateurs dans votre annuaire Active Directory, consultez Gérer les utilisateurs et les groupes dans Microsoft AD géré par AWS dans le Guide d'administration AWS Directory Service.
4. Créez ou modifiez un cluster de bases de données Aurora MySQL. Si vous utilisez CLI ou l'API RDS dans la demande de création, spécifiez un identificateur de domaine avec le paramètre `Domain`. Utilisez l'identificateur `d-*` généré lors de la création de votre annuaire et le nom du rôle IAM que vous avez créé.

   Si vous modifiez un cluster de bases de données Aurora MySQL existante pour utiliser l'authentification Kerberos, définissez les paramètres de domaine et de rôle IAM pour le cluster de bases de données. Recherchez le cluster de bases de données dans le même VPC que l'annuaire du domaine.
5. Utilisez les informations d'identification de l'utilisateur principal Amazon RDS pour vous connecter au cluster de bases de données Aurora MySQL. Créez l'utilisateur de base de données dans Aurora MySQL en suivant les instructions données dansÉtape 6 : Créer des utilisateurs Aurora MySQL utilisant l'authentification Kerberos.

   Les utilisateurs que vous créez de cette façon peuvent se connecter au cluster de bases de données Aurora MySQL en utilisant l'authentification Kerberos. Pour de plus amples informations, consultez Connexion à Aurora MySQL avec l'authentification Kerberos.

Pour utiliser l'authentification Kerberos à l'aide d'un annuaire Microsoft Active Directory sur site ou auto-géré, créez une approbation de forêt. Une approbation de forêt est une relation d'approbation entre deux groupes de domaines. L'approbation peut être unidirectionnelle ou bidirectionnelle. Pour de plus amples informations sur la configuration des approbations de forêts avec Directory Service, veuillez consulter Quand créer une relation d'approbation ? dans le Guide d'administration AWS Directory Service.

Limites de l'authentification Kerberos pour Aurora MySQL

Les limitates suivantes s'appliquent à l'authentification Kerberos pour Aurora MySQL :

- L'authentification Kerberos est prise en charge pour Aurora MySQL versions 3.03 et ultérieures.

  Pour plus d'informations sur la prise en charge d'une Région AWS, consultez Authentification Kerberos avec Aurora MySQL.
- Pour utiliser l'authentification Kerberos avec Aurora MySQL, votre client ou connecteur MySQL doit utiliser la version 8.0.26 ou ultérieure sur les plateformes Unix, 8.0.27 ou ultérieure sur Windows. Sinon, le plug-in `authentication_kerberos_client` côté client n'est pas disponible et vous ne pouvez pas vous authentifier.
- Seul AWS Managed Microsoft AD est pris en charge sur Aurora MySQL. Toutefois, vous pouvez joindre des clusters de bases de données Aurora MySQL à des domaines Microsoft AD gérés partagés qui appartiennent à différents comptes dans la même Région AWS.

  Vous pouvez également utiliser votre propre annuaire Active Directory sur site. Pour de plus amples informations, consultez Étape 2 : (Facultatif) Créer une approbation pour un annuaire Active Directory sur site.
- Lorsque vous utilisez Kerberos pour authentifier un utilisateur qui se connecte au cluster Aurora MySQL à partir de clients MySQL ou de pilotes du système d'exploitation Windows, la casse des caractères du nom d'utilisateur de base de données doit correspondre à celle de l'utilisateur dans Active Directory. Par exemple, si l'utilisateur apparaît dans Active Directory en tant qu'`Admin`, le nom d'utilisateur de base de données doit être `Admin`.

  Cependant, vous pouvez désormais utiliser la comparaison des noms d'utilisateur sans distinction de casse avec le plug-in `authentication_kerberos`. Pour de plus amples informations, consultez Étape 8 : (Facultatif) Configurer la comparaison des noms d’utilisateur sans distinction de casse.
- Vous devez redémarrer les instances de base de données de lecteur après avoir activé la fonction pour installer le plug-in `authentication_kerberos`.
- La réplication vers des instances de base de données qui ne prennent pas en charge le plug-in `authentication_kerberos` peut entraîner un échec de réplication.
- Pour que les bases de données globales Aurora utilisent l'authentification Kerberos, vous devez la configurer pour chaque cluster de bases de données de la base de données globale.
- Le nom de domaine doit comporter moins de 62 caractères.

- Ne modifiez pas le port du cluster de bases de données après avoir activé l'authentification Kerberos. Si vous modifiez le port, l'authentification Kerberos ne fonctionnera plus.

Configuration de l’authentification Kerberos pour les clusters de bases de données Aurora MySQL

AWS Managed Microsoft ADÀ utiliser pour configurer l'authentification Kerberos pour un cluster de base de données Aurora MySQL. Pour configurer l’authentification Kerberos, procédez comme suit :

Rubriques

Étape 1 : créer un répertoire à l'aide de AWS Managed Microsoft AD

Directory Servicecrée un Active Directory entièrement géré dans le AWS cloud. Lorsque vous créez un AWS Managed Microsoft AD annuaire, il Directory Service crée deux contrôleurs de domaine et des serveurs DNS (Domain Name System) en votre nom. Les serveurs de répertoire sont créés dans des sous-réseaux différents d’un VPC. Cette redondance permet de s’assurer que votre annuaire reste accessible, y compris en cas de défaillance.

Lorsque vous créez un AWS Managed Microsoft AD répertoire, il Directory Service exécute les tâches suivantes en votre nom :

- Configuration d’un annuaire Active Directory dans le VPC.
- Création d’un compte d’administrateur d’annuaire avec le nom d’utilisateur `Admin` et le mot de passe spécifié. Ce compte est utilisé pour gérer votre annuaire.

> **Note**
>
> N'oubliez pas d'enregistrer ce mot de passe. Directory Servicene le stocke pas. Vous pouvez le réinitialiser, mais vous ne pouvez pas le récupérer.

- Création d’un groupe de sécurité pour les contrôleurs de l’annuaire.

Lorsque vous lancez unAWS Managed Microsoft AD, AWS crée une unité organisationnelle (UO) qui contient tous les objets de votre répertoire. Cette unité organisationnelle porte le nom NetBIOS que vous avez saisi lorsque vous avez créé votre annuaire. Il se trouve dans la racine du domaine, qui est détenue et gérée parAWS.

Le `Admin` compte créé avec votre AWS Managed Microsoft AD annuaire dispose d'autorisations pour les activités administratives les plus courantes de votre unité d'organisation, notamment :

- Création, mise à jour et suppression des utilisateurs
- Ajouter des ressources à votre domaine, comme des serveurs de fichiers ou d’impression, puis attribuer des autorisations pour ces ressources aux utilisateurs dans votre unité organisationnelle
- Créez des conteneurs OUs et des conteneurs supplémentaires
- Déléguer des autorités
- Restaurer des objets supprimés de la corbeille Active Directory
- Exécuter les PowerShell modules Windows AD et DNS sur le service Web Active Directory

Le compte `Admin` dispose également de droits pour exécuter les activités suivantes au niveau du domaine :

- Gérer les configurations DNS (ajouter, supprimer ou mettre à jour des enregistrements, des zones et des redirecteurs)
- Afficher les journaux d’événements DNS
- Afficher les journaux d’événements de sécurité

Pour créer un répertoire avec AWS Managed Microsoft AD

1. Connectez-vous à la Directory Service console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/directoryservicev2/.

2. Dans le panneau de navigation, choisissez Directories (Répertoires), puis Set up Directory (Configurer un répertoire).
3. Choisissez AWS Managed Microsoft AD. AWS Managed Microsoft ADest la seule option que vous pouvez actuellement utiliser avec Amazon RDS.
4. Entrez les informations suivantes :

   Nom de DNS de l'annuaire

   Nom complet de l'annuaire, par exemple **`corp.example.com`**.

   Nom NetBIOS de l'annuaire

   Nom court de l'annuaire, par exemple **`CORP`**.

   Description de l'annuaire

   (Facultatif) Une description de l'annuaire.

   Mot de passe administrateur

   Mot de passe de l'administrateur de l'annuaire. Le processus de création d'un annuaire crée un compte d'administrateur avec le nom d'utilisateur Admin et ce mot de passe.

   Le mot de passe de l'administrateur de l'annuaire ne peut pas contenir le terme « admin ». Le mot de passe est sensible à la casse et doit comporter entre 8 et 64 caractères. Il doit également contenir au moins un caractère de trois des quatre catégories suivantes :

   - Lettres minuscules (a–z)
   - Lettres majuscules (A–Z)
   - Chiffres (0–9)
   - Caractères non alphanumériques (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

   Confirmer le mot de passe

   Mot de passe de l'administrateur saisi à nouveau.
5. Choisissez Suivant.
6. Entrez les informations suivantes dans la section Networking (Réseaux), puis choisissez Suivant (Next) :

VPC

VPC de l’annuaire. Créez le cluster de bases de données Aurora MySQL dans ce même VPC.

Subnets

Sous-réseaux pour les serveurs d’annuaires. Les deux sous-réseaux doivent être dans des zones de disponibilité différentes.

7. Vérifiez les informations concernant l’annuaire et effectuez les modifications nécessaires. Lorsque les informations sont correctes, choisissez Create directory (Créer l’annuaire).
Page de détails de l’annuaire lors de la création

La création de l’annuaire prend plusieurs minutes. Lorsqu’il est créé, la valeur du champ Statut devient Actif.

Pour consulter les informations relatives à votre annuaire, choisissez le nom de l’annuaire dans la liste. Notez la valeur ID de l’annuaire. Vous en aurez besoin pour créer ou modifier votre cluster de bases de données Aurora MySQL.
ID de l’annuaire sur la page Détails de l’annuaire

Étape 2 : (Facultatif) Créer une approbation pour un annuaire Active Directory sur site

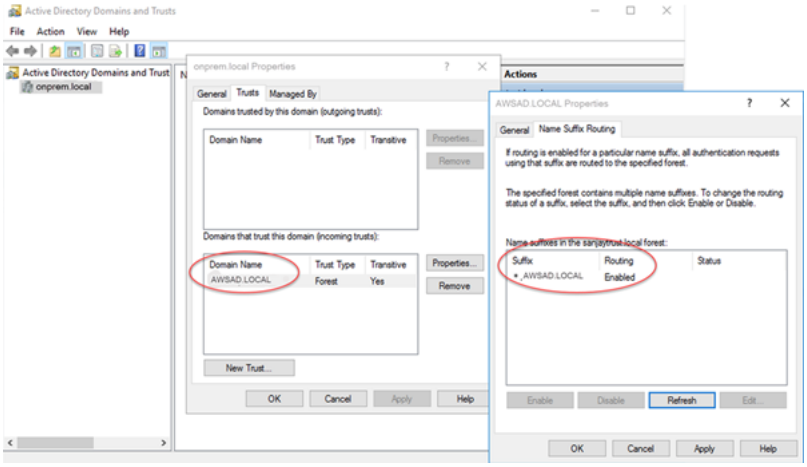
Si vous ne prévoyez pas d’utiliser votre propre Microsoft Active Directory sur site, passez à Étape 3 : Créer un rôle IAM pour une utilisation par Amazon Aurora.

Pour utiliser l'authentification Kerberos avec votre Active Directory local, vous devez créer une relation de domaine de confiance en utilisant une approbation forestière entre votre Microsoft Active Directory local et l'AWS Managed Microsoft ADannuaire (créé dans). Étape 1 : créer un répertoire à l'aide de AWS Managed Microsoft AD L’approbation peut être unidirectionnelle. Dans ce cas, l’annuaire AWS Managed Microsoft AD approuve Microsoft Active Directory sur site. L’approbation peut également être bidirectionnelle. Dans ce cas, les deux Active Directory s’approuvent mutuellement. Pour plus d'informations sur la configuration des approbations à l'aide Directory Service de la section Quand créer une relation de confiance dans le Guide d'AWS Directory Serviceadministration.

> **Note**
>
> Si vous utilisez un annuaire Microsoft Active Directory sur site :

- Les clients Windows ne peuvent pas se connecter à l’aide de points de terminaison Aurora personnalisés. Pour en savoir plus, consultez Connexions de point de terminaison Amazon Aurora.
- Pour les bases de données globales :
  - Les clients Windows peuvent se connecter à l’aide de points de terminaison d’instance ou de points de terminaison de cluster dans la Région AWS principale de la base de données globale.
  - Les clients Windows ne peuvent pas se connecter à l'aide des points de terminaison du cluster dans le secondaireRégions AWS.

Assurez-vous que le nom de domaine de votre Microsoft Active Directory sur site inclut un routage de suffixe DNS correspondant à la relation d’approbation nouvellement créée. La capture d’écran suivante présente un exemple.



Étape 3 : Créer un rôle IAM pour une utilisation par Amazon Aurora

Pour qu'Amazon Aurora fasse appel Directory Service à vous, vous avez besoin d'un rôle Gestion des identités et des accès AWS (IAM) qui utilise la politique IAM gérée. `AmazonRDSDirectoryServiceAccess` Ce rôle permet à Aurora d’effectuer des appels vers Directory Service.

Lorsque vous créez un cluster de base de données à l'aide deAWS Management Console, et que vous en avez l'`iam:CreateRole`autorisation, la console crée automatiquement ce rôle. Dans ce cas, le nom du rôle est `rds-directoryservice-kerberos-access-role`. Sinon, vous devez

créer le rôle IAM manuellement. Lorsque vous créez ce rôle IAM`Directory Service`, choisissez et associez la politique AWS gérée `AmazonRDSDirectoryServiceAccess` à celui-ci.

Pour plus d'informations sur la création de rôles IAM pour un service, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service dans le Guide de l'utilisateur IAM.

Vous pouvez également créer des politiques avec les autorisations obligatoires au lieu d'utiliser la politique gérée IAM `AmazonRDSDirectoryServiceAccess`. Dans ce cas, le rôle IAM doit avoir la politique d'approbation IAM suivante :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Le rôle doit également avoir la politique de rôle IAM suivante.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
```

```
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

Étape 4 : Créer et configurer des utilisateurs

Vous pouvez créer des utilisateurs à l'aide de l'outil Active Directory Users and Computers. Cet outil fait partie des outils Active Directory Domain Services et Active Directory Lightweight Directory Services (Services de domaine Active Directory et Services d'annuaire légers Active Directory). Les utilisateurs représentent des individus ou des entités individuelles qui ont accès à votre annuaire.

Pour créer des utilisateurs dans un Directory Service annuaire, vous utilisez une EC2 instance locale ou Amazon basée sur Microsoft Windows qui est jointe à votre Directory Service annuaire. Vous devez être connecté à l'instance en tant qu'utilisateur disposant de privilèges pour créer des utilisateurs. Pour plus d'informations, consultez Gérer des utilisateurs et des groupes dans AWS Managed Microsoft AD dans le Guide d'administration d'AWS Directory Service.

Étape 5 : Créer ou modifier un cluster de bases de données Aurora MySQL

Créez ou modifiez un cluster de bases de données Aurora MySQL à utiliser avec votre annuaire. Vous pouvez utiliser la console ou l'AWS CLIAPI RDS pour associer un cluster de base de données à un annuaire. Vous pouvez effectuer cette tâche de différentes manières :

- Créez un nouveau cluster de base de données Aurora MySQL à l'aide de la console, de la commande create-db-clusterCLI ou de l'opération Create DBCluster RDS API.

  Pour obtenir des instructions, veuillez consulter Création d'un cluster de bases de données Amazon Aurora.
- Modifiez un cluster de base de données Aurora MySQL existant à l'aide de la console, de la commande modify-db-clusterCLI ou de l'opération DBClusterModify RDS API.

  Pour obtenir des instructions, veuillez consulter Modification d'un cluster de bases de données Amazon Aurora.

- Restaurez un cluster de base de données Aurora MySQL à partir d'un instantané de base de données à l'aide de la console, de la commande CLI restore-db-cluster-from-snapshot ou de l'opération d'API Restore DBCluster FromSnapshot RDS.

  Pour obtenir des instructions, veuillez consulter Restauration à partir d’un instantané de cluster de bases de données.

- Restaurez un cluster de base de données Aurora MySQL à point-in-time l'aide de la console, de la commande restore-db-cluster-to- point-in-time CLI ou de l'opération Restore DBCluster ToPointInTime RDS API.

  Pour obtenir des instructions, veuillez consulter Restauration d’un cluster de bases de données à une date définie.

L’authentification Kerberos est uniquement prise en charge pour les clusters de bases de données Aurora MySQL dans un VPC. Le cluster de bases de données peut se trouver dans le même VPC que l’annuaire ou dans un autre VPC. Le VPC du cluster de bases de données doit avoir un groupe de sécurité VPC qui autorise les communications sortantes vers votre annuaire.

Console

Lorsque vous utilisez la console pour créer, modifier ou restaurer un cluster de bases de données, choisissez Kerberos authentication (Authentification Kerberos) dans la section Database authentication (Authentification de base de données). Choisissez Browse Directory (Parcourir les répertoires), puis sélectionnez le répertoire, ou choisissez Create a new directory (Créer un nouveau répertoire).
Paramètre d’authentification Kerberos lors de la création d’un cluster de bases de données

AWS CLI

Lorsque vous utilisez l'API AWS CLI ou RDS, associez un cluster de base de données à un annuaire. Les paramètres suivants sont nécessaires pour que le cluster de bases de données utilise l’annuaire du domaine que vous avez créé :

- Pour le paramètre `--domain`, vous devez indiquer l’identifiant du domaine (identifiant « d-* ») généré lors de la création de l’annuaire.
- Pour le paramètre `--domain-iam-role-name`, utilisez le rôle que vous avez créé qui utilise la politique IAM gérée `AmazonRDSDirectoryServiceAccess`.

Par exemple, la commande d'interface de ligne de commande suivante modifie un cluster de bases de données de façon à utiliser un annuaire.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --domain d-ID \
    --domain-iam-role-name role-name
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --domain d-ID ^
    --domain-iam-role-name role-name
```

> **Important**
>
> Si vous modifiez un cluster de bases de données pour activer l'authentification Kerberos, redémarrez les instances de base de données de lecteur après avoir effectué la modification.

Étape 6 : Créer des utilisateurs Aurora MySQL utilisant l'authentification Kerberos

Le cluster de base de données est joint au AWS Managed Microsoft AD domaine. Vous pouvez ainsi créer des utilisateurs Aurora MySQL à partir des utilisateurs Active Directory de votre domaine. Les autorisations de base de données sont gérées via des autorisations Aurora MySQL standard qui sont accordées et révoquées à partir de ces utilisateurs.

Vous pouvez autoriser un utilisateur Active Directory à s'authentifier avec Aurora MySQL. Pour ce faire, utilisez d'abord les informations d'identification de l'utilisateur principal Amazon RDS pour vous connecter au cluster de bases de données Aurora MySQL comme avec n'importe quel autre cluster de bases de données. Après vous être connecté, créez un utilisateur authentifié en externe avec l'authentification Kerberos dans Aurora MySQL comme indiqué ici :

```
CREATE USER user_name@'host_name' IDENTIFIED WITH 'authentication_kerberos' BY
 'realm_name';
```

- Remplacez *user_name* par le nom de l'utilisateur. Les utilisateurs (personnes et applications) de votre domaine peuvent désormais se connecter au cluster de bases de données à partir d'un ordinateur client joint au domaine à l'aide de l'authentification Kerberos.
- Remplacez *host_name* par le nom d'hôte. Vous pouvez utiliser `%` comme un caractère générique. Vous pouvez également utiliser des adresses IP spécifiques pour le nom d'hôte.
- Remplacez *realm_name* par le nom de domaine du répertoire du domaine. Le nom de domaine est généralement identique au nom de domaine DNS en lettres majuscules, par exemple `CORP.EXAMPLE.COM`. Un domaine est un groupe de systèmes qui utilise le même centre de distribution de clés Kerberos.

L'exemple suivant crée un utilisateur de base de données dont le nom `Admin` s'authentifie auprès de l'annuaire Active Directory à l'aide du nom de domaine `MYSQL.LOCAL`.

```
CREATE USER Admin@'%' IDENTIFIED WITH 'authentication_kerberos' BY 'MYSQL.LOCAL';
```

Modification d'un identifiant Aurora MySQL existant

Vous pouvez également modifier un identifiant Aurora MySQL existant pour utiliser l'authentification Kerberos avec la syntaxe suivante :

```
ALTER USER user_name IDENTIFIED WITH 'authentication_kerberos' BY 'realm_name';
```

Étape 7 : Configurer un client MySQL

Pour configurer un client MySQL, procédez comme suit :

1. Créez un fichier `krb5.conf` (ou équivalent) pointant vers le domaine.
2. Vérifiez que le trafic peut circuler entre l'hôte client etDirectory Service. Utilisez un utilitaire réseau tel que Netcat pour les opérations suivantes :
   - Vérifiez le trafic via DNS pour le port 53.
   - Vérifiez le trafic dépassé TCP/UDP pour le port 53 et pour Kerberos, qui inclut les ports 88 et 464 pour. Directory Service
3. Vérifiez que le trafic peut circuler entre l'hôte du client et l'instance de base de données via le port de la base de données. Par exemple, utilisez `mysql` pour vous connecter à la base de données et y accéder.

Voici un exemple de `krb5.conf` contenu pourAWS Managed Microsoft AD.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

Vous trouverez ci-après un exemple de contenu `krb5.conf` pour un annuaire Microsoft Active Directory sur site.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
 .rds.amazonaws.com = EXAMPLE.COM
 .amazonaws.com.cn = EXAMPLE.COM
 .amazon.com = EXAMPLE.COM
```

Étape 8 : (Facultatif) Configurer la comparaison des noms d’utilisateur sans distinction de casse

Par défaut, la casse des caractères du nom d’utilisateur de base de données MySQL doit correspondre à celle de l’identifiant Active Directory. Cependant, vous pouvez

désormais utiliser la comparaison des noms d'utilisateur sans distinction de casse avec le plug-in `authentication_kerberos`. Pour ce faire, vous devez définir le paramètre `authentication_kerberos_caseins_cmp` de cluster de bases de données sur `true`.

Pour utiliser la comparaison des noms d'utilisateur sans distinction de casse

1. Créez un groupe personnalisé de paramètres de cluster de bases de données. Suivez la procédure fournie dans Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora.
2. Modifiez le nouveau groupe de paramètres pour définir la valeur de `authentication_kerberos_caseins_cmp` sur `true`. Suivez la procédure fournie dans Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora.
3. Associez le groupe de paramètres de cluster de bases de données à votre cluster de bases de données Aurora MySQL. Suivez la procédure fournie dans Association d'un groupe de paramètres de cluster de bases de données à un cluster de bases de données Amazon Aurora.
4. Redémarrez le cluster de bases de données.

Connexion à Aurora MySQL avec l'authentification Kerberos

Pour éviter les erreurs, utilisez un client MySQL avec la version 8.0.26 ou ultérieure sur les plateformes Unix, ou 8.0.27 ou ultérieure sur Windows.

Utilisation de l'identifiant Kerberos Aurora MySQL pour se connecter au cluster de bases de données

Pour vous connecter à Aurora MySQL à l'aide de l'authentification Kerberos, vous devez vous connecter comme l'utilisateur de base de données que vous avez créé à l'aide des instructions fournies dans Étape 6 : Créer des utilisateurs Aurora MySQL utilisant l'authentification Kerberos.

À partir d'une invite de commande, connectez-vous à un des points de terminaison associés à votre cluster de bases de données Aurora MySQL. Lorsque vous êtes invité à entrer le mot de passe, entrez le mot de passe Kerberos associé à ce nom d'utilisateur.

Lorsque vous vous authentifiez avec Kerberos, un ticket d'attribution de tickets (TGT) est généré s'il n'en existe pas déjà un. Le plug-in `authentication_kerberos` utilise le TGT pour obtenir un ticket de service, qui est ensuite présenté au serveur de base de données Aurora MySQL.

Vous pouvez utiliser le client MySQL pour vous connecter à Aurora MySQL avec une authentification Kerberos sous Windows ou Unix.

Unix

Vous pouvez vous connecter avec l'une des méthodes suivantes :

- Obtenez le TGT manuellement. Dans ce cas, il n'est pas nécessaire de fournir le mot de passe au client MySQL.
- Fournissez le mot de passe pour la connexion Active Directory directement au client MySQL.

Le plug-in côté client est pris en charge sur les plateformes Unix pour les versions client de MySQL 8.0.26 et ultérieures.

Pour vous connecter en obtenant le TGT manuellement

1. Sur l'interface de ligne de commande, utilisez la commande suivante pour obtenir le TGT.

   ```
   kinit user_name
   ```

2. Utilisez la commande `mysql` suivante pour vous connecter au point de terminaison de l'instance de base de données de votre cluster de bases de données.

   ```
   mysql -h DB_instance_endpoint -P 3306 -u user_name -p
   ```

   > **Note**
   >
   > L'authentification peut échouer si le keytab a fait l'objet d'une rotation sur l'instance de base de données. Dans ce cas, obtenez un nouveau TGT en exécutant à nouveau `kinit`.

Pour vous connecter directement

1. Sur l'interface de ligne de commande, utilisez la commande `mysql` suivante pour vous connecter au point de terminaison de l'instance de base de données de votre cluster de bases de données.

   ```
   mysql -h DB_instance_endpoint -P 3306 -u user_name -p
   ```

2. Saisissez le mot de passe de l'utilisateur Active Directory.

Windows

Sous Windows, l'authentification est généralement effectuée au moment de la connexion. Vous n'avez donc pas besoin d'obtenir le TGT manuellement pour vous connecter au cluster de bases de données Aurora MySQL. La casse du nom d'utilisateur de base de données doit correspondre à la casse des caractères de l'utilisateur dans Active Directory. Par exemple, si l'utilisateur apparaît dans Active Directory en tant qu'`Admin`, le nom d'utilisateur de base de données doit être `Admin`.

Le plug-in côté client est pris en charge sur les plateformes Windows pour les versions client de MySQL 8.0.27 et ultérieures.

Pour vous connecter directement

- Sur l'interface de ligne de commande, utilisez la commande `mysql` suivante pour vous connecter au point de terminaison de l'instance de base de données de votre cluster de bases de données.

```
mysql -h DB_instance_endpoint -P 3306 -u user_name
```

Authentification Kerberos avec des bases de données globales Aurora

L'authentification Kerberos for Aurora MySQL est prise en charge pour les bases de données globales Aurora. Pour authentifier les utilisateurs du cluster de bases de données secondaire à l'aide de l'Active Directory du cluster de bases de données principal, répliquez l'Active Directory sur la Région AWS secondaire. Vous activez l'authentification Kerberos sur le cluster secondaire en utilisant le même ID de domaine que pour le cluster principal. La réplication AWS Managed Microsoft AD est prise en charge uniquement avec la version Enterprise d'Active Directory. Pour plus d'informations, consultez Multi-Region replication (Réplication multi-régions) dans le Guide d'administration AWS Directory Service.

Migration de RDS for MySQL vers Aurora MySQL

Après avoir migré de RDS for MySQL avec l'authentification Kerberos activée vers Aurora MySQL, modifiez les utilisateurs créés avec le plug-in `auth_pam` pour qu'ils utilisent le plug-in `authentication_kerberos`. Par exemple :

```
ALTER USER user_name IDENTIFIED WITH 'authentication_kerberos' BY 'realm_name';
```

Prévention de la mise en cache des tickets

Si aucun TGT valide n'existe au démarrage de l'application cliente MySQL, l'application peut obtenir le TGT et le mettre en cache. Si vous souhaitez empêcher la mise en cache du TGT, définissez un paramètre de configuration dans le fichier `/etc/krb5.conf`.

> **Note**
>
> Cette configuration s'applique uniquement aux hôtes clients exécutant Unix, et non Windows.

Pour empêcher la mise en cache du TGT

- Ajoutez une section `[appdefaults]` à `/etc/krb5.conf` comme suit :

```
[appdefaults]
  mysql = {
    destroy_tickets = true
  }
```

Journalisation pour l'authentification Kerberos

La variable d'environnement `AUTHENTICATION_KERBEROS_CLIENT_LOG` définit le niveau de journalisation pour l'authentification Kerberos. Vous pouvez utiliser les journaux pour le débogage côté client.

Les valeurs autorisées sont comprises entre 1 et 5. Les messages du journal sont écrits sur la sortie d'erreur standard. La table suivante décrit chaque niveau de journalisation.

Logging level (Niveau de journalisation)	Description
1 ou non défini	Aucune journalisation
2	Messages d’erreur
3	Messages d'erreur et d'avertissement
4	Messages d'erreur, d'avertissement et d'information

Logging level (Niveau de journalisation)	Description
5	Messages d'erreur, d'avertissement, d'information et de débogage

Gestion d'un cluster de bases de données dans un domaine

Vous pouvez utiliser l'AWS CLI ou l'API RDS pour gérer votre cluster de bases de données et sa relation avec votre annuaire Active Directory géré. Par exemple, vous pouvez associer un annuaire Active Directory pour l'authentification Kerberos et dissocier un annuaire Active Directory pour désactiver l'authentification Kerberos. Vous pouvez également transférer un cluster de bases de données vers un autre afin qu'il soit authentifié en externe par un annuaire Active Directory.

Par exemple, l'API Amazon RDS vous permet d'effectuer les actions suivantes :

- Pour retenter l'activation de l'authentification Kerberos en cas d'échec d'appartenance, utilisez l'opération d'API `ModifyDBInstance` et spécifiez l'ID d'annuaire d'appartenance actuelle.
- Pour mettre à jour le nom du rôle IAM de l'appartenance, utilisez l'opération d'API `ModifyDBInstance` et spécifiez l'ID d'annuaire de l'appartenance actuelle et le nouveau rôle IAM.
- Pour désactiver l'authentification Kerberos sur un cluster de bases de données, utilisez l'opération d'API `ModifyDBInstance` et spécifiez `none` comme paramètre de domaine.
- Pour déplacer un cluster de bases de données d'un domaine à un autre, utilisez l'opération d'API `ModifyDBInstance` et spécifiez l'identifiant du nouveau domaine en tant que paramètre de domaine.
- Pour répertorier l'appartenance pour chaque cluster de bases de données, utilisez l'opération d'API `DescribeDBInstances`.

Présentation de l'appartenance au domaine

Après la création ou la modification de votre cluster de bases de données, il devient un membre du domaine. Vous pouvez consulter le statut de l'appartenance au domaine pour le cluster de bases de données en exécutant la commande d'interface de ligne de commande describe-db-clusters. Le statut du cluster de bases de données peut avoir les valeurs suivantes :

- `kerberos-enabled` : l'authentification Kerberos est activée sur le cluster de bases de données.

- `enabling-kerberos` : AWS est en train d'activer l'authentification Kerberos sur ce cluster de bases de données.
- `pending-enable-kerberos` : l'activation de l'authentification Kerberos est en attente sur ce cluster de bases de données.
- `pending-maintenance-enable-kerberos` – AWS tentera d'activer l'authentification Kerberos sur ce cluster de bases de données lors de la prochaine fenêtre de maintenance planifiée.
- `pending-disable-kerberos` : la désactivation de l'authentification Kerberos est en attente sur ce cluster de bases de données.
- `pending-maintenance-disable-kerberos` – AWS tentera de désactiver l'authentification Kerberos sur ce cluster de bases de données lors de la prochaine fenêtre de maintenance planifiée.
- `enable-kerberos-failed` : un problème de configuration a empêché AWS d'activer l'authentification Kerberos sur le cluster de bases de données. Vérifiez et corrigez votre configuration avant d'émettre à nouveau la commande de modification du cluster de bases de données.
- `disabling-kerberos` : AWS est en train de désactiver l'authentification Kerberos sur ce cluster de bases de données.

Une demande d'activation de l'authentification Kerberos peut échouer à cause d'un problème de connectivité réseau ou d'un rôle IAM incorrect. Par exemple, supposons que vous créez un cluster de bases de données ou modifiez un cluster de bases de données et que la tentative d'activation de l'authentification Kerberos échoue. Si cela se produit, réémettez la commande modify ou modifiez le cluster de bases de données nouvellement créé pour joindre le domaine.

Migration de données vers un cluster de bases de données Amazon Aurora MySQL

Vous avez plusieurs options pour la migration des données de votre base de données existante vers un cluster de bases de données Amazon Aurora MySQL. Vos options de migration dépendent également de la base de données à partir de laquelle vous effectuez la migration et de la taille des données que vous migrez.

Il existe deux types différents de migration : physique et logique. La migration physique signifie que des copies physiques des fichiers de base de données sont utilisées pour migrer la base de données. La migration logique signifie que la migration s'effectue en appliquant des modifications logiques à la base de données, telles que des insertions, des mises à jour et des suppressions.

La migration physique présente les avantages suivants :

- La migration physique est plus rapide que la migration logique, notamment pour les bases de données volumineuses.
- Les performances de base de données ne sont pas réduites lorsqu'une sauvegarde est effectuée pour une migration physique.
- La migration physique peut migrer tout le contenu de la base de données source, y compris les composants de base de données complexes.

La migration physique présente les limites suivantes :

- Le paramètre `innodb_page_size` doit être défini sur sa valeur par défaut (`16KB`).
- Le paramètre `innodb_data_file_path` doit être configuré avec un seul fichier de données qui utilise le nom de fichier de données par défaut `"ibdata1:12M:autoextend"`. Les bases de données comportant deux fichiers de données, ou avec un fichier de données portant un nom différent, ne peuvent pas faire l'objet d'une migration à l'aide de cette méthode.

  Voici des exemples de noms de fichier non autorisés : `"innodb_data_file_path=ibdata1:50M; ibdata2:50M:autoextend"` et `"innodb_data_file_path=ibdata01:50M:autoextend"`.
- Le paramètre `innodb_log_files_in_group` doit être défini sur sa valeur par défaut (`2`).

La migration logique présente les avantages suivants :

- Vous pouvez migrer des sous-ensembles de la base de données, comme par exemple des tables spécifiques ou des parties d’une table.
- Les données peuvent être migrées quelle que soit la structure de stockage physique.

La migration logique présente les limites suivantes :

- La migration logique est généralement plus lente que la migration physique.
- Les composants de base de données complexes peuvent ralentir le processus de migration logique. Dans certains cas, les composants de base de données complexes peuvent même bloquer la migration logique.

Le tableau ci-dessous décrit vos options et le type de migration pour chaque option.

Migration à partir de	Type de migration	Solution
Une instance de base de données RDS pour MySQL	Physique	Vous pouvez migrer les données d'une instance de base de données RDS for MySQL en créant d'abord un réplica en lecture Aurora MySQL d'une instance de base de données MySQL. Lorsque le retard du réplica entre l'instance de base de données MySQL et le réplica en lecture Aurora MySQL est égal à 0, vous pouvez diriger vos applicati ons clientes vers la lecture à partir du réplica en lecture Aurora, puis arrêter la réplication pour transformer le réplica en lecture Aurora MySQL en cluster de bases de données Aurora MySQL pour la lecture et l'écriture. Pour plus d'informations, consultez Migration des données d'une instance de base de données RDS for MySQL vers un cluster de bases de données Amazon Aurora MySQL à l'aide d'un réplica en lecture Aurora.
Un instantané de bases de données RDS pour MySQL	Physique	Vous pouvez migrer les données directement d'un instantané de bases de données RDS for MySQL vers un cluster de bases de données Amazon Aurora MySQL. Pour en savoir plus, consultez Migration d'un instantané RDS for MySQL vers Aurora.
Une base de données MySQL externe à Amazon RDS	Logique	Vous pouvez créer un vidage de vos données à l'aide de l'utilitaire `mysqldump` , puis importer ces

Migration à partir de	Type de migration	Solution
		données dans un cluster de bases de données Amazon Aurora MySQL existant. Pour en savoir plus, consultez Migration logique de MySQL vers Amazon Aurora MySQL à l'aide de mysqldump. Pour exporter des métadonnées pour les utilisateurs de bases de données pendant la migration à partir d'une base de données MySQL externe, vous pouvez aussi utiliser une commande MySQL Shell à la place de `mysqldump` . Pour plus d'informations, consultez Utilitaire de vidage d'instance, Utilitaire de vidage de schéma et Utilitaire de vidage de table. **Note** L'utilitaire mysqlpump est obsolète depuis MySQL 8.0.34.

Migration à partir de	Type de migration	Solution
Une base de données MySQL externe à Amazon RDS	Physique	Vous pouvez copier les fichiers de sauvegarde de votre base de données vers un compartiment Amazon Simple Storage Service (Amazon S3), puis restaurer un cluster de bases de données Amazon Aurora MySQL à partir de ces fichiers. Cette option peut être considérablement plus rapide que la migration des données à l'aide de `mysqldump` . Pour plus d'informations, consultez Migration physique depuis MySQL à l'aide de Percona XtraBackup et Amazon S3.
Une base de données MySQL externe à Amazon RDS	Logique	Vous pouvez sauvegarder les données de votre base de données sous forme de fichiers texte et copier ces derniers vers un compartiment Amazon S3. Vous pouvez ensuite charger ces données dans un cluster de bases de données Aurora MySQL existant à l'aide de la commande MySQL `LOAD DATA FROM S3`. Pour plus d'informations, consultez Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3.

Migration à partir de	Type de migration	Solution
Une base de données qui n'est pas compatible avec MySQL	Logique	Vous pouvez utiliser AWS Database Migration Service (AWS DMS) pour migrer les données d'une base de données qui n'est pas compatible avec MySQL. Pour plus d'informations sur AWS DMS, consultez Présentation d'AWS Database Migration Service.

> **Note**
>
> Si vous effectuez la migration d'une base de données MySQL externe à Amazon RDS, les options de migration décrites dans le tableau sont prises en charge seulement si votre base de données prend en charge les espaces de table InnoDB ou MyISAM.
> Si la base de données MySQL que vous migrez vers Aurora MySQL utilise `memcached`, supprimez `memcached` avant de la migrer.
> Vous ne pouvez pas migrer de certaines anciennes versions de MySQL 8.0, notamment des versions 8.0.11, 8.0.13 et 8.0.15, vers Aurora MySQL 3.05 et versions ultérieures. Nous vous recommandons de passer à MySQL version 8.0.28 avant de procéder à la migration.

Migration des données d'une base de données MySQL externe vers un cluster de bases de données Amazon Aurora MySQL

Si votre base de données prend en charge les espaces de table InnoDB ou MySQL, ces options permettent la migration de vos données vers un cluster de bases de données Amazon Aurora MySQL :

- Vous pouvez créer un vidage de vos données à l'aide de l'utilitaire `mysqldump`, puis importer ces données dans un cluster de bases de données Amazon Aurora MySQL existant. Pour de plus amples informations, consultez Migration logique de MySQL vers Amazon Aurora MySQL à l'aide de mysqldump.
- Vous pouvez copier les fichiers des sauvegardes complètes et incrémentielles de votre base de données vers un compartiment Amazon S3, puis restaurer vers un cluster de bases de données Amazon Aurora MySQL à partir de ces fichiers. Cette option peut être considérablement plus rapide que la migration des données à l'aide de `mysqldump`. Pour de plus amples informations, consultez Migration physique depuis MySQL à l'aide de Percona XtraBackup et Amazon S3.

Rubriques

Migration physique depuis MySQL à l'aide de Percona XtraBackup et Amazon S3

Vous pouvez copier les fichiers de sauvegarde complète et incrémentielle de votre base de données MySQL version 5.7 ou 8.0 source vers un compartiment Amazon S3. Vous pouvez ensuite effectuer une restauration sur un cluster de bases de données Amazon Aurora MySQL avec la même version majeure du moteur de base de données à partir de ces fichiers.

Cette option peut être considérablement plus rapide que la migration des données à l'aide de `mysqldump`, parce que l'utilisation de `mysqldump` réexécute toutes les commandes pour recréer le schéma et les données de votre base de données source dans votre nouveau cluster de bases de données Aurora MySQL. En copiant vos fichiers de données MySQL source, Aurora MySQL peut immédiatement utiliser ces fichiers en tant que données d'un cluster de bases de données Aurora MySQL.

Vous pouvez également réduire au maximum la durée d'indisponibilité en utilisant la réplication des journaux binaires pendant le processus de migration. Si vous utilisez la réplication des journaux

binaires, la base de données MySQL externe reste ouverte aux transactions pendant que les données sont migrées vers le cluster de bases de données Aurora MySQL. Une fois le cluster de bases de données Aurora MySQL créé, utilisez la réplication des journaux binaires pour synchroniser le cluster de bases de données Aurora MySQL avec les transactions qui ont eu lieu après la sauvegarde. Une fois le cluster de bases de données Aurora MySQL synchronisé avec la base de données MySQL, vous terminez la migration en basculant complètement vers le cluster de bases de données Aurora MySQL pour les nouvelles transactions. Pour plus d'informations, consultez Synchronisation du cluster de bases de données Amazon Aurora MySQL sur la base de données MySQL avec la réplication.

Table des matières

Limites et considérations

Les limites et considérations suivantes s'appliquent à la restauration vers un cluster de bases de données Amazon Aurora MySQL à partir d'un compartiment Amazon S3 :

- Vous pouvez migrer vos données uniquement vers un nouveau cluster de bases de données, et non vers un cluster de bases de données existant.
- Vous devez utiliser Percona XtraBackup pour sauvegarder vos données sur S3. Pour de plus amples informations, veuillez consulter Installation de Percona XtraBackup.
- Le compartiment Amazon S3 et le cluster de base de données Aurora MySQL doivent se trouver dans la même AWS région.
- Vous ne pouvez pas restaurer à partir des éléments suivants :
  - Une exportation d'un instantané de cluster de bases de données sur Amazon S3. Vous ne pouvez pas non plus migrer des données à partir de l'exportation d'un instantané de cluster de bases de données dans votre compartiment S3.
  - Une base de données source chiffrée, mais vous pouvez chiffrer les données en cours de migration. Vous pouvez également laisser les données non chiffrées pendant le processus de migration.
  - Une base de données MySQL 5.5 ou 5.6
- Percona Server for MySQL n'est pas pris en charge en tant que base de données source, car il peut contenir des tables `compression_dictionary*` dans le schéma `mysql`.
- Vous ne pouvez pas effectuer de restauration dans un cluster de bases de données Aurora Serverless.
- La rétromigration n'est pas prise en charge pour les versions majeurs ni les versions mineures. Par exemple, vous ne pouvez pas migrer de MySQL version 8.0 vers Aurora MySQL version 2 (compatible avec MySQL 5.7). De même, vous ne pouvez pas migrer de MySQL version 8.0.32 vers Aurora MySQL version 3.03, compatible avec la version 8.0.26 de la communauté MySQL.
- Vous ne pouvez pas migrer de certaines anciennes versions de MySQL 8.0, notamment des versions 8.0.11, 8.0.13 et 8.0.15, vers Aurora MySQL 3.05 et versions ultérieures. Nous vous recommandons de passer à MySQL version 8.0.28 avant de procéder à la migration.
- L'importation à partir d'Amazon S3 n'est pas prise en charge sur la classe d'instance de base de données db.t2.micro. Toutefois, vous pouvez procéder à une restauration vers une autre classe d'instance de base de données, puis modifier la classe d'instance de base de données ultérieurement. Pour plus d'informations sur les classes d'instance de base de données, consultez Classes d'instance de base de données Amazon Aurora.

- Amazon S3 limite la taille d'un fichier chargé dans un compartiment S3 à 5 To. Si un fichier de sauvegarde dépasse 5 To, vous devez diviser celui-ci en plusieurs fichiers plus petits.
- Amazon RDS limite le nombre de fichiers chargés dans un compartiment S3 à 1 million. Si les données de sauvegarde de votre base de données, y compris toutes les sauvegardes complètes et incrémentielles, dépassent 1 million de fichiers, utilisez un fichier Gzip (.gz), tar (.tar.gz) ou Percona xbstream (.xbstream) pour stocker les fichiers de sauvegarde complète et incrémentielle dans le compartiment S3. Percona XtraBackup 8.0 prend uniquement en charge Percona xbstream pour la compression.
- Pour fournir des services de gestion à chaque cluster de bases de données, l'utilisateur `rdsadmin` est créé lors de la création du cluster de bases de données. Comme il s'agit d'un utilisateur réservé dans RDS, les limitations suivantes s'appliquent :
  - Les fonctions, les procédures, les vues, les événements et les déclencheurs avec le définisseur `'rdsadmin'@'localhost'` ne sont pas importés. Pour plus d'informations, consultez Objets stockés avec 'rdsadmin'@'localhost' comme définisseur et Privilèges d'utilisateur principal avec Amazon Aurora MySQL..
  - Lorsque le cluster de bases de données Aurora MySQL est créé, un utilisateur principal est créé avec les privilèges maximum pris en charge. Lors de la restauration à partir d'une sauvegarde, tous les privilèges non pris en charge attribués aux utilisateurs importés sont automatiquement supprimés durant l'importation.

    Pour identifier les utilisateurs susceptibles d'être concernés, consultez Comptes d'utilisateur avec des privilèges non pris en charge. Pour plus d'informations sur les privilèges pris en charge dans Aurora MySQL, consultez Modèle de privilège basé sur les rôles.
- Pour Aurora MySQL version 3, les privilèges dynamiques ne sont pas importés. Les privilèges dynamiques pris en charge par Aurora peuvent être importés après la migration. Pour plus d'informations, consultez Privilèges dynamiques dans Aurora MySQL version 3.
- Les tables créées par l'utilisateur dans le schéma `mysql` ne sont pas migrées.
- Le paramètre `innodb_data_file_path` doit être configuré avec un seul fichier de données qui utilise le nom de fichier de données par défaut `ibdata1:12M:autoextend`. Les bases de données comportant deux fichiers de données, ou avec un fichier de données portant un nom différent, ne peuvent pas faire l'objet d'une migration à l'aide de cette méthode.

  Voici des exemples de noms de fichiers non autorisés : `innodb_data_file_path=ibdata1:50M`, `ibdata2:50M:autoextend` et `innodb_data_file_path=ibdata01:50M:autoextend`.

- Vous ne pouvez pas migrer à partir d'une base de données source dotée de tables définies à l'extérieur du répertoire de données MySQL par défaut.
- La taille maximale prise en charge pour les sauvegardes non compressées utilisant cette méthode est actuellement limitée à 64 Tio. Pour les sauvegardes compressées, cette limite est abaissée pour tenir compte de l'espace requis pour la décompression. Dans de tels cas, la taille de sauvegarde maximale prise en charge serait (`64 TiB – compressed backup size`).
- Aurora MySQL ne prend pas en charge l'importation de MySQL ni d'autres composants et plugins externes.
- Aurora MySQL ne restaure pas tous les éléments de votre base de données. Nous vous recommandons d'enregistrer le schéma de base de données et les valeurs pour les éléments suivants à partir de votre base de données MySQL source et de les ajouter à votre cluster de bases de données Aurora MySQL restauré après qu'il a été créé :
  - Comptes utilisateurs
  - Fonctions
  - Procédures stockées
  - Informations de fuseau horaire. Les informations de fuseau horaire sont chargées depuis le système d'exploitation local de votre cluster de bases de données Aurora MySQL. Pour plus d'informations, consultez Fuseau horaire local pour les clusters de bases de données Amazon Aurora.

Avant de commencer

Avant de pouvoir copier vos données dans un compartiment Amazon S3 et de les restaurer dans un cluster de bases de données à partir de ces fichiers, vous devez effectuer les opérations suivantes :

- Installez Percona XtraBackup sur votre serveur local.
- Autoriser Aurora MySQL à accéder à votre compartiment Amazon S3 en votre nom.

Installation de Percona XtraBackup

Amazon Aurora peut restaurer un cluster de base de données à partir de fichiers créés à l'aide de Percona XtraBackup. Vous pouvez installer Percona XtraBackup depuis Téléchargements de logiciels - Percona.

Pour la migration vers MySQL 5.7, utilisez Percona XtraBackup 2.4.

Pour la migration vers MySQL 8.0, utilisez Percona XtraBackup 8.0. Assurez-vous que la version Percona est compatible avec la XtraBackup version du moteur de votre base de données source.

Autorisations requises

Pour migrer vos données MySQL vers un cluster de bases de données Amazon Aurora MySQL, plusieurs autorisations sont requises :

- L'utilisateur qui demande à Aurora de créer un nouveau cluster à partir d'un compartiment Amazon S3 doit être autorisé à répertorier les compartiments pour votre AWS compte. Vous accordez cette autorisation à l'utilisateur à l'aide d'une politique Gestion des identités et des accès AWS (IAM).
- Aurora nécessite l'autorisation d'agir en votre nom pour accéder au compartiment Amazon S3 dans lequel vous stockez les fichiers utilisés pour créer votre cluster de bases de données Amazon Aurora MySQL. Vous accordez à Aurora les autorisations requises à l'aide d'un rôle de service IAM.
- L'utilisateur qui fait la demande doit avoir également l'autorisation de répertorier les rôles IAM pour votre compte AWS.
- Si l'utilisateur qui fait la demande doit créer le rôle de service IAM ou demander la création du rôle de service IAM par Aurora (à l'aide de la console), cet utilisateur doit avoir l'autorisation de créer un rôle IAM pour votre compte AWS.
- Si vous prévoyez de chiffrer les données pendant le processus de migration, mettez à jour la politique IAM de l'utilisateur qui effectuera la migration afin d'accorder l'accès RDS aux données AWS KMS keys utilisées pour chiffrer les sauvegardes. Pour obtenir des instructions, veuillez consulter Création d'une politique IAM pour accéder aux ressources AWS KMS.

Par exemple, la politique IAM suivante accorde à un utilisateur les autorisations minimales requises pour utiliser la console afin de répertorier les rôles IAM, créer un rôle IAM et répertorier les compartiments Amazon S3 de votre compte, ainsi que les clés KMS.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles",
```

```
                "iam:CreateRole",
                "iam:CreatePolicy",
                "iam:AttachRolePolicy",
                "s3:ListBucket",
                "kms:ListKeys"
            ],
            "Resource": "*"
        }
    ]
}
```

En outre, pour qu'un utilisateur associe un rôle IAM à un compartiment Amazon S3, l'utilisateur IAM doit avoir l'autorisation `iam:PassRole` pour ce rôle IAM. Cette autorisation permet à un administrateur de limiter les rôles IAM qu'un utilisateur peut associer aux compartiments Amazon S3.

Par exemple, la stratégie IAM suivante permet à un utilisateur d'associer le rôle nommé `S3Access` à un compartiment Amazon S3.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
        {
            "Sid":"AllowS3AccessRole",
            "Effect":"Allow",
            "Action":"iam:PassRole",
            "Resource":"arn:aws:iam::123456789012:role/S3Access"
        }
    ]
}
```

Pour plus d'informations sur les autorisations utilisateur IAM, consultez Gestion de l'accès à l'aide de politiques.

Création d'un rôle de service IAM

Vous pouvez AWS Management Console créer un rôle pour vous en choisissant l'option Créer un nouveau rôle (présentée plus loin dans cette rubrique). Si vous sélectionnez cette option et spécifiez

un nom pour le nouveau rôle, Aurora crée le rôle de service IAM nécessaire pour qu'Aurora accède à votre compartiment Amazon S3 avec le nom que vous fournissez.

Vous pouvez aussi créer manuellement le rôle à l'aide de la procédure suivante.

Pour créer un rôle IAM permettant à Aurora d'accéder à Amazon S3

1. Suivez les étapes de Création d'une stratégie IAM pour accéder aux ressources Amazon S3.
2. Suivez les étapes de Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS.
3. Suivez les étapes de Association d'un rôle IAM à un cluster de bases de données Amazon Aurora MySQL.

Sauvegarde de fichiers à restaurer comme cluster de bases de données Amazon Aurora MySQL

Vous pouvez créer une sauvegarde complète de vos fichiers de base de données MySQL à l'aide de Percona XtraBackup et télécharger les fichiers de sauvegarde dans un compartiment Amazon S3. Si vous utilisez déjà Percona XtraBackup pour sauvegarder les fichiers de votre base de données MySQL, vous pouvez également télécharger vos répertoires et fichiers de sauvegarde complets et incrémentiels existants dans un compartiment Amazon S3.

Rubriques

- Création d'une sauvegarde complète avec Percona XtraBackup
- Utilisation de sauvegardes incrémentielles avec Percona XtraBackup
- Considérations relatives à la sauvegarde

Création d'une sauvegarde complète avec Percona XtraBackup

Pour créer une sauvegarde complète de vos fichiers de base de données MySQL qui peuvent être restaurés depuis Amazon S3 afin de créer un cluster de bases de données Aurora MySQL, utilisez l' XtraBackup utilitaire Percona (`xtrabackup`) pour sauvegarder votre base de données.

Par exemple, la commande suivante crée une sauvegarde d'une base de données MySQL et stocke les fichiers dans le dossier `/on-premises/s3-restore/backup`.

```
xtrabackup --backup --user=<myuser> --password=<password> --target-dir=</on-premises/
s3-restore/backup>
```

Si vous souhaitez compresser votre sauvegarde en un seul fichier (qui peut être divisé, si nécessaire), vous pouvez utiliser l'option `--stream` pour enregistrer votre sauvegarde dans l'un des formats suivants :

- Gzip (.gz)
- tar (.tar)
- Percona xbstream (.xbstream)

La commande suivante crée une sauvegarde de votre base de données MySQL, divisée en plusieurs fichiers Gzip.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | gzip - | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar.gz
```

La commande suivante crée une sauvegarde de votre base de données MySQL, divisée en plusieurs fichiers tar.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=tar \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.tar
```

La commande suivante crée une sauvegarde de votre base de données MySQL, divisée en plusieurs fichiers xbstream.

```
xtrabackup --backup --user=<myuser> --password=<password> --stream=xbstream \
   --target-dir=</on-premises/s3-restore/backup> | split -d --bytes=500MB \
   - </on-premises/s3-restore/backup/backup>.xbstream
```

> **Note**
>
> Si vous obtenez l'erreur suivante, cela indique peut-être que vous avez mélangé des formats de fichiers dans votre commande :
>
> ```
> ERROR:/bin/tar: This does not look like a tar archive
> ```

Une fois que vous avez sauvegardé votre base de données MySQL à l'aide de l' XtraBackup utilitaire Percona, vous pouvez copier vos répertoires et fichiers de sauvegarde dans un compartiment Amazon S3.

Pour plus d'informations sur la création et le chargement d'un fichier dans un compartiment Amazon S3, consultez Mise en route sur Amazon Simple Storage Service dans le Guide de démarrage Amazon S3.

Utilisation de sauvegardes incrémentielles avec Percona XtraBackup

Amazon Aurora MySQL prend en charge les sauvegardes complètes et incrémentielles créées à l'aide de XtraBackup Percona. Si vous utilisez déjà Percona XtraBackup pour effectuer des sauvegardes complètes et incrémentielles de vos fichiers de base de données MySQL, vous n'avez pas besoin de créer une sauvegarde complète et de télécharger les fichiers de sauvegarde sur Amazon S3. Au lieu de cela, vous pouvez économiser beaucoup de temps en copiant vos fichiers et répertoires de sauvegarde existants pour vos sauvegardes complètes et incrémentielles dans un compartiment Amazon S3. Pour plus d'informations, consultez Création d'une sauvegarde incrémentielle (langue française non garantie) sur le site web de Percona.

Lorsque vous copiez les fichiers existants des sauvegardes complètes et incrémentielles dans un compartiment Amazon S3, vous devez copier de façon récursive le contenu du répertoire de base. Ce contenu inclut la sauvegarde complète, ainsi que tous les fichiers et répertoires des sauvegardes incrémentielles. Cette copie doit conserver la structure de répertoire dans le compartiment Amazon S3. Aurora effectue une itération sur l'ensemble des fichiers et répertoires. Aurora utilise le fichier `xtrabackup-checkpoints` inclus avec chaque sauvegarde incrémentielle pour identifier le répertoire de base et ordonner les sauvegardes incrémentielles selon leur plage de numéros de séquence de journal.

Pour plus d'informations sur la création et le chargement d'un fichier dans un compartiment Amazon S3, consultez Mise en route sur Amazon Simple Storage Service dans le Guide de démarrage Amazon S3.

Considérations relatives à la sauvegarde

Aurora ne prend pas en charge les sauvegardes partielles créées à l'aide de Percona XtraBackup. Vous ne pouvez pas utiliser les options suivantes pour créer une sauvegarde partielle lorsque vous sauvegardez les fichiers source pour votre base de données : `--tables`, `--tables-exclude`, `--tables-file`, `--databases`, `--databases-exclude` ou `--databases-file`.

Pour plus d'informations sur la sauvegarde de votre base de données avec Percona XtraBackup, consultez Percona XtraBackup - Documentation et utilisation de journaux binaires sur le site Web de Percona.

Aurora prend en charge les sauvegardes incrémentielles créées à l'aide de Percona XtraBackup. Pour plus d'informations, consultez Création d'une sauvegarde incrémentielle (langue française non garantie) sur le site web de Percona.

Aurora utilise vos fichiers de sauvegarde sur la base des noms de fichier. Veillez à nommer vos fichiers de sauvegarde avec l'extension de fichier appropriée basée sur le format de fichier, par exemple, `.xbstream` pour les fichiers stockés en utilisant le format Percona xbstream.

Aurora utilise vos fichiers de sauvegarde dans l'ordre alphabétique, ainsi que l'ordre numérique naturel. Utilisez toujours l'option `split` lorsque vous émettez la commande `xtrabackup` pour vous assurer que vos fichiers de sauvegarde sont écrits et nommés dans l'ordre approprié.

Amazon S3 limite la taille d'un fichier chargé vers un compartiment Amazon S3 à 5 To. Si les données de sauvegarde de votre base de données dépassent 5 To, utilisez la commande `split` pour diviser les fichiers de sauvegarde en plusieurs fichiers de moins de 5 To chacun.

Aurora limite le nombre de fichiers source chargés dans un compartiment Amazon S3 à 1 million de fichiers. Dans certains cas, si les données de sauvegarde de votre base de données, y compris toutes les sauvegardes complètes et incrémentielles, comprennent un grand nombre de fichiers. Le cas échéant, utilisez un fichier tarball (.tar.gz) pour stocker les fichiers des sauvegardes complètes et incrémentielles dans le compartiment Amazon S3.

Lorsque vous chargez un fichier dans compartiment Amazon S3, vous pouvez utiliser le chiffrement côté serveur pour chiffrer vos données. Vous pouvez ensuite restaurer un cluster de bases de données Amazon Aurora MySQL à partir de ces fichiers chiffrés. Amazon Aurora MySQL peut restaurer un cluster de bases de données avec des fichiers chiffrés à l'aide des types de chiffrement côté serveur suivants :

- Chiffrement côté serveur avec des clés gérées par Amazon S3 (SSE-S3) : chaque objet est chiffré à l'aide d'une clé unique utilisant un chiffrement multi-facteur fort.
- Chiffrement côté serveur avec clés AWS KMS gérées (SSE-KMS) : similaire au SSE-S3, mais vous avez la possibilité de créer et de gérer vous-même les clés de chiffrement, ainsi que d'autres différences.

Pour plus d'informations sur l'utilisation du chiffrement côté serveur lors du chargement de fichiers dans un compartiment Amazon S3, consultez Protection des données à l'aide d'un chiffrement côté serveur dans le Manuel du développeur Amazon S3.

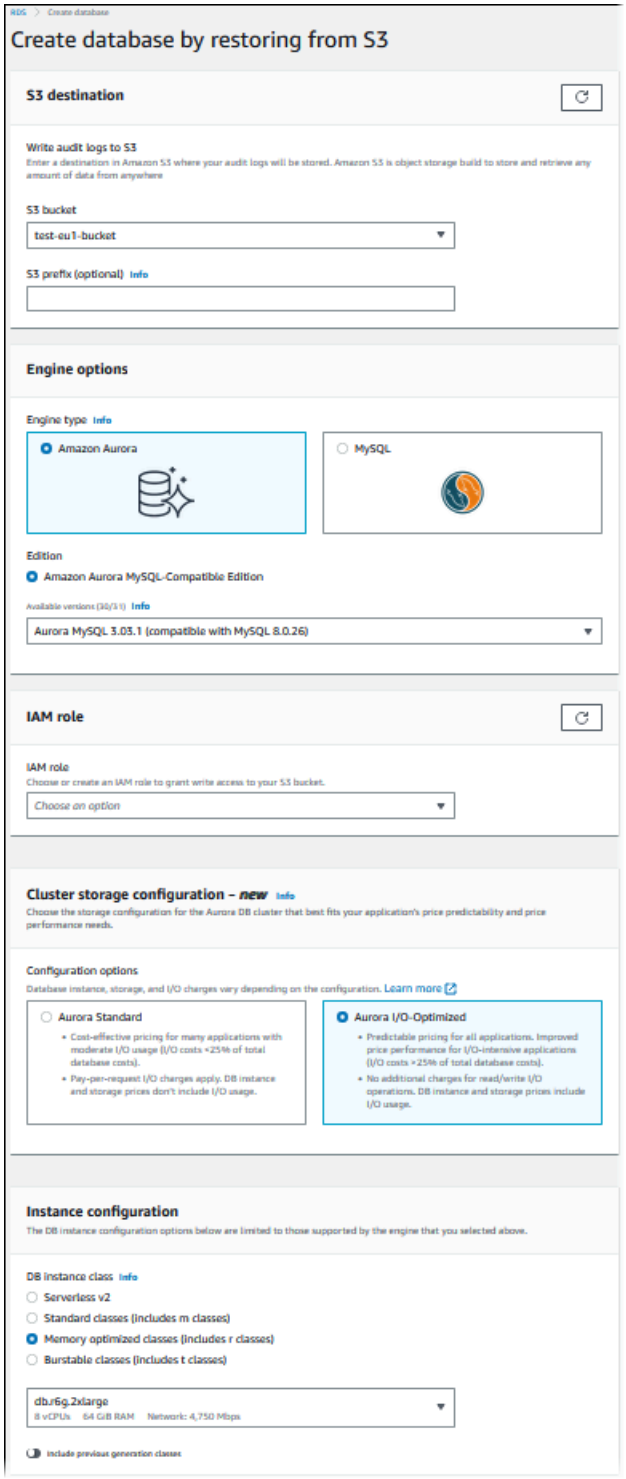
Restauration d'un cluster de bases de données Amazon Aurora MySQL à partir d'un compartiment Amazon S3

Vous pouvez restaurer vos fichiers de sauvegarde à partir de votre compartiment Amazon S3 pour créer un nouveau cluster de bases de données Amazon Aurora MySQL à l'aide de la console Amazon RDS.

Pour restaurer un cluster de bases de données Amazon Aurora MySQL à partir de fichiers d'un compartiment Amazon S3

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le coin supérieur droit de la console Amazon RDS, choisissez la AWS région dans laquelle vous souhaitez créer votre cluster de base de données. Choisissez la même AWS région que le compartiment Amazon S3 qui contient la sauvegarde de votre base de données.
3. Dans le panneau de navigation, choisissez Bases de données, puis Restaurer à partir de S3.
4. Choisissez Restaurer à partir de S3.

   La page Créer une base de données par restauration à partir de S3 s'affiche.



5. Sous Destination S3 :

   a. Sélectionnez le compartiment S3 qui contient vos fichiers de sauvegarde.

   b. (Facultatif) Pour Préfixe du chemin de dossier S3, saisissez un préfixe de chemin de fichier pour les fichiers stockés dans votre compartiment Amazon S3.

Si vous ne spécifiez pas de préfixe, RDS crée votre cluster/instance de base de données à l'aide de tous les fichiers et dossiers du dossier racine du compartiment S3. Si vous indiquez un préfixe, RDS crée votre instance de base de données à l'aide des fichiers et dossiers du compartiment S3 pour lesquels le chemin du fichier commence par le préfixe spécifié.

Par exemple, supposons que vous stockez vos fichiers de sauvegarde sur S3 dans un sous-dossier appelé « sauvegardes » et que vous avez plusieurs ensembles de fichiers de sauvegarde, chacun dans son propre répertoire (gzip_backup1, gzip_backup2, etc.). Dans ce cas, vous devez spécifier un préfixe sauvegardes/gzip_backup1 pour restaurer les fichiers dans le dossier gzip_backup1.

6. Sous Options du moteur :

   a. Pour Engine type (Type de moteur), sélectionnez Amazon Aurora.

   b. Pour Version, sélectionnez la version Aurora MySQL du moteur de votre instance de base de données restaurée.

7. Dans le champ Rôle IAM, vous pouvez choisir un rôle IAM existant.

8. (Facultatif) Vous pouvez également créer un nouveau rôle IAM en choisissant Créer un nouveau rôle. Dans ce cas :

   a. Entrez le Nom du rôle IAM.

   b. Déterminez si vous souhaitez Autoriser l'accès à la clé KMS :

      - Si vous n'avez pas chiffré les fichiers de sauvegarde, choisissez Non.
      - Si vous avez chiffré les fichiers de sauvegarde avec AES-256 (SSE-S3) lors de leur chargement dans Amazon S3, choisissez Non. Dans ce cas, les données sont chiffrées automatiquement.
      - Si vous avez chiffré les fichiers de sauvegarde avec un chiffrement côté serveur AWS KMS (SSE-KMS) lorsque vous les avez chargés sur Amazon S3, choisissez Oui. Ensuite, choisissez la clé KMS appropriée pour AWS KMS key.

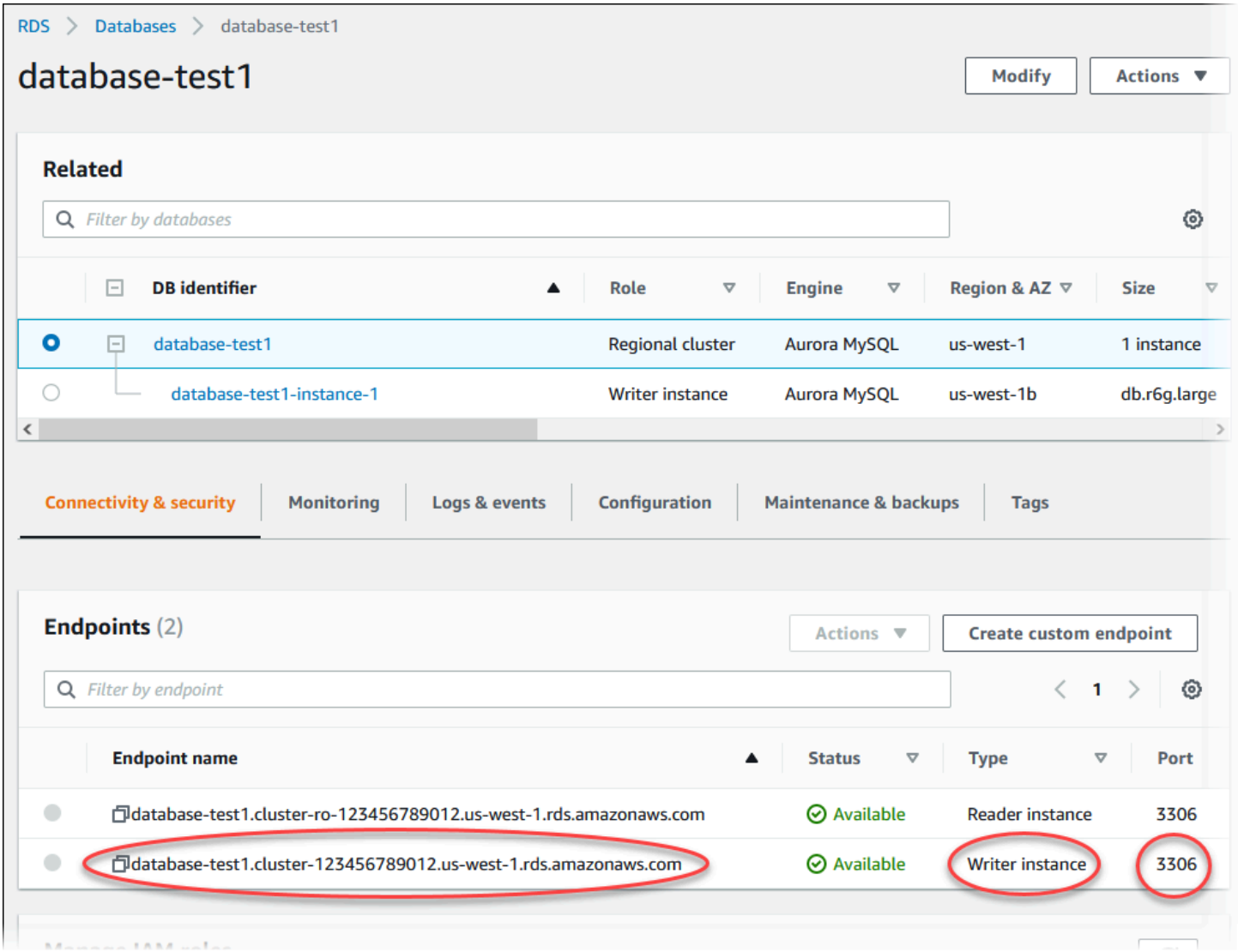
        AWS Management Consolecrée une politique IAM qui permet à Aurora de déchiffrer les données.

Pour plus d'informations, consultez Protection des données à l'aide d'un chiffrement côté serveur dans le Manuel du développeur Amazon S3.

9. Choisissez les paramètres de votre cluster de bases de données, tels que la configuration du stockage pour le cluster de bases de données, la classe d'instance de la base de données, l'identifiant du cluster de bases de données et les informations d'identification de connexion. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour les clusters de bases de données Aurora.
10. Personnalisez les autres paramètres de votre cluster de bases de données Aurora MySQL selon vos besoins.
11. Sélectionnez Create database (Créer une base de données) pour lancer votre instance de base de données Aurora.

Sur la console Amazon RDS, la nouvelle instance de base de données s'affiche dans la liste des instances de bases de données. L'instance de base de données a le statut creating (création en cours) jusqu'à ce qu'elle soit créée et prête à l'emploi. Lorsque l'état devient available, vous pouvez vous connecter à l'instance principale de votre cluster DB. En fonction du stockage et de la classe d'instance de base de données alloués, la mise à disposition de la nouvelle instance de base de données peut nécessiter plusieurs minutes.

Pour afficher le cluster nouvellement créé, choisissez la vue Bases de données dans la console Amazon RDS et choisissez le cluster de bases de données. Pour plus d'informations, consultez Affichage d'un cluster de bases de données Amazon Aurora.



Notez le port et le point de terminaison enregistreur du cluster de bases de données. Utilisez le point de terminaison enregistreur et le port du cluster de bases de données dans vos chaînes de connexion JDBC et ODBC pour toute application qui exécute des opérations de lecture et d'écriture.

Synchronisation du cluster de bases de données Amazon Aurora MySQL sur la base de données MySQL avec la réplication

Pendant la migration, pour limiter ou éviter toute durée d'indisponibilité, vous pouvez répliquer les transactions validées sur votre base de données MySQL sur votre cluster de bases de données Aurora MySQL. La réplication permet au cluster de bases de données d'être synchrone avec les transactions de la base de données MySQL traitées pendant la migration. Une fois que le cluster de bases de données est totalement synchronisé, vous pouvez arrêter la réplication et terminer la migration vers Aurora MySQL.

Rubriques

Configuration de votre base de données MySQL externe et de votre cluster de bases de données Aurora MySQL pour la réplication chiffrée

Pour répliquer des données en toute sécurité, vous pouvez utiliser la réplication chiffrée.

> **Note**
>
> Si vous n'avez pas besoin d'utiliser la réplication chiffrée, vous pouvez ignorer ces étapes et passer aux instructions de Synchronisation du cluster de bases de données Amazon Aurora MySQL sur la base de données MySQL externe.

Pour pouvoir utiliser la réplication chiffrée, vous devez impérativement disposer des éléments suivants :

- Le protocole SSL doit être activé sur la base de données source MySQL principale.
- Une clé client et un certificat client doivent être préparés pour le cluster de bases de données Aurora MySQL.

Pendant la réplication chiffrée, le cluster de bases de données Aurora MySQL agit comme client du serveur de base de données MySQL. Les certificats et les clés privées du client Aurora MySQL sont au format .pem dans les fichiers.

Pour configurer votre base de données MySQL externe et votre cluster de bases de données Aurora MySQL pour la réplication chiffrée

1. Vérifiez bien que vous êtes prêt à procéder à la réplication chiffrée :

   - Si le protocole SSL n'est pas activé sur la base de données source MySQL principale et que vous ne disposez pas d'une clé client et d'un certificat client prêts, activez le protocole SSL sur le serveur de base de données MySQL et générez la clé client et le certificat client requis.

- Si le protocole SSL est activé sur la base de données source principale, fournissez une clé et un certificat client pour le cluster de bases de données Aurora MySQL. En leur absence, générez une nouvelle clé et un nouveau certificat pour le cluster de bases de données Aurora MySQL. Pour signer le certificat client, vous devez disposer de la clé d'autorité de certification utilisée pour configurer le protocole SSL sur la base de données source MySQL principale.

Pour plus d'informations, consultez Création de certificats et clés SSL à l'aide d'openssl dans la documentation MySQL.

Vous avez besoin du certificat de l'autorité de certification, de la clé client et du certificat client.

2. Connectez-vous au cluster de bases de données Aurora MySQL en tant qu'utilisateur principal à l'aide du protocole SSL.

   Pour plus d'informations sur la connexion à un cluster de bases de données Aurora MySQL avec le protocole SSL, consultez Connexions TLS aux clusters de bases de données Aurora MySQL.

3. Exécutez la procédure stockée mysql.rds_import_binlog_ssl_material pour importer les informations SSL dans le cluster de bases de données Aurora MySQL.

   Pour le paramètre `ssl_material_value`, insérez les informations des fichiers au format .pem pour le cluster de bases de données Aurora MySQL dans les données utiles JSON correctes.

   L'exemple suivant importe des informations SSL dans un cluster de bases de données Aurora MySQL. Dans les fichiers au format .pem, le code du corps est généralement plus long que le code du corps affiché dans l'exemple.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
```

```
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

Pour plus d'informations, consultez mysql.rds_import_binlog_ssl_material et Connexions TLS aux clusters de bases de données Aurora MySQL.

> **Note**
>
> Après l'exécution de la procédure, les secrets sont stockés dans les fichiers. Pour supprimer les fichiers ultérieurement, vous pouvez exécuter la procédure stockée mysql.rds_remove_binlog_ssl_material.

Synchronisation du cluster de bases de données Amazon Aurora MySQL sur la base de données MySQL externe

Vous pouvez synchroniser votre cluster de bases de données Amazon Aurora MySQL sur la base de données MySQL à l'aide de la réplication.

Pour synchroniser votre cluster de bases de données Aurora MySQL sur la base de données MySQL à l'aide de la réplication

1. Vérifiez que le fichier /etc/my.cnf de la base de données MySQL externe dispose des entrées correspondantes.

   Si la réplication chiffrée n'est pas obligatoire, assurez-vous que la base de données MySQL externe est démarrée avec des journaux binaires (binlogs) activés et le protocole SSL désactivé. Les entrées correspondantes dans le fichier /etc/my.cnf pour les données non chiffrées sont les suivantes.

   ```
   log-bin=mysql-bin
   server-id=2133421
   innodb_flush_log_at_trx_commit=1
   sync_binlog=1
   ```

Si la réplication chiffrée est obligatoire, assurez-vous que la base de données MySQL externe est démarrée avec le protocole SSL et les journaux binaires activés. Les entrées dans le fichier /etc/my.cnf incluent les emplacements de fichier .pem pour le serveur de base de données MySQL.

```
log-bin=mysql-bin
server-id=2133421
innodb_flush_log_at_trx_commit=1
sync_binlog=1

# Setup SSL.
ssl-ca=/home/sslcerts/ca.pem
ssl-cert=/home/sslcerts/server-cert.pem
ssl-key=/home/sslcerts/server-key.pem
```

Vous pouvez vérifier que SSL est activé avec la commande suivante.

```
mysql> show variables like 'have_ssl';
```

Votre sortie doit ressembler à ce qui suit.

```
+~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~+~-~-~-~-~-~-~-~+
| Variable_name | Value |
+~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~+~-~-~-~-~-~-~-~+
| have_ssl      | YES   |
+~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~+~-~-~-~-~-~-~-~+
1 row in set (0.00 sec)
```

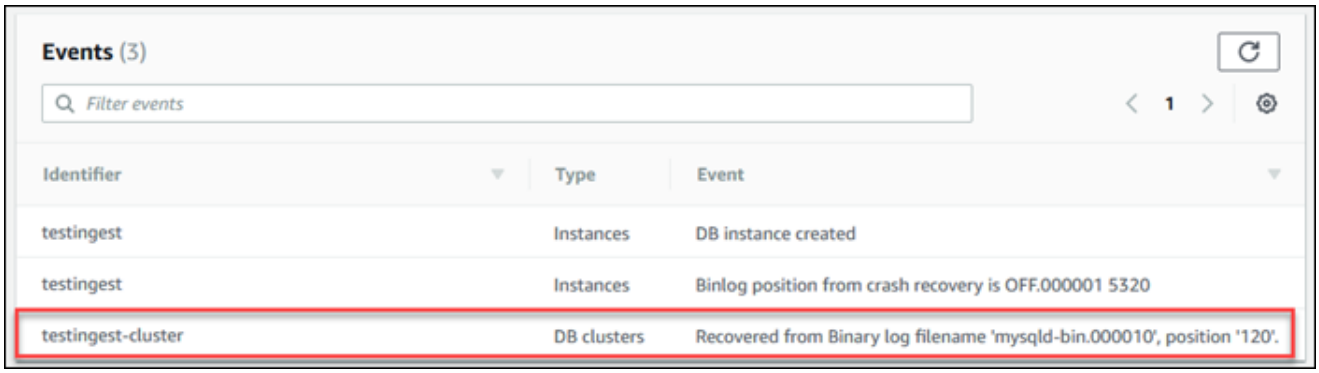
2. Déterminez la position de début de votre journal binaire pour la réplication. Vous spécifiez la position pour démarrer la réplication à une étape ultérieure.

   En utilisant le AWS Management Console

   a. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.

   b. Dans le panneau de navigation, sélectionnez Events.

c. Dans la liste Événements, notez la position dans l'événement Recovered from Binary log filename (Récupéré à partir du nom de fichier journal binaire).



En utilisant le AWS CLI

Vous pouvez également obtenir le nom et la position du fichier binlog à l'aide de la commande describe-eventsAWS CLI. Ce qui suit présente un exemple de commande `describe-events`.

```
PROMPT> aws rds describe-events
```

Dans la sortie, identifiez l'événement qui affiche la position du journal binaire.

3. Tandis que vous êtes connecté à la base de données MySQL externe, créez un utilisateur à utiliser pour la réplication. Ce compte est utilisé exclusivement pour la réplication et doit être limité à votre domaine pour améliorer la sécurité. Voici un exemple de.

```
mysql> CREATE USER '<user_name>'@'<domain_name>' IDENTIFIED BY '<password>';
```

L'utilisateur nécessite les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE`. Accordez ces privilèges à l'utilisateur.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO
 '<user_name>'@'<domain_name>';
```

Si vous avez besoin d'utiliser la réplication chiffrée, demandez des connexions SSL à l'utilisateur de la réplication. Par exemple, vous pouvez utiliser l'instruction suivante pour demander les connexions SSL sur le compte d'utilisateur *<user_name>*.

```
GRANT USAGE ON *.* TO '<user_name>'@'<domain_name>' REQUIRE SSL;
```

> **Note**
>
> Si `REQUIRE SSL` n'est pas inclus, la connexion de réplication peut revenir de façon silencieuse à une connexion non chiffrée.

4. Dans la console Amazon RDS, ajoutez l'adresse IP du serveur qui héberge la base de données MySQL externe au groupe de sécurité VPC du cluster de bases de données Aurora MySQL. Pour plus d'informations sur la modification d'un groupe de sécurité de VPC, consultez Groupes de sécurité pour votre VPC dans le Guide de l'utilisateur Amazon Virtual Private Cloud.

   Il se peut aussi que vous ayez besoin de configurer votre réseau local pour autoriser les connexions à partir de l'adresse IP de votre cluster de bases de données Aurora MySQL, de telle sorte qu'elle puisse communiquer avec votre base de données MySQL externe. Pour rechercher l'adresse IP du cluster de bases de données Aurora MySQL, utilisez la commande `host`.

   ```
   host <db_cluster_endpoint>
   ```

   Le nom d'hôte est le nom DNS du point de terminaison du cluster de bases de données Aurora MySQL.

5. Activez la réplication des journaux binaires en exécutant la procédure stockée mysql.rds_reset_external_master (Aurora MySQL version 2) ou mysql.rds_reset_external_source (Aurora MySQL version 3). Cette procédure stockée possède la syntaxe suivante.

   ```
   CALL mysql.rds_set_external_master (
     host_name
     , host_port
     , replication_user_name
     , replication_user_password
     , mysql_binary_log_file_name
     , mysql_binary_log_file_location
     , ssl_encryption
   );

   CALL mysql.rds_set_external_source (
   ```

```
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Pour obtenir des informations sur les paramètres, consultez mysql.rds_reset_external_master (Aurora MySQL version 2) et mysql.rds_reset_external_source (Aurora MySQL version 3).

Pour `mysql_binary_log_file_name` et `mysql_binary_log_file_location`, utilisez la position dans l'événement Recovered from Binary log filename (Récupéré à partir du nom de fichier journal binaire) que vous avez notée précédemment.

Si les données du cluster de bases de données Aurora MySQL ne sont pas chiffrées, le paramètre `ssl_encryption` doit être défini sur `0`. Si les données sont chiffrées, le paramètre `ssl_encryption` doit être défini sur `1`.

L'exemple suivant présente l'exécution de la procédure pour un cluster de bases de données Aurora MySQL dont les données sont chiffrées.

```
CALL mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'password',
  'mysql-bin.000010',
  120,
  1);

CALL mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'password',
  'mysql-bin.000010',
  120,
  1);
```

Cette procédure stockée définit les paramètres que le cluster de bases de données Aurora MySQL utilise pour se connecter à la base de données MySQL externe et pour lire son journal binaire. Si les données sont chiffrées, il télécharge également le certificat de l'autorité de certification, le certificat du client et la clé du client sur le disque local.

6. Démarrez la réplication des journaux binaires en exécutant la procédure stockée mysql.rds_start_replication.

```
CALL mysql.rds_start_replication;
```

7. Contrôlez le décalage entre le cluster de bases de données Aurora MySQL et la base de données principale de réplication MySQL. Pour cela, connectez-vous au cluster de bases de données Aurora MySQL et exécutez la commande suivante.

```
Aurora MySQL version 2:
SHOW SLAVE STATUS;

Aurora MySQL version 3:
SHOW REPLICA STATUS;
```

Dans la sortie de la commande, le champ `Seconds Behind Master` indique à quelle distance le cluster de bases de données Aurora MySQL se trouve du principal MySQL. Lorsque cette valeur est `0` (zéro), le cluster de bases de données Aurora MySQL s'est synchronisé avec le principal, et vous pouvez passer à l'étape suivante pour arrêter la réplication.

8. Connectez-vous à la base de données principale de réplication MySQL et arrêtez la réplication. Pour ce faire, exécutez la procédure stockée mysql.rds_stop_replication.

```
CALL mysql.rds_stop_replication;
```

Réduction du délai de migration physique vers Amazon Aurora MySQL

Vous pouvez apporter les modifications de base de données suivantes pour accélérer le processus de migration d'une base de données vers Amazon Aurora MySQL.

> **Important**
>
> Veillez à effectuer ces mises à jour sur une copie d'une base de données de production, plutôt que sur une base de données de production. Vous pouvez ensuite sauvegarder la copie et la restaurer sur votre cluster de bases de données Aurora MySQL pour éviter toute interruption de service sur votre base de données de production.

Types de table non pris en charge

Aurora MySQL prend uniquement en charge le moteur InnoDB pour les tables de base de données. Si vous avez des tables MyISAM dans votre base de données, elles doivent être converties avant la migration vers Aurora MySQL. Le processus de conversion nécessite un espace supplémentaire pour la conversion de MyISAM en InnoDB pendant la procédure de migration.

Pour réduire le risque d'un espace insuffisant ou pour accélérer le processus de migration, convertissez toutes vos tables MyISAM en tables InnoDB avant de les migrer. La taille de la table InnoDB obtenue est équivalente à celle requise par Aurora MySQL pour cette table. Pour convertir une table MyISAM en InnoDB, exécutez la commande suivante :

```
ALTER TABLE schema.table_name engine=innodb, algorithm=copy;
```

Aurora MySQL ne prend pas en charge les tables ni les pages compressées, à savoir les tables créées avec `ROW_FORMAT=COMPRESSED` ou `COMPRESSION = {"zlib"|"lz4"}`.

Pour réduire le risque d'un espace insuffisant ou pour accélérer le processus de migration, développez vos tables compressées en définissant `ROW_FORMAT` sur `DEFAULT`, `COMPACT`, `DYNAMIC` ou `REDUNDANT`. Pour les pages compressées, définissez `COMPRESSION="none"`.

Pour plus d'informations, consultez Formats de ligne InnoDB et Compression des pages et des tables InnoDB dans la documentation MySQL.

Vous pouvez utiliser le script SQL suivant sur votre instance de base de données MySQL existante pour afficher les tables de votre base de données qui sont des tables MyISAM ou des tables compressées.

```
-- This script examines a MySQL database for conditions that block
-- migrating the database into Aurora MySQL.
-- It must be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.
```

```
-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6 or higher. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
    as major_minor
  ) as T
where major_minor <> 506;


-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
```

```
  or
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

Comptes d'utilisateur avec des privilèges non pris en charge

Les comptes d'utilisateur dotés de privilèges non pris en charge par Aurora MySQL sont importés sans les privilèges non pris en charge. Pour obtenir la liste des privilèges pris en charge, consultez Modèle de privilège basé sur les rôles.

Vous pouvez exécuter la requête SQL suivante sur votre base de données source pour répertorier les comptes d'utilisateur dotés de privilèges non pris en charge.

```
SELECT
    user,
    host
FROM
    mysql.user
WHERE
    Shutdown_priv = 'y'
    OR File_priv = 'y'
    OR Super_priv = 'y'
    OR Create_tablespace_priv = 'y';
```

Privilèges dynamiques dans Aurora MySQL version 3

Les privilèges dynamiques ne sont pas importés. Aurora MySQL version 3 prend en charge les privilèges dynamiques suivants.

```
'APPLICATION_PASSWORD_ADMIN',
'CONNECTION_ADMIN',
'REPLICATION_APPLIER',
'ROLE_ADMIN',
'SESSION_VARIABLES_ADMIN',
'SET_USER_ID',
'XA_RECOVER_ADMIN'
```

L'exemple de script suivant accorde les privilèges dynamiques pris en charge aux comptes d'utilisateur dans le cluster de bases de données Aurora MySQL.

```
-- This script finds the user accounts that have Aurora MySQL supported dynamic
 privileges
-- and grants them to corresponding user accounts in the Aurora MySQL DB cluster.

/home/ec2-user/opt/mysql/8.0.26/bin/mysql -uusername -pxxxxx -P8026 -h127.0.0.1 -BNe
 "SELECT
  CONCAT('GRANT ', GRANTS, ' ON *.* TO ', GRANTEE ,';') AS grant_statement
  FROM (select GRANTEE, group_concat(privilege_type) AS GRANTS FROM
 information_schema.user_privileges
      WHERE privilege_type IN (
        'APPLICATION_PASSWORD_ADMIN',
        'CONNECTION_ADMIN',
        'REPLICATION_APPLIER',
        'ROLE_ADMIN',
        'SESSION_VARIABLES_ADMIN',
        'SET_USER_ID',
        'XA_RECOVER_ADMIN')
      AND GRANTEE NOT IN (\"'mysql.session'@'localhost'\",
\"'mysql.infoschema'@'localhost'\",\"'mysql.sys'@'localhost'\") GROUP BY GRANTEE)
      AS PRIVGRANTS; " | /home/ec2-user/opt/mysql/8.0.26/bin/mysql -u master_username -
p master_password -h DB_cluster_endpoint
```

Objets stockés avec ’rdsadmin’@’localhost’ comme définisseur

Les fonctions, les procédures, les vues, les événements et les déclencheurs avec `'rdsadmin'@'localhost'` comme définisseur ne sont pas importés.

Vous pouvez utiliser le script SQL suivant sur votre base de données MySQL source pour répertorier les objets stockés qui possèdent le définisseur non pris en charge.

```
-- This SQL query lists routines with `rdsadmin`@`localhost` as the definer.

SELECT
    ROUTINE_SCHEMA,
    ROUTINE_NAME
FROM
    information_schema.routines
WHERE
    definer = 'rdsadmin@localhost';

-- This SQL query lists triggers with `rdsadmin`@`localhost` as the definer.

SELECT
```

```
    TRIGGER_SCHEMA,
    TRIGGER_NAME,
    DEFINER
FROM
    information_schema.triggers
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists events with `rdsadmin`@`localhost` as the definer.

SELECT
    EVENT_SCHEMA,
    EVENT_NAME
FROM
    information_schema.events
WHERE
    DEFINER = 'rdsadmin@localhost';

-- This SQL query lists views with `rdsadmin`@`localhost` as the definer.
SELECT
    TABLE_SCHEMA,
    TABLE_NAME
FROM
    information_schema.views
WHERE
    DEFINER = 'rdsadmin@localhost';
```

Migration logique de MySQL vers Amazon Aurora MySQL à l'aide de mysqldump

Étant donné qu'Amazon Aurora MySQL est une base de données compatible avec MySQL, vous pouvez utiliser l'utilitaire `mysqldump` pour copier les données de votre base de données MySQL ou l'utilitaire `mariadb-dump` pour copier les données de votre base de données MariaDB vers un cluster de bases de données Aurora MySQL existant.

Pour découvrir comment procéder avec des bases de données MySQL ou MariaDB très volumineuses, consultez les rubriques suivantes dans le Guide de l'utilisateur Amazon Relational Database Service :

- MySQL : Importation de données vers une base de données Amazon RDS for MySQL avec une durée d'indisponibilité réduite
- MariaDB : Importation de données vers une base de données Amazon RDS for MariaDB avec une durée d'indisponibilité réduite

Pour les bases de données MySQL ou MariaDB comportant de plus petits volumes de données, consultez les rubriques suivantes dans le Guide de l'utilisateur Amazon Relational Database Service :

- MySQL — Importation de données à partir d'une base de données MySQL externe vers une instance de base de données Amazon RDS for MySQL
- MariaDB — Importation de données à partir d'une base de données MariaDB vers une instance de base de données Amazon RDS for MariaDB

Migration de données à partir d'une instance de base de données RDS MySQL vers un cluster de base de données Amazon Aurora MySQL

Vous pouvez migrer (copier) les données directement d'un cluster de base de données Amazon Aurora MySQL vers une instance de base de données RDS for MySQL.

Rubriques

> **Note**
>
> Étant donné qu'Amazon Aurora MySQL est compatible avec MySQL, vous pouvez migrer les données de votre base de données MySQL en configurant la réplication entre votre base de données MySQL et un cluster de base de données Amazon Aurora MySQL. Pour de plus amples informations, consultez Réplication avec Amazon Aurora.

Migration d’un instantané RDS for MySQL vers Aurora

Vous pouvez migrer un instantané de bases de données d’une instance de base de données RDS for MySQL pour créer un cluster de bases de données Aurora MySQL. Le nouveau cluster de bases de données Aurora MySQL est rempli avec les données de l’instance de base de données RDS for MySQL initiale. L’instantané de base de données doit avoir été effectué à partir d’une instance de base de données Amazon RDS exécutant une version MySQL compatible avec Aurora MySQL.

Vous pouvez migrer un instantané de base de données manuel ou automatique. Après que le cluster DB a été créé, vous pouvez créer des réplicas Aurora facultatifs.

> **Note**
>
> Vous pouvez également migrer une instance de base de données RDS for MySQL vers un cluster de bases de données Aurora MySQL en créant un réplica en lecture Aurora de votre instance de base de données RDS for MySQL source. Pour plus d’informations, consultez Migration des données d’une instance de base de données RDS for MySQL vers un cluster de bases de données Amazon Aurora MySQL à l’aide d’un réplica en lecture Aurora.

Vous ne pouvez pas migrer de certaines anciennes versions de MySQL 8.0, notamment des versions 8.0.11, 8.0.13 et 8.0.15, vers Aurora MySQL 3.05 et versions ultérieures. Nous vous recommandons de passer à MySQL version 8.0.28 avant de procéder à la migration.

Les étapes générales à suivre sont les suivantes :

1. Déterminez la quantité d'espace à allouer à votre cluster de bases de données Aurora MySQL. Pour plus d'informations, consultez De quel espace ai-je besoin ?.
2. Utilisez la console pour créer l'instantané dans la région AWS où l'instance Amazon RDS MySQL se trouve. Pour plus d'informations sur la création d'un instantané de base de données, consultez Création d'un instantané de base de données.
3. Si l'instantané de bases de données ne se trouve pas dans la même région AWS que votre cluster de bases de données, utilisez la console Amazon RDS pour copier l'instantané de base de données dans cette région AWS. Pour plus d'informations sur la copie d'un instantané de base de données, consultez Copie d'un instantané.
4. Utilisez la console pour migrer l'instantané de bases de données et créer un cluster de bases de données Aurora MySQL avec les mêmes bases de données que l'instance de base de données MySQL initiale.

⚠ Warning

Amazon RDS limite chaque compte AWS à une copie d'instantané à la fois dans chaque région AWS.

De quel espace ai-je besoin ?

Lorsque vous migrez un instantané d'une instance de base de données MySQL vers un cluster de bases de données Aurora MySQL, Aurora utilise un volume Amazon Elastic Block Store (Amazon EBS) pour mettre en forme les données de l'instantané avant de les migrer. Dans certains cas, un espace supplémentaire est nécessaire pour mettre en forme les données de la migration.

Les tables autres que les tables MyISAM et qui ne sont pas compressées peuvent atteindre jusqu'à 16 To. Si vous avez des tables MyISAM, Aurora doit utiliser un espace supplémentaire du volume pour rendre ces tables compatibles avec Aurora MySQL. Si vous avez compressé les tables, Aurora doit utiliser un espace supplémentaire du volume pour développer ces tables avant de les stocker sur

le volume de cluster Aurora. En raison de cette exigence d'espace supplémentaire, vous devez vous assurer qu'aucune des tables MyISAM et compressées, migrées depuis votre instance de base de données MySQL, ne dépasse 8 To en taille.

Réduction de la quantité d'espace requise pour migrer les données vers Amazon Aurora MySQL

Il se peut que vous souhaitiez modifier votre schéma de base de données avant la migration vers Amazon Aurora. Cette modification peut être utile dans les cas suivants :

- Vous voulez accélérer le processus de migration.
- Vous avez un doute quant à la quantité d'espace que vous devez allouer.
- Vous avez tenté de migrer vos données et la migration a échoué en raison d'un manque d'espace alloué.

Vous pouvez effectuer les modifications suivantes pour améliorer le processus de migration d'une base de données vers Amazon Aurora.

> **Important**
>
> Veillez à effectuer ces mises à jour sur une nouvelle instance de base de données restaurée à partir d'un instantané d'une base de données de production, plutôt que sur une instance de production. Vous pouvez ensuite migrer les données depuis l'instantané de votre nouvelle instance de base de données vers votre cluster de bases de données Aurora pour éviter toute interruption de service sur votre base de données de production.

Type de table	Limitation ou instruction
Tables MyISAM	Aurora MySQL prend uniquement en charge les tables InnoDB. Si vous avez des tables MyISAM dans votre base de données, elles doivent être converties avant d'être migrées vers Aurora MySQL. Le processus de conversion nécessite un espace supplémentaire pour la conversion de MyISAM en InnoDB pendant la procédure de migration. Pour réduire le risque d'un espace insuffisant ou pour accélérer le processus de migration, convertissez toutes vos tables MyISAM

Type de table	Limitation ou instruction
	en tables InnoDB avant de les migrer. La taille de la table InnoDB obtenue est équivalente à celle requise par Aurora MySQL pour cette table. Pour convertir une table MyISAM en InnoDB, exécutez la commande suivante : `alter table <schema>.<table_name> engine=innodb, algorithm=copy;`
Tables compressées	Aurora MySQL ne prend pas en charge les tables compressées (c'est-à-dire les tables créées avec `ROW_FORMAT=COMPRESSED`). Pour réduire le risque d'un espace insuffisant ou pour accélérer le processus de migration, développez vos tables compressées en définissant `ROW_FORMAT` sur `DEFAULT`, `COMPACT`, `DYNAMIC` ou `REDUNDANT`. Pour plus d'informations, consultez Formats de ligne InnoDB dans la documentation sur MySQL.

Vous pouvez utiliser le script SQL suivant sur votre instance de base de données MySQL existante pour afficher les tables de votre base de données qui sont des tables MyISAM ou des tables compressées.

```
-- This script examines a MySQL database for conditions that block
-- migrating the database into Amazon Aurora.
-- It needs to be run from an account that has read permission for the
-- INFORMATION_SCHEMA database.

-- Verify that this is a supported version of MySQL.

select msg as `==> Checking current version of MySQL.`
from
  (
  select
    'This script should be run on MySQL version 5.6 or higher. ' +
    'Earlier versions are not supported.' as msg,
    cast(substring_index(version(), '.', 1) as unsigned) * 100 +
      cast(substring_index(substring_index(version(), '.', 2), '.', -1)
      as unsigned)
```

```
    as major_minor
  ) as T
where major_minor <> 506;


-- List MyISAM and compressed tables. Include the table size.

select concat(TABLE_SCHEMA, '.', TABLE_NAME) as `==> MyISAM or Compressed Tables`,
round(((data_length + index_length) / 1024 / 1024), 2) "Approx size (MB)"
from INFORMATION_SCHEMA.TABLES
where
  ENGINE <> 'InnoDB'
  and
  (
    -- User tables
    TABLE_SCHEMA not in ('mysql', 'performance_schema',
                         'information_schema')
    or
    -- Non-standard system tables
    (
      TABLE_SCHEMA = 'mysql' and TABLE_NAME not in
        (
          'columns_priv', 'db', 'event', 'func', 'general_log',
          'help_category', 'help_keyword', 'help_relation',
          'help_topic', 'host', 'ndb_binlog_index', 'plugin',
          'proc', 'procs_priv', 'proxies_priv', 'servers', 'slow_log',
          'tables_priv', 'time_zone', 'time_zone_leap_second',
          'time_zone_name', 'time_zone_transition',
          'time_zone_transition_type', 'user'
        )
    )
  )
  or
  (
    -- Compressed tables
       ROW_FORMAT = 'Compressed'
  );
```

Le script produit un résultat similaire à celui de l'exemple suivant. L'exemple suivant propose deux tables qui doivent être converties de MyISAM en InnoDB. Le résultat inclut aussi la taille approximative de chaque table en mégaoctets (Mo).

```
+---------------------------------+------------------+
```

```
| ==> MyISAM or Compressed Tables | Approx size (MB) |
+---------------------------------+------------------+
| test.name_table                 |          2102.25 |
| test.my_table                   |            65.25 |
+---------------------------------+------------------+
2 rows in set (0.01 sec)
```

Migration d'un instantané de base de données RDS for MySQL vers un cluster de bases de données Aurora MySQL

Vous pouvez migrer un instantané de base de données d'une instance de base de données RDS for MySQL afin de créer un cluster de bases de données Aurora MySQL à l'aide d'AWS Management Console ou d'AWS CLI. Le nouveau cluster de bases de données Aurora MySQL est rempli avec les données de l'instance de base de données RDS for MySQL initiale. Pour plus d'informations sur la création d'un instantané de base de données, consultez Création d'un instantané de base de données.

Si l'instantané de bases de données ne se trouve pas dans la région AWS où vous voulez que vos données résident, copiez-le dans cette région AWS. Pour plus d'informations sur la copie d'un instantané de base de données, consultez Copie d'un instantané.

Console

Lorsque vous migrez l'instantané de base de données à l'aide de la AWS Management Console, celle-ci effectue les actions nécessaires pour créer uniquement le cluster de bases de données.

Vous pouvez aussi choisir que votre nouveau cluster de bases de données Aurora MySQL soit chiffré au repos à l'aide d'une AWS KMS key.

Pour migrer un instantané de base de données MySQL à l'aide de la AWS Management Console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Commencez la migration à partir de l'instance de base de données MySQL ou de l'instantané :

   Pour lancer la migration à partir de l'instance de base de données :

   1. Dans le panneau de navigation, choisissez Bases de données, puis sélectionnez l'instance de base de données MySQL.
   2. Pour Actions, choisissez Migrer l'instantané le plus récent.

Pour lancer la migration à partir de l'instantané :

1. Choisissez Instantanés.
2. Sur la page Instantanés, choisissez l'instantané que vous voulez migrer vers un cluster de bases de données Aurora MySQL.
3. Choisissez Actions d'instantané, puis Migrer l'instantané.

La page Migrer la base de données apparaît.

3. Définissez les valeurs suivantes dans la page Migrer la base de données :

   - Migrate to DB Engine : sélectionnez `aurora`.
   - Version du moteur de base de données : sélectionnez la version du moteur de base de données pour le cluster de bases de données Aurora MySQL.
   - Classe d'instance de base de données : sélectionnez une classe d'instance de base de données qui possède le stockage et la capacité requis pour votre base de données. Par exemple, `db.r3.large`. Les volumes de cluster Aurora croissent automatiquement au fur et à mesure que la quantité de données de votre base de données augmente. Un volume de cluster Aurora peut croître jusqu'à la taille maximale de 128 tébioctets (Tio). Par conséquent, vous devez uniquement sélectionner une classe d'instance de base de données qui correspond à vos besoins de stockage actifs. Pour plus d'informations, consultez Présentation du stockage Amazon Aurora.
   - Identifiant d'instance de base de données : saisissez un nom de cluster de bases de données, unique pour votre compte dans la région AWS sélectionnée. Cet identifiant est utilisé dans les adresses de point de terminaison des instances de votre cluster DB. Vous pouvez choisir de complexifier le nom, par exemple en incluant la région AWS et le moteur de base de données que vous avez sélectionnés : par exemple, **`aurora-cluster1`**.

     L'identifiant d'instance de base de données obéit aux contraintes suivantes :

     - Il doit contenir entre 1 et 63 caractères alphanumériques ou traits d'union.
     - Son premier caractère doit être une lettre.
     - Il ne peut pas se terminer par un trait d'union ou contenir deux traits d'union consécutifs.
     - Doit être unique pour toutes les instances de base de données par compte AWS et par région AWS.

- Virtual Private Cloud (VPC) : si vous disposez d'un VPC existant, vous pouvez l'utiliser avec votre cluster de bases de données Aurora MySQL en sélectionnant l'identifiant de votre VPC, par exemple `vpc-a464d1c1`. Pour plus d'informations sur la création d'un VPC, consultez Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement).

  Sinon, vous pouvez choisir de demander à Aurora de créer un VPC pour vous en sélectionnant Créer un VPC.

- Groupe de sous-réseaux de base de données : si vous disposez d'un groupe de sous-réseaux existant, vous pouvez l'utiliser avec votre cluster de bases de données Aurora MySQL en sélectionnant l'identifiant de votre groupe de sous-réseaux, par exemple `gs-subnet-group1`.

  Sinon, vous pouvez choisir de demander à Aurora de créer un groupe de sous-réseaux pour vous en sélectionnant Create a new subnet group (Créer un groupe de sous-réseaux).

- Accessible publiquement : sélectionnez Non pour spécifier que les instances de votre cluster de bases de données ne sont accessibles que par les ressources situées à l'intérieur de votre VPC. Sélectionnez Oui pour spécifier que les instances de votre cluster de bases de données sont accessibles par les ressources du réseau public. La valeur par défaut est Oui.

  > **Note**
  >
  > Il n'est pas nécessaire que votre cluster de bases de données de production se trouve dans un sous-réseau public, parce que seuls vos serveurs d'applications nécessitent l'accès à votre cluster de bases de données. Si votre cluster de bases de données n'a pas besoin d'être dans un sous-réseau public, définissez Accessible publiquement sur Non.

- Zone de disponibilité : sélectionnez la zone de disponibilité devant héberger l'instance principale de votre cluster de bases de données Aurora MySQL. Pour laisser Aurora choisir une zone de disponibilité à votre place, sélectionnez Aucune préférence.

- Port de la base de données : saisissez le port par défaut à utiliser lors de la connexion aux instances du cluster de bases de données Aurora MySQL. La valeur par défaut est `3306`.

  > **Note**
  >
  > Il se peut que vous soyez derrière un pare-feu d'entreprise qui n'autorise pas l'accès aux ports par défaut, tels que le port par défaut MySQL 3306. Dans ce cas, fournissez

une valeur de port que votre pare-feu d'entreprise autorise. Souvenez-vous plus tard de cette valeur de port lorsque vous vous connecterez au cluster de bases de données Aurora MySQL.

- Chiffrement : choisissez Activer le chiffrement pour que votre nouveau cluster de bases de données Aurora MySQL soit chiffré au repos. Si vous choisissez Activer le chiffrement, vous devez choisir une clé KMS comme valeur de AWS KMS key.

  Si votre instantané de base de données n'est pas chiffré, spécifiez une clé de chiffrement de manière à ce que votre cluster de bases de données soit chiffré au repos.

  Si votre instantané de base de données est chiffré, spécifiez une clé de chiffrement de manière à ce que votre cluster de bases de données soit chiffré au repos avec la clé de chiffrement spécifiée. Vous pouvez spécifier la clé de chiffrement utilisée par l'instantané de bases de données ou une clé différente. Vous ne pouvez pas créer de cluster de bases de données non chiffré à partir d'un instantané de base de données chiffré.
- Mise à niveau automatique des versions mineures : ce paramètre ne s'applique pas aux clusters de bases de données Aurora MySQL.

  Pour plus d'informations sur les mises à jour de moteur pour Aurora MySQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL.

4. Choisissez Migrer pour migrer votre instantané de base de données.
5. Sélectionnez Instances, puis choisissez l'icône en forme de flèche pour afficher les détails du cluster DB et surveiller la progression de la migration. Sur la page des détails, vous pouvez trouver le point de terminaison du cluster utilisé pour se connecter à l'instance principale du cluster de bases de données. Pour plus d'informations sur la connexion à un cluster de bases de données Aurora MySQL, consultez Connexion à un cluster de bases de données Amazon Aurora.

AWS CLI

Vous pouvez créer un cluster de bases de données Aurora à partir d'un instantané de bases de données d'une instance de base de données RDS for MySQL par l'intermédiaire de la commande `restore-db-cluster-from-snapshot` avec les paramètres suivants :

- `--db-cluster-identifier` : nom du cluster de bases de données à créer.

- `--engine aurora-mysql` – Pour un cluster de bases de données compatible avec MySQL 5.7 ou 8.0.
- `--kms-key-id` – La AWS KMS key avec laquelle chiffrer éventuellement le cluster de bases de données, selon que votre instantané de base de données est chiffré ou non.
  - Si votre instantané de base de données n'est pas chiffré, spécifiez une clé de chiffrement de manière à ce que votre cluster de bases de données soit chiffré au repos. Sinon, votre cluster de bases de données ne sera pas chiffré.
  - Si votre instantané de base de données est chiffré, spécifiez une clé de chiffrement de manière à ce que votre cluster de bases de données soit chiffré au repos avec la clé de chiffrement spécifiée. Sinon, votre cluster de bases de données est chiffré au repos avec la clé de chiffrement de l'instantané de base de données.

    > **Note**
    >
    > Vous ne pouvez pas créer de cluster de bases de données non chiffré à partir d'un instantané de base de données chiffré.

- `--snapshot-identifier` : l'Amazon Resource Name (ARN) de l'instantané de bases de données à migrer. Pour plus d'informations sur les ARN Amazon RDS, consultez Amazon Relational Database Service (Amazon RDS).

Lorsque vous migrez l'instantané de bases de données à l'aide de la commande `RestoreDBClusterFromSnapshot`, celle-ci crée le cluster de bases de données et l'instance principale.

Dans cet exemple, vous créez un cluster de bases de données compatible avec MySQL 5.7 nommé *`mydbcluster`* à partir d'un instantané de base de données avec un ARN défini sur *`mydbsnapshotARN`*.

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora-mysql
```

Pour Windows :

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora-mysql
```

Dans cet exemple, vous créez un cluster de bases de données compatible avec MySQL 5.7 nommé *mydbcluster* à partir d'un instantané de base de données avec un ARN défini sur *mydbsnapshotARN*.

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mydbcluster \
    --snapshot-identifier mydbsnapshotARN \
    --engine aurora-mysql
```

Pour Windows :

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier mydbcluster ^
    --snapshot-identifier mydbsnapshotARN ^
    --engine aurora-mysql
```

Migration des données d’une instance de base de données RDS for MySQL vers un cluster de bases de données Amazon Aurora MySQL à l’aide d’un réplica en lecture Aurora

Aurora utilise la fonctionnalité de réplication de journaux binaires des moteurs de base de données MySQL pour créer un type spécial de cluster de bases de données appelé un réplica en lecture Aurora pour une instance de base de données RDS for MySQL source. Les mises à jour apportées à l’instance de base de données RDS for MySQL source sont répliquées de façon asynchrone sur le réplica en lecture Aurora.

Nous vous recommandons d’utiliser cette fonctionnalité pour effectuer une migration d’une instance de base de données RDS for MySQL vers un cluster de bases de données Aurora MySQL en créant un réplica en lecture Aurora de votre instance de base de données RDS for MySQL source. Lorsque le retard du réplica entre l’instance de base de données RDS for MySQL et le réplica en lecture Aurora est égal à 0, vous pouvez diriger vos applications clientes vers le réplica en lecture Aurora, puis arrêter la réplication pour transformer le réplica en lecture Aurora en cluster de bases de données Aurora MySQL autonome. La migration peut prendre du temps, environ quelques heures par téraoctets (Tio) de données.

Pour obtenir la liste des régions où Aurora est disponible, consultez Amazon Aurora dans la Références générales AWS.

Lorsque vous créez un réplica en lecture Aurora d’une instance de base de données RDS for MySQL, Amazon RDS crée un instantané de base de données de votre instance de base de données RDS for MySQL source (privé pour Amazon RDS et sans frais). Amazon RDS migre ensuite les données de l’instantané de base de données vers le réplica en lecture Aurora. Une fois que les données de l’instantané de base de données ont été migrées vers le nouveau cluster de bases de données Aurora MySQL, Amazon RDS démarre la réplication entre votre instance de base de données RDS for MySQL et le cluster de bases de données Aurora MySQL. Si votre instance de base de données RDS for MySQL contient des tables qui utilisent des moteurs de stockage autres qu’InnoDB ou qui utilisent un format de ligne compressé, vous pouvez accélérer le processus de création d’un réplica en lecture Aurora. Pour cela, il faut modifier ces tables pour utiliser le moteur de stockage InnoDB et le format de ligne dynamique avant de créer votre réplica en lecture Aurora. Pour plus d’informations sur le processus de copie d’un instantané de base de données MySQL vers un cluster de bases de données Aurora MySQL, consultez Migration de données à partir d'une instance de base de données RDS MySQL vers un cluster de base de données Amazon Aurora MySQL.

Vous ne pouvez avoir qu'un seul réplica en lecture Aurora pour une instance de base de données RDS for MySQL.

> **Note**
>
> Des problèmes de réplication peuvent survenir en raison de différences de fonctionnalités entre Aurora MySQL et la version du moteur de base de données MySQL de votre instance de base de données RDS for MySQL qui est l'instance principale de réplication. Si vous rencontrez une erreur, vous pouvez obtenir de l'aide dans le Forum de la communauté Amazon RDS ou en contactant l'AWS Support.
> Vous ne pouvez pas créer de réplica en lecture Aurora si votre instance de base de données RDS for MySQL est déjà la source d'un réplica en lecture entre régions.
> Vous ne pouvez pas migrer de certaines anciennes versions de RDS for MySQL 8.0, notamment des versions 8.0.11, 8.0.13 et 8.0.15, vers Aurora MySQL 3.05 et versions ultérieures. Nous vous recommandons de passer à RDS for MySQL version 8.0.28 avant de procéder à la migration.

Pour plus d'informations sur les réplicas en lecture MySQL, consultez Utilisation des réplicas en lecture des instances de base de données MariaDB, MySQL et PostgreSQL.

Création d'un réplica en lecture Aurora

Vous pouvez créer un réplica en lecture Aurora pour une instance de base de données RDS for MySQL à l'aide de la console, de l'interface AWS CLI ou de l'API RDS.

Console

Pour créer un réplica en lecture Aurora à partir d'une instance de base de données RDS for MySQL source

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez l'instance de base de données MySQL que vous souhaitez utiliser comme source pour votre réplica en lecture Aurora.
4. Sous Actions, choisissez Créer un réplica en lecture Aurora.

5. Choisissez les spécifications de cluster de bases de données que vous voulez utiliser pour le réplica en lecture Aurora, comme décrit dans le tableau suivant.

Option	Description
Classe d’instance de base de données	Choisissez une classe d’instance de base de données qui définit les exigences de mémoire et de traitement pour l’instance principale du cluster de bases de données. Pour plus d’informations sur les options de classe d’instance de base de données, consultez Classes d’instance de base de données Amazon Aurora.
déploiement multi-AZ	Choisissez Create Replica in Different Zone (Créer un réplica dans une autre zone) pour créer un réplica autonome du nouveau cluster de bases de données dans une autre Zone de disponibilité dans la région AWS cible à des fins de prise en charge du basculement. Pour plus d’informations sur les zones de disponibilité multiples, consultez Régions et zones de disponibilité.

Option	Description
Identifiant d’instance de base de données	Saisissez le nom de l’instance principale de votre cluster de bases de données de réplica en lecture Aurora. Cet identifiant est utilisé dans l’adresse de point de terminaison de l’instance principale du nouveau cluster de bases de données. L’identifiant d’instance de base de données obéit aux contraintes suivantes : • Il doit contenir entre 1 et 63 caractères alphanumériques ou traits d’union. • Son premier caractère doit être une lettre. • Il ne peut pas se terminer par un trait d’union ou contenir deux traits d’union consécutifs. • Il doit être unique pour toutes les instances de bases de données de chaque compte AWS, pour chaque région AWS. Étant donné que le cluster de bases de données de réplica en lecture Aurora est créé à partir d’un instantané de l’instance de base de données source, le nom et le mot de passe d’utilisateur principal du réplica en lecture Aurora sont les mêmes que le nom et le mot de passe d’utilisateur principal de l’instance de base de données source.
Cloud privé virtuel (VPC)	Sélectionnez le VPC pour héberger le cluster de bases de données. Sélectionnez Créer un nouveau VPC pour qu’Aurora crée un VPC pour vous. Pour plus d’informations, consultez Prérequis des clusters de bases de données.

Option	Description
Groupe de sous-réseaux de base de données	Sélectionnez le groupe de sous-réseaux DB à utiliser pour le cluster de bases de données. Sélectionnez Créer un groupe de sous-réseaux DB pour qu'Aurora crée un groupe de sous-réseaux de base de données pour vous. Pour plus d'informations, consultez Prérequis des clusters de bases de données.
Accessible publiquement	Sélectionnez `Yes` pour attribuer au cluster de bases de données une adresse IP publique ; sinon, sélectionnez `No`. Les instances de votre cluster de bases de données peuvent être un mélange d'instances de base de données publiques et privées. Pour plus d'informations sur le masquage des instances de l'accès public, consultez Masquer un cluster de bases de données dans un VPC depuis Internet.
Zone de disponibilité	Déterminez si vous voulez spécifier une zone de disponibilité particulière. Pour plus d'informations sur les zones de disponibilité, consultez Régions et zones de disponibilité.
Groupe de sécurité VPC (pare-feu)	Sélectionnez Create new VPC security group (Créer un groupe de sécurité VPC) pour qu'Aurora crée un groupe de sécurité VPC pour vous. Choisissez Select existing VPC security groups (Sélectionner des groupes de sécurité VPC existants) afin de spécifier un ou plusieurs groupes de sécurité VPC pour sécuriser l'accès réseau au cluster de bases de données. Pour plus d'informations, consultez Prérequis des clusters de bases de données.

Option	Description
Port de la base de données	Spécifiez le port utilisé par les applications et les utilitaires pour accéder à la base de données. Les clusters de bases de données Aurora MySQL utilisent par défaut le port MySQL 3306. Les pare-feu de certaines entreprises bloquent les connexions vers ce port. Si le pare-feu de votre entreprise bloque le port par défaut, choisissez un autre port pour le nouveau cluster DB.
Groupe de paramètres de base de données	Sélectionnez un groupe de paramètres de base de données pour le cluster de bases de données Aurora MySQL. Aurora possède un groupe de paramètres de base de données par défaut que vous pouvez utiliser, ou bien vous pouvez créer votre propre groupe de paramètres de base de données. Pour plus d'informations sur les groupes de paramètres DB, consultez Groupes de paramètres pour Amazon Aurora.
Groupe de paramètres de cluster de bases de données	Sélectionnez un groupe de paramètres de cluster de bases de données pour le cluster de bases de données Aurora MySQL. Aurora possède un groupe de paramètres de cluster de bases de données par défaut que vous pouvez utiliser, ou bien vous pouvez créer votre propre groupe de paramètres de cluster de bases de données. Pour plus d'informations sur les groupes de paramètres de cluster DB, consultez Groupes de paramètres pour Amazon Aurora.

Option	Description
Chiffrement	Choisissez Disable encryption (Désactiver le chiffrement) si vous ne voulez pas que votre nouveau cluster de bases de données Aurora soit chiffré. Choisissez Activer le chiffrement pour que votre nouveau cluster de bases de données Aurora soit chiffré au repos. Si vous choisissez Activer le chiffrement, vous devez choisir une clé KMS comme valeur de AWS KMS key. Si votre instance de base de données MySQL n'est pas chiffrée, spécifiez une clé de chiffrement de manière à ce que votre cluster de bases de données soit chiffré au repos. Si votre instance de base de données MySQL est chiffrée, spécifiez une clé de chiffrement de manière à ce que votre cluster de bases de données soit chiffré au repos avec la clé de chiffrement spécifiée. Vous pouvez spécifier la clé de chiffrement utilisée par l'instance de base de données MySQL ou une clé différente. Vous ne pouvez pas créer de cluster de bases de données non chiffré à partir d'une instance de base de données MySQL chiffrée.
Priorité	Choisissez une priorité de basculement pour le cluster DB. Si vous ne sélectionnez pas de valeur, la valeur par défaut est tier-1. Cette priorité détermine l'ordre dans lequel les réplicas Aurora sont promus lors de la reprise après une défaillance de l'instance principale. Pour plus d'informations, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.

Option	Description
Période de rétention des sauvegardes	Sélectionnez la durée, comprise entre 1 et 35 jours, pendant laquelle Aurora conserve les copies de sauvegarde de la base de données. Les copies de sauvegarde peuvent être utilisées pour les restaurations à un instant dans le passé de votre base de données à la seconde.
Surveillance améliorée	Choisissez Activer la surveillance améliorée pour activer la collecte de métriques en temps réel pour le système d'exploitation sur lequel votre cluster de bases de données s'exécute. Pour plus d'informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.
Monitoring Role (Rôle de surveillance)	Disponible uniquement si l'option Surveillance améliorée est définie sur Activer la surveillance améliorée. Choisissez le rôle IAM que vous avez créé pour autoriser Aurora à communiquer avec Amazon CloudWatch Logs en votre nom ou choisissez Par défaut pour qu'Aurora crée automatiquement un rôle nommé `rds-monitoring-role` . Pour plus d'informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.
Granularité	Disponible uniquement si l'option Surveillance améliorée est définie sur Activer la surveillance améliorée. Définissez l'intervalle, en secondes, entre les recueils des métriques pour votre cluster DB.
Mise à niveau automatique de versions mineures	Ce paramètre ne s'applique pas aux clusters de bases de données Aurora MySQL. Pour plus d'informations sur les mises à jour de moteur pour Aurora MySQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL.

Option	Description
Fenêtre de maintenance	Choisissez Sélectionner la fenêtre et spécifiez la plage de temps hebdomadaire au cours de laquelle la maintenance peut avoir lieu. Vous pouvez également sélectionner Aucune préférence afin qu'Aurora affecte une période de manière aléatoire.

6. Choisissez Créer un réplica en lecture.

AWS CLI

Pour créer un réplica en lecture Aurora à partir d'une instance de base de données RDS for MySQL source, utilisez les commandes AWS CLI `create-db-cluster` et `create-db-instance` pour créer un nouveau cluster de bases de données Aurora MySQL. Quand vous appelez la commande `create-db-cluster`, incluez le paramètre `--replication-source-identifier` pour identifier l'Amazon Resource Name (ARN) de l'instance de base de données MySQL source. Pour plus d'informations sur les ARN Amazon RDS, consultez Amazon Relational Database Service (Amazon RDS).

Ne spécifiez pas le nom d'utilisateur principal, le mot de passe principal ni le nom de base de données, car le réplica en lecture Aurora utilise les mêmes nom d'utilisateur principal, mot de passe principal et nom de base de données que l'instance de base de données MySQL source.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine
 aurora \
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 \
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:primary-
mysql-instance
```

Pour Windows :

```
aws rds create-db-cluster --db-cluster-identifier sample-replica-cluster --engine
 aurora ^
    --db-subnet-group-name mysubnetgroup --vpc-security-group-ids sg-c7e5b0d2 ^
    --replication-source-identifier arn:aws:rds:us-west-2:123456789012:db:primary-
mysql-instance
```

Si vous utilisez la console pour créer un réplica en lecture Aurora, Aurora crée automatiquement l'instance principale de votre réplica en lecture Aurora de cluster de bases de données. Si vous utilisez l'AWS CLI pour créer un réplica en lecture Aurora, vous devez créer explicitement l'instance principale de votre cluster de bases de données. L'instance principale est la première instance créée dans un cluster de bases de données.

Vous pouvez créer une instance principale pour votre cluster de bases de données en utilisant la commande `create-db-instance` de l'AWS CLI avec les paramètres suivants.

- `--db-cluster-identifier`

  Nom du cluster de bases de données.
- `--db-instance-class`

  Nom de la classe d'instance de base de données à utiliser pour votre instance principale.
- `--db-instance-identifier`

  Nom de votre instance principale.
- `--engine aurora`

Dans cet exemple, vous créez une instance principale nommée *myreadreplicainstance* pour le cluster de bases de données nommé *myreadreplicacluster*, en utilisant la classe d'instance de base de données spécifiée dans *myinstanceclass*.

Example

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance \
    --db-cluster-identifier myreadreplicacluster \
    --db-instance-class myinstanceclass \
    --db-instance-identifier myreadreplicainstance \
    --engine aurora
```

Pour Windows :

```
aws rds create-db-instance ^
    --db-cluster-identifier myreadreplicacluster ^
    --db-instance-class myinstanceclass ^
```

```
    --db-instance-identifier myreadreplicainstance ^
    --engine aurora
```

API RDS

Pour créer un réplica en lecture Aurora à partir d'une instance de base de données RDS for MySQL source, utilisez les commandes d'API Amazon RDS `CreateDBCluster` et `CreateDBInstance` afin de créer un nouveau cluster de bases de données Aurora et une instance principale. N'indiquez pas le nom d'utilisateur principal, le mot de passe principal ni le nom de base de données, car le réplica en lecture Aurora utilise les mêmes nom d'utilisateur principal, mot de passe principal et nom de base de données que l'instance de base de données RDS for MySQL source.

Vous pouvez créer un nouveau cluster de bases de données Aurora pour un réplica en lecture Aurora à partir d'une instance de base de données RDS for MySQL source à l'aide de la commande d'API Amazon RDS `CreateDBCluster` avec les paramètres suivants :

- `DBClusterIdentifier`

  Nom du cluster de bases de données à créer.

- `DBSubnetGroupName`

  Nom du groupe de sous-réseaux de base de données à associer à ce cluster de bases de données.

- `Engine=aurora`
- `KmsKeyId`

  AWS KMS key avec laquelle chiffrer éventuellement le cluster de bases de données, selon que votre instance de base de données MySQL est chiffrée ou non.

  - Si votre instance de base de données MySQL n'est pas chiffrée, spécifiez une clé de chiffrement de manière à ce que votre cluster de bases de données soit chiffré au repos. Sinon, votre cluster de bases de données est chiffré au repos avec la clé de chiffrement par défaut de votre compte.
  - Si votre instance de base de données MySQL est chiffrée, spécifiez une clé de chiffrement de manière à ce que votre cluster de bases de données soit chiffré au repos avec la clé de chiffrement spécifiée. Sinon, votre cluster de bases de données est chiffré au repos avec la clé de chiffrement de l'instance de base de données MySQL.

> **Note**
>
> Vous ne pouvez pas créer de cluster de bases de données non chiffré à partir d'une instance de base de données MySQL chiffrée.

- `ReplicationSourceIdentifier`

  Amazon Resource Name (ARN) de l'instance de base de données MySQL source. Pour plus d'informations sur les ARN Amazon RDS, consultez Amazon Relational Database Service (Amazon RDS).

- `VpcSecurityGroupIds`

  Liste des groupes de sécurité VPC EC2 à associer à ce cluster de bases de données.

Dans cet exemple, vous créez un cluster de bases de données nommé *myreadreplicacluster* à partir d'une instance de base de données MySQL source avec un ARN défini sur *mysqlprimaryARN*, associé à un groupe de sous-réseaux de base de données nommé *mysubnetgroup* et à un groupe de sécurité VPC nommé *mysecuritygroup*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBCluster
    &DBClusterIdentifier=myreadreplicacluster
    &DBSubnetGroupName=mysubnetgroup
    &Engine=aurora
    &ReplicationSourceIdentifier=mysqlprimaryARN
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-10-31
    &VpcSecurityGroupIds=mysecuritygroup
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20150927/us-east-1/rds/aws4_request
    &X-Amz-Date=20150927T164851Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=6a8f4bd6a98f649c75ea04a6b3929ecc75ac09739588391cd7250f5280e716db
```

Si vous utilisez la console pour créer un réplica en lecture Aurora, Aurora crée automatiquement l'instance principale de votre réplica en lecture Aurora de cluster de bases de données. Si vous

utilisez l'AWS CLI pour créer un réplica en lecture Aurora, vous devez créer explicitement l'instance principale de votre cluster de bases de données. L'instance principale est la première instance créée dans un cluster de bases de données.

Vous pouvez créer une instance principale pour votre cluster de bases de données en utilisant la commande d'API Amazon RDS `CreateDBInstance` avec les paramètres suivants :

- `DBClusterIdentifier`

  Nom du cluster de bases de données.

- `DBInstanceClass`

  Nom de la classe d'instance de base de données à utiliser pour votre instance principale.

- `DBInstanceIdentifier`

  Nom de votre instance principale.

- `Engine=aurora`

Dans cet exemple, vous créez une instance principale nommée *`myreadreplicainstance`* pour le cluster de bases de données nommé *`myreadreplicacluster`*, en utilisant la classe d'instance de base de données spécifiée dans *`myinstanceclass`*.

Example

```
https://rds.us-east-1.amazonaws.com/
    ?Action=CreateDBInstance
    &DBClusterIdentifier=myreadreplicacluster
    &DBInstanceClass=myinstanceclass
    &DBInstanceIdentifier=myreadreplicainstance
    &Engine=aurora
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-09-01
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20140424/us-east-1/rds/aws4_request
    &X-Amz-Date=20140424T194844Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=bee4aabc750bf7dad0cd9e22b952bd6089d91e2a16592c2293e532eeaab8bc77
```

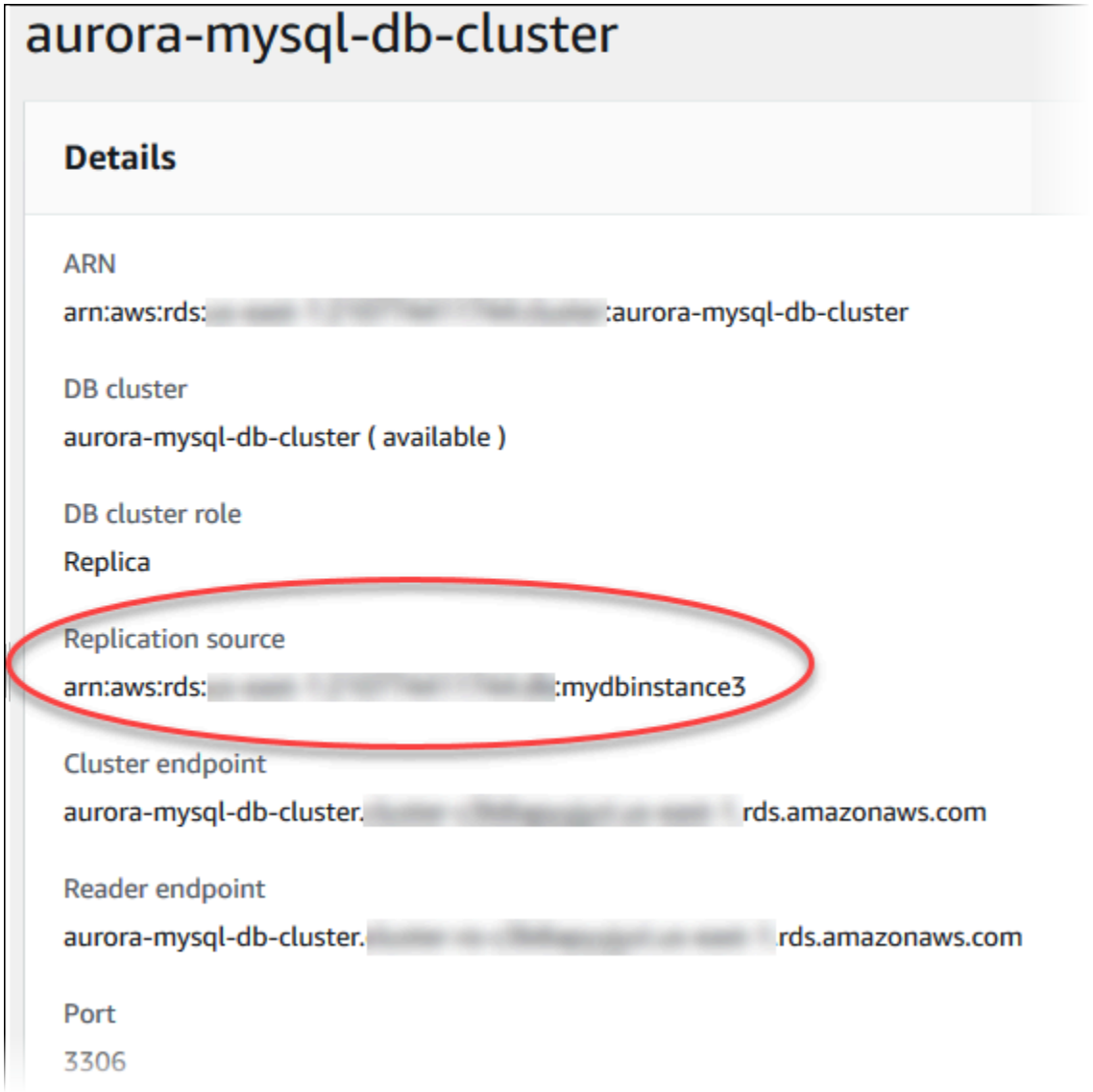
Affichage d’un réplica en lecture Aurora

Vous pouvez afficher les relations de réplication MySQL vers Aurora MySQL de vos clusters de bases de données Aurora MySQL à l’aide de l’AWS Management Console ou de l’AWS CLI.

Console

Pour afficher l’instance de base de données MySQL principale pour un réplica en lecture Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Sélectionnez le cluster de bases de données pour le réplica en lecture Aurora afin d’afficher ses détails. Les informations de l’instance de base de données MySQL principale figurent dans le champ Replication source (Source de réplication).



AWS CLI

Pour afficher les relations de réplication MySQL vers Aurora MySQL de vos clusters de bases de données Aurora MySQL à l'aide de l'AWS CLI, utilisez les commandes `describe-db-clusters` et `describe-db-instances`.

Pour déterminer quelles est l'instance de base de données MySQL principale, utilisez la commande `describe-db-clusters` et indiquez l'identifiant de cluster du réplica en lecture Aurora pour l'option `--db-cluster-identifier`. Reportez-vous à l'élément `ReplicationSourceIdentifier` dans le résultat de l'ARN de l'instance de base de données qui est le principal de réplication.

Pour déterminer quel cluster de bases de données est le réplica en lecture Aurora, utilisez la commande `describe-db-instances` et indiquez l'identifiant d'instance de l'instance de base de données MySQL pour l'option `--db-instance-identifier`. Reportez-vous à l'élément `ReadReplicaDBClusterIdentifiers` dans la sortie de l'identifiant de cluster de bases de données du réplica en lecture Aurora.

Example

Pour Linux, macOS ou Unix :

```
aws rds describe-db-clusters \
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances \
    --db-instance-identifier mysqlprimary
```

Pour Windows :

```
aws rds describe-db-clusters ^
    --db-cluster-identifier myreadreplicacluster
```

```
aws rds describe-db-instances ^
    --db-instance-identifier mysqlprimary
```

Promotion d'un réplica en lecture Aurora

Une fois que la migration est terminée, vous pouvez effectuer la promotion du réplica en lecture Aurora en cluster de bases de données autonome à l'aide d'AWS Management Console ou d'AWS CLI.

Vous pouvez ensuite diriger vos applications clientes vers le point de terminaison du réplica en lecture Aurora. Pour plus d'informations sur les points de terminaison Aurora, consultez Connexions de point de terminaison Amazon Aurora. La promotion doit s'achever rapidement, et vous pouvez lire le réplica en lecture Aurora ou écrire dans ce réplica lors de la promotion. Toutefois, vous ne pouvez pas supprimer l'instance de base de données MySQL principale, ni supprimer le lien entre l'instance de base de données et le réplica en lecture Aurora à ce moment-là.

Avant de promouvoir le réplica en lecture Aurora, arrêtez toute écriture sur l'instance de base de données MySQL source, puis attendez que le retard du réplica en lecture Aurora soit égal à 0. Vous

pouvez afficher le retard de réplica pour un réplica en lecture Aurora en appelant la commande `SHOW SLAVE STATUS` (Aurora MySQL version 2) ou`SHOW REPLICA STATUS` (Aurora MySQL version 3) sur votre réplica en lecture Aurora. Vérifiez la valeur Seconds behind master (Secondes de retard sur l'instance principale).

Vous pouvez commencer à écrire dans le réplica en lecture Aurora une fois que les transactions d'écriture sur le réplica principal ont été interrompues et que le retard du réplica est égal à 0. Si vous écrivez dans le réplica en lecture Aurora en amont et que vous modifiez les tables qui sont également modifiées sur le principal MySQL, vous risquez d'interrompre la réplication sur Aurora. Si cela se produit, vous devez supprimer et recréer votre cluster en lecture Aurora.

Console

Pour promouvoir un réplica en lecture Aurora en cluster de bases de données Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Sélectionnez le cluster de bases de données pour le réplica en lecture Aurora.
4. Pour Actions, choisissez Promote (Promouvoir).
5. Sélectionnez Promote read replica (Promouvoir le réplica en lecture).

Ensuite, vérifiez que la promotion est terminée à l'aide de la procédure suivante.

Pour confirmer que le réplica en lecture Aurora a été promu

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, sélectionnez Events.
3. Sur la page Events (Événements), vérifiez qu'il existe un événement `Promoted Read Replica cluster to a stand-alone database cluster` pour le cluster que vous avez promu.

Une fois que la promotion est terminée, l'instance de base de données MySQL principale et le réplica en lecture Aurora sont détachés, et vous pouvez supprimer en toute sécurité l'instance de base de données si vous le souhaitez.

AWS CLI

Pour promouvoir un réplica en lecture Aurora en cluster de bases de données autonome, utilisez la commande de l'AWS CLI `promote-read-replica-db-cluster`.

Example

Pour Linux, macOS ou Unix :

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

Pour Windows :

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

Gestion d'Amazon Aurora MySQL

Les sections suivantes expliquent comment gérer un cluster de base de données Amazon Aurora MySQL DB.

Rubriques

Gestion des performances et dimensionnement pour Amazon Aurora MySQL

Dimensionnement des instances de bases de données Aurora MySQL

Vous pouvez dimensionner les instances de bases de données Aurora MySQL de deux façons : le dimensionnement d'instance et le dimensionnement en lecture. Pour plus d'informations sur le dimensionnement en lecture, consultez Dimensionnement en lecture.

Vous pouvez mettre à l'échelle votre cluster de bases de données Aurora MySQL en modifiant la classe d'instance de base de données pour chaque instance du cluster de bases de données. Aurora MySQL prend en charge plusieurs classes d'instance de base de données optimisées pour Aurora. N'utilisez pas les classes d'instance db.t2 ou db.t3 pour des clusters Aurora d'une taille supérieure à 40 To. Pour obtenir les spécifications des classes d'instance de base de données prises en charge par Aurora MySQL, consultez Classes d'instance de base de données Amazon Aurora.

> **Note**
>
> Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à la production. Pour plus de détails sur les classes d'instance T, consultez Utilisation de classes d'instance T pour le développement et les tests.

Nombre maximal de connexions à une instance de base de données Aurora MySQL

Le nombre maximal de connexions autorisées à une instance de base de données Aurora MySQL est déterminé par le paramètre `max_connections` du groupe de paramètres de niveau instance de l'instance de base de données.

Le tableau ci-dessous répertorie la valeur par défaut résultante de `max_connections` pour chaque classe d'instance de base de données disponible dans Aurora MySQL. Vous pouvez accroître le nombre maximal de connexions à votre instance de base de données Aurora MySQL en définissant une classe d'instance de base de données qui offre davantage de mémoire à l'instance ou en attribuant au paramètre `max_connections` une valeur supérieure (jusqu'à 16 000) dans le groupe de paramètres de base de données de votre instance.

> **Tip**
>
> Si vos applications ouvrent et ferment fréquemment des connexions, ou si elles ont ouvert un grand nombre de connexions de longue durée, nous vous recommandons d'utiliser Proxy Amazon RDS. RDS Proxy est un proxy de base de données entièrement géré et hautement disponible qui utilise le regroupement de connexions pour partager les connexions de base de données de manière sécurisée et efficace. Pour en savoir plus sur RDS Proxy, consultez Proxy Amazon RDS pour Aurora.

Pour obtenir plus de détails sur la façon dont les instances Aurora Serverless v2 gèrent ce paramètre, consultez Nombre maximal de connexions pour Aurora Serverless v2.

Classe d'instance	Valeur par défaut de max_connections		
db.t2.small	45		
db.t2.medium	90		
db.t3.small	45		
db.t3.medium	90		

Classe d’instance	Valeur par défaut de max_connections		
db.t3.large	135		
db.t4g.medium	90		
db.t4g.large	135		
db.r3.large	1 000		
db.r3.xlarge	2000		
db.r3.2xlarge	3000		
db.r3.4xlarge	4000		
db.r3.8xlarge	5000		
db.r4.large	1 000		
db.r4.xlarge	2000		
db.r4.2xlarge	3000		
db.r4.4xlarge	4000		
db.r4.8xlarge	5000		
db.r4.16xlarge	6000		
db.r5.large	1000		
db.r5.xlarge	2000		
db.r5.2xlarge	3000		
db.r5.4xlarge	4000		

Classe d’instance	Valeur par défaut de max_connections		
db.r5.8xlarge	5000		
db.r5.12xlarge	6 000		
db.r5.16xlarge	6 000		
db.r5.24xlarge	7000		
db.r6g.large	1000		
db.r6g.xlarge	2000		
db.r6g.2xlarge	3000		
db.r6g.4xlarge	4000		
db.r6g.8xlarge	5000		
db.r6g.12xlarge	6 000		
db.r6g.16xlarge	6000		
db.r6i.large	1 000		
db.r6i.xlarge	2000		
db.r6i.2xlarge	3000		
db.r6i.4xlarge	4000		
db.r6i.8xlarge	5000		
db.r6i.12xlarge	6 000		
db.r6i.16xlarge	6 000		

Classe d’instance	Valeur par défaut de max_connections		
db.r6i.24xlarge	7000		
db.r6i.32xlarge	7000		
db.r7g.large	1 000		
db.r7g.xlarge	2000		
db.r7g.2xlarge	3000		
db.r7g.4xlarge	4000		
db.r7g.8xlarge	5000		
db.r7g.12xlarge	6 000		
db.r7g.16xlarge	6 000		
db.r7i.large	1 000		
db.r7i.xlarge	2000		
db.r7i.2xlarge	3000		
db.r7i.4xlarge	4000		
db.r7i.8xlarge	5000		
db.r7i.12xlarge	6 000		
db.r7i.16xlarge	6 000		
db.r7i.24xlarge	7000		
db.r7i.48xlarge	8000		

Classe d’instance	Valeur par défaut de max_connections		
db.r8g.large	1 000		
db.r8g.xlarge	2000		
db.r8g.2xlarge	3000		
db.r8g.4xlarge	4000		
db.r8g.8xlarge	5000		
db.r8g.12xlarge	6 000		
db.r8g.16xlarge	6 000		
db.r8g.24xlarge	7000		
db.r8g.48xlarge	8000		
db.x2g.large	2000		
db.x2g.xlarge	3000		
db.x2g.2xlarge	4000		
db.x2g.4xlarge	5000		
db.x2g.8xlarge	6 000		
db.x2g.12xlarge	7000		
db.x2g.16xlarge	7000		

> **Tip**
>
> Le calcul du paramètre `max_connections` utilise le log base 2 (qui est différent du logarithme naturel) et la valeur `DBInstanceClassMemory` en octets pour la classe d'instance Aurora MySQL sélectionnée. Ce paramètre accepte uniquement les valeurs entières. Les parties décimales sont tronquées lors des calculs. Cette formule implémente des limites de connexion comme suit :
>
> - Incrément de 1 000 connexions pour les instances R3, R4 et R5
> - Incrément de 45 connexion pour les variantes de mémoire d'instance T2 et T3
>
> Exemple : pour db.r6g.large, alors que la formule calcule 1 069,2, le système implémente 1 000 pour maintenir des modèles incrémentiels cohérents.

Si vous créez un groupe de paramètres dans le but de personnaliser votre limite de connexions par défaut, vous constaterez que cette limite est dérivée d'une formule basée sur la valeur de `DBInstanceClassMemory`. Comme le montre le tableau précédent, la formule produit des limites de connexion qui augmentent de 1 000 à mesure que la mémoire double à chaque nouvel échelon pour les instances R3, R4 et R5, et de 45 pour les différentes tailles de mémoire des instances T2 et T3.

Consultez Spécification des paramètres de base de données pour obtenir plus de détails sur le mode de calcul de `DBInstanceClassMemory`.

Aurora MySQL et les instances de bases de données RDS pour MySQL ont des niveaux de surcharge mémoire différents. Par conséquent, la valeur `max_connections` peut être différente pour Aurora MySQL et les instances de bases de données RDS for MySQL qui utilisent la même classe d'instance. Les valeurs de la table s'appliquent uniquement aux instances de base de données Aurora MySQL.

> **Note**
>
> Les limites de connectivité bien inférieures des instances T2 et T3 s'expliquent par le fait qu'avec Aurora, ces classes d'instance sont destinées uniquement à des scénarios de développement et de test, et non à des charges de travail de production.

Les limites de connexion par défaut sont adaptées aux systèmes qui utilisent les valeurs par défaut des autres gros consommateurs de mémoire, comme les pools de mémoires tampons et les caches de requêtes. Si vous modifiez ces autres paramètres pour votre cluster, pensez à ajuster la limite de connexion pour prendre en compte l'augmentation ou la diminution de la mémoire disponible sur les instances de base de données.

Limites de stockage temporaires pour Aurora MySQL

Aurora MySQL stocke les tables et les index dans le sous-système de stockage Aurora. Aurora MySQL utilise un stockage temporaire ou local séparé pour les tables temporaires autres qu'InnoDB et les fichiers temporaires non persistants. Le stockage local inclut notamment des fichiers qui sont utilisés à des fins telles que le tri de grands jeux de données pendant le traitement des requêtes ou les opérations de génération d'index. Il n'inclut pas les tables temporaires InnoDB.

Pour plus d'informations sur les tables temporaires dans Aurora MySQL version 3, consultez Nouveau comportement de table temporaire dans Aurora MySQL version 3. Pour plus d'informations sur les tables temporaires dans la version 2, consultez Comportement d'espace de table temporaire dans Aurora MySQL version 2.

Les données et les fichiers temporaires de ces volumes sont perdus lors du démarrage et de l'arrêt de l'instance de base de données, ainsi que lors du remplacement de l'hôte.

Ces volumes de stockage locaux sont basés sur Amazon Elastic Block Store (EBS) et peuvent être étendus en utilisant une classe d'instance de base de données plus grande. Pour plus d'informations sur le stockage, consultez Stockage Amazon Aurora.

Le stockage local est également utilisé pour importer des données depuis Amazon S3 à l'aide de `LOAD DATA FROM S3` ou `LOAD XML FROM S3`, et pour exporter des données vers S3 à l'aide de SELECT INTO OUTFILE S3. Pour plus d'informations sur l'importation à partir de S3 et l'exportation vers S3, consultez les détails suivants :

- Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3
- Enregistrement de données d'un cluster de bases de données Amazon Aurora MySQL dans des fichiers texte stockés dans un compartiment Amazon S3

Aurora MySQL utilise un stockage permanent distinct pour les journaux d'erreurs, les journaux généraux, les journaux de requêtes lentes et les journaux d'audit pour la plupart des classes d'instance de base de données Aurora MySQL (à l'exception des types de classes d'instance à

performances évolutives tels que db.t2, db.t3 et db.t4g). Les données de ce volume sont conservées lors du démarrage et de l'arrêt de l'instance de base de données, ainsi que lors du remplacement de l'hôte.

Ce volume de stockage permanent est également basé sur Amazon EBS et a une taille fixe en fonction de la classe d'instance de base de données. Il ne peut pas être étendu en utilisant une classe d'instance de base de données plus grande.

Le tableau suivant indique la quantité maximale de stockage temporaire et permanent disponible pour chaque classe d'instance de base de données Aurora MySQL. Pour plus d'informations sur la prise en charge d'une classe d'instance de base de données pour Aurora, consultez Classes d'instance de base de données Amazon Aurora.

Classe d'instance de base de données	Stockage temporaire/local maximal disponible (Gio)	Stockage maximal supplémentaire disponible pour les fichiers journaux (Gio)
db.x2g.16xlarge	1280	500
db.x2g.12xlarge	960	500
db.x2g.8xlarge	640	500
db.x2g.4xlarge	320	500
db.x2g.2xlarge	160	60
db.x2g.xlarge	80	60
db.x2g.large	40	60
db.r8g.48xlarge	3840	500
db.r8g.24xlarge	1920	500
db.r8g.16xlarge	1280	500
db.r8g.12xlarge	960	500
db.r8g.8xlarge	640	500

Classe d’instance de base de données	Stockage temporaire/local maximal disponible (Gio)	Stockage maximal supplémentaire disponible pour les fichiers journaux (Gio)
db.r8g.4xlarge	320	500
db.r8g.2xlarge	160	60
db.r8g.xlarge	80	60
db.r8g.large	32	60
db.r7i.48xlarge	3840	500
db.r7i.24xlarge	1920	500
db.r7i.16xlarge	1280	500
db.r7i.12xlarge	960	500
db.r7i.8xlarge	640	500
db.r7i.4xlarge	320	500
db.r7i.2xlarge	160	60
db.r7i.xlarge	80	60
db.r7i.large	32	60
db.r7g.16xlarge	1280	500
db.r7g.12xlarge	960	500
db.r7g.8xlarge	640	500
db.r7g.4xlarge	320	500
db.r7g.2xlarge	160	60
db.r7g.xlarge	80	60

Classe d’instance de base de données	Stockage temporaire/local maximal disponible (Gio)	Stockage maximal supplémentaire disponible pour les fichiers journaux (Gio)
db.r7g.large	32	60
db.r6i.32xlarge	2560	500
db.r6i.24xlarge	1920	500
db.r6i.16xlarge	1280	500
db.r6i.12xlarge	960	500
db.r6i.8xlarge	640	500
db.r6i.4xlarge	320	500
db.r6i.2xlarge	160	60
db.r6i.xlarge	80	60
db.r6i.large	32	60
db.r6g.16xlarge	1280	500
db.r6g.12xlarge	960	500
db.r6g.8xlarge	640	500
db.r6g.4xlarge	320	500
db.r6g.2xlarge	160	60
db.r6g.xlarge	80	60
db.r6g.large	32	60
db.r5.24xlarge	1920	500
db.r5.16xlarge	1280	500

Classe d’instance de base de données	Stockage temporaire/local maximal disponible (Gio)	Stockage maximal supplémentaire disponible pour les fichiers journaux (Gio)
db.r5.12xlarge	960	500
db.r5.8xlarge	640	500
db.r5.4xlarge	320	500
db.r5.2xlarge	160	60
db.r5.xlarge	80	60
db.r5.large	32	60
db.r4.16xlarge	1280	500
db.r4.8xlarge	640	500
db.r4.4xlarge	320	500
db.r4.2xlarge	160	60
db.r4.xlarge	80	60
db.r4.large	32	60
db.t4g.large	32	–
db.t4g.medium	32	–
db.t3.large	32	–
db.t3.medium	32	–
db.t3.small	32	–
db.t2.medium	32	–
db.t2.small	32	–

> **Important**
>
> Ces valeurs représentent la quantité maximale théorique de stockage disponible sur chaque instance de base de données. Le stockage local réel à votre disposition pourrait être inférieur. Aurora utilise du stockage local pour ses processus de gestion, et l'instance de base de données utilise du stockage local avant même que vous chargiez des données. Vous pouvez surveiller le stockage temporaire disponible pour une instance de base de données spécifique à l'aide de la métrique `FreeLocalStorage` CloudWatch décrite dans CloudWatch Métriques Amazon pour Amazon Aurora. Vous pouvez vérifier la quantité de stockage disponible à l'heure actuelle. Vous pouvez également représenter la quantité de stockage disponible au fil du temps. La surveillance du stockage disponible au fil du temps vous aide à déterminer si la valeur augmente ou diminue, mais aussi à trouver les valeurs minimales, maximales ou moyennes.
> (Cela ne s'applique pas à Aurora Serverless v2).

Retour en arrière d'un cluster de bases de données Aurora

Édition compatible avec Amazon Aurora MySQL vous permet d'effectuer un retour en arrière d'un cluster de bases de données à une heure spécifique, sans restaurer les données à partir d'une sauvegarde.

Table des matières

Présentation du retour en arrière

Le retour en arrière permet de « faire revenir en arrière » le cluster de bases de données à l'heure que vous spécifiez. Le retour en arrière ne remplace pas une sauvegarde de votre cluster de bases de données afin de pouvoir la restaurer à un instant dans le passé. Toutefois, le retour en arrière fournit les avantages suivants par rapport aux fonctions traditionnelles de sauvegarde et de restauration :

- Vous pouvez facilement annuler des erreurs. Si vous effectuez par erreur une action destructrice, comme une opération DELETE sans clause WHERE, vous pouvez exécuter un retour en arrière pour faire revenir le cluster de bases de données à une heure précédant l'action destructrice sans aucune interruption minimale du service.
- Vous pouvez effectuer rapidement un retour en arrière d'un cluster de bases de données. La restauration d'un cluster de bases de données à un instant dans le passé lance un nouveau cluster de bases de données et restaure celui-ci à partir des données de sauvegarde ou d'un instantané de cluster de bases de données ; cette opération peut prendre plusieurs heures. Le retour en arrière d'un cluster de bases de données ne nécessite pas de nouveau cluster de bases de données et fait revenir en arrière le cluster de bases de données en quelques minutes.
- Vous pouvez explorer les précédentes modifications de données. Vous pouvez effectuer des retours en arrière d'un cluster de bases de données de manière répétée en arrière et en avant dans le temps afin de déterminer le moment où une modification particulière a eu lieu. Par exemple, vous pouvez effectuer un retour en arrière d'un cluster de bases de données de trois heures, puis effectuer un autre retour en arrière en avant d'une heure. Dans ce cas, l'heure du retour en arrière est antérieure de deux heures par rapport à l'heure d'origine.

> **Note**
>
> Pour plus d'informations sur la restauration d'un cluster de bases de données à un instant dans le passé, consultez Présentation de la sauvegarde et de la restauration d'un cluster de bases de données Aurora.

Fenêtre de retour en arrière

Avec le retour en arrière, il y a une fenêtre de retour en arrière cible et une fenêtre de retour en arrière réelle :

- La fenêtre de retour en arrière cible correspond au laps de temps pendant lequel vous souhaitez pouvoir effectuer un retour en arrière de votre cluster de bases de données. Lorsque vous activez le retour en arrière, vous spécifiez une fenêtre de retour en arrière cible. Par exemple, vous pouvez spécifier une fenêtre de retour en arrière cible de 24 heures si vous souhaitez pouvoir effectuer un retour en arrière d'une journée du cluster de bases de données.
- La fenêtre de retour en arrière réelle correspond au laps de temps réel pendant lequel vous pouvez effectuer un retour en arrière de votre cluster de bases de données ; sa valeur peut être inférieure à celle de la fenêtre de retour en arrière cible. La fenêtre de retour en arrière réelle est basée sur votre charge de travail et sur le stockage disponible pour les informations sur les modifications de la base de données, appelées enregistrements de modification.

À mesure que vous effectuez des mises à jour de votre cluster de bases de données Aurora avec le retour en arrière activé, vous générez des enregistrements de modifications. Aurora conserve les enregistrements de modifications pour la fenêtre de retour en arrière cible, et leur stockage vous est facturé sur une base horaire. La fenêtre de retour en arrière cible et la charge de travail sur votre cluster de bases de données déterminent le nombre d'enregistrements de modification que vous stockez. La charge de travail correspond au nombre de modifications que vous apportez au cluster de bases de données sur un laps de temps donné. Si votre charge de travail est lourde, vous stockez davantage d'enregistrements dans votre fenêtre de retour en arrière que si votre charge de travail est légère.

Vous pouvez considérer votre fenêtre de retour en arrière cible comme étant l'objectif du laps de temps maximal pendant lequel vous souhaitez pouvoir faire un retour en arrière de votre cluster de bases de données. Dans la plupart des cas, vous pouvez effectuer un retour en arrière correspondant au laps de temps maximal spécifié. Toutefois, dans certains cas, le cluster de bases de données ne peut pas stocker suffisamment d'enregistrements de modification pour effectuer un retour en arrière correspondant au laps de temps maximal, et votre fenêtre de retour en arrière réelle est plus petite que votre fenêtre de retour en arrière cible. Généralement, la fenêtre de retour en arrière réelle est plus petite que la fenêtre de retour en arrière cible lorsque la charge de travail est particulièrement lourde sur votre cluster de bases de données. Lorsque votre fenêtre de retour en arrière réelle est plus petite que votre fenêtre de retour en arrière cible, nous vous envoyons une notification.

Lorsque le retour en arrière est activé pour un cluster de bases de données, et que vous supprimez une table stockée dans ce cluster, Aurora conserve cette table dans les enregistrements de modification du retour en arrière. Ainsi, vous pouvez revenir en arrière à une heure antérieure à celle où vous avez supprimé la table. Si vous ne disposez pas de suffisamment d'espace dans

votre fenêtre de retour en arrière pour stocker la table, il est possible que celle-ci soit supprimée des enregistrements de modification du retour en arrière.

Heure de retour en arrière

Aurora procède toujours au retour en arrière à une heure cohérente pour le cluster de bases de données. Cela élimine la possibilité de transactions non validées une fois le retour en arrière terminé. Lorsque vous spécifiez une heure de retour en arrière, Aurora choisit automatiquement l'heure cohérente la plus proche possible. Cette approche signifie que le retour en arrière terminé peut ne pas correspondre exactement à l'heure que vous spécifiez, mais vous pouvez déterminer l'heure exacte d'un retour en arrière à l'aide de la commande describe-db-cluster-backtracks AWS CLI. Pour de plus amples informations, veuillez consulter Extraction de retours sur trace existants.

Limites du retour en arrière

Les limites suivantes s'appliquent à un retour en arrière :

- Le retour en arrière est disponible uniquement pour les clusters de bases de données créés avec la fonction de retour en arrière activée. Vous ne pouvez pas modifier un cluster de bases de données pour activer le retour en arrière. Vous pouvez activer la fonction de retour en arrière lorsque vous créez un nouveau cluster de bases de données, restaurez un instantané de cluster de bases de données.
- La limite pour une fenêtre de retour en arrière est de 72 heures.
- Le retour en arrière affecte l'ensemble du cluster de bases de données. Par exemple, vous ne pouvez pas effectuer un retour en arrière sélectif sur une seule table ou une seule mise à jour de données.
- Vous ne pouvez pas créer de réplicas lecture entre régions à partir d'un cluster compatible avec le retour en arrière, mais vous pouvez toujours activer la réplication des journaux binaires (binlog) sur le cluster. Si vous essayez d'effectuer un retour en arrière sur un cluster de bases de données pour lequel la journalisation binaire est activée, une erreur se produit généralement, sauf si vous avez choisi de forcer le retour en arrière. Toute tentative visant à forcer un retour en arrière interrompra les répliques de lecture en aval et interférera avec d'autres opérations telles que blue/green les déploiements.
- Vous ne pouvez pas effectuer un retour en arrière d'un clone de base de données à une heure antérieure à l'heure à laquelle le clone a été créé. Toutefois, vous pouvez utiliser la base de données d'origine pour effectuer un retour en arrière à une heure antérieure à l'heure à laquelle le clone a été créé. Pour plus d'informations sur le clonage de base de données, consultez Clonage d'un volume pour un cluster de bases de données Amazon Aurora.

- Le retour en arrière entraîne une brève interruption de l’instance de base de données. Vous devez arrêter ou mettre en pause vos applications avant de démarrer une opération de retour en arrière, afin de vous assurer qu’il n’y a aucune nouvelle demande en lecture ou en écriture. Au cours de l’opération de retour en arrière, Aurora met en pause la base de données, ferme les connexions ouvertes et annule toutes les lectures et écritures non enregistrées. Il attend ensuite que l’opération de retour en arrière se termine.
- Vous ne pouvez pas restaurer un instantané interrégional d'un cluster compatible avec le retour en arrière dans une AWS région qui ne prend pas en charge le retour en arrière.
- Si vous effectuez une mise à niveau sur place de la version 2 vers la version 3 d’Aurora MySQL pour un cluster prenant en charge le retour en arrière, vous ne pouvez pas effectuer un retour en arrière à une heure antérieure à celle où la mise à jour a eu lieu.

Disponibilité des régions et des versions

Le retour en arrière n’est pas disponible pour Aurora PostgreSQL.

Voici les moteurs pris en charge et la disponibilité des régions pour le retour en arrière avec Aurora MySQL.

Région	Aurora MySQL version 3	Aurora MySQL version 2
USA Est (Virginie du Nord)	Toutes les versions	Toutes les versions
USA Est (Ohio)	Toutes les versions	Toutes les versions
USA Ouest (Californie du Nord)	Toutes les versions	Toutes les versions
USA Ouest (Oregon)	Toutes les versions	Toutes les versions
Afrique (Le Cap)	–	–
Asie-Pacifique (Hong Kong)	–	–

Région	Aurora MySQL version 3	Aurora MySQL version 2
Asie-Pacifique (Jakarta)	–	–
Asie-Pacifique (Malaisie)	–	–
Asie-Pacifique (Melbourne)	–	–
Asie-Pacifique (Mumbai)	Toutes les versions	Toutes les versions
Asie-Pacifique (Nouvelle-Zélande)	–	–
Asie-Pacifique (Osaka)	Toutes les versions	Versions 2.07.3 et ultérieures
Asie-Pacifique (Séoul)	Toutes les versions	Toutes les versions
Asie-Pacifique (Singapour)	Toutes les versions	Toutes les versions
Asie-Pacifique (Sydney)	Toutes les versions	Toutes les versions
Asie-Pacifique (Taipei)	–	–
Asie-Pacifique (Thaïlande)	–	–
Asie-Pacifique (Tokyo)	Toutes les versions	Toutes les versions
Canada (Centre)	Toutes les versions	Toutes les versions

Région	Aurora MySQL version 3	Aurora MySQL version 2
Canada-Ouest (Calgary)	–	–
Chine (Pékin)	–	–
China (Ningxia)	–	–
Europe (Francfort)	Toutes les versions	Toutes les versions
Europe (Irlande)	Toutes les versions	Toutes les versions
Europe (Londres)	Toutes les versions	Toutes les versions
Europe (Milan)	–	–
Europe (Paris)	Toutes les versions	Toutes les versions
Europe (Espagne)	–	–
Europe (Stockholm)	–	–
Europe (Zurich)	–	–
Israël (Tel Aviv)	–	–
Mexique (Centre)	–	–
Middle East (Bahrain)	–	–
Moyen-Orient (EAU)	–	–

Région	Aurora MySQL version 3	Aurora MySQL version 2
Amérique du Sud (São Paulo)	–	–
AWS GovCloud (USA Est)	–	–
AWS GovCloud (US-Ouest)	–	–

Considérations relatives aux mises à niveau pour les clusters compatibles avec le retour en arrière

Vous pouvez mettre à niveau un cluster de bases de données prenant en charge le retour en arrière d'Aurora MySQL version 2 vers la version 3, car toutes les versions mineures d'Aurora MySQL version 3 sont prises en charge pour le retour en arrière.

Configuration du retour en arrière d'un cluster de bases de données Aurora MySQL

Pour utiliser le retour sur trace, vous devez activez la fonction et spécifier une fenêtre de retour sur trace cible. Dans le cas contraire, le retour sur trace est désactivé.

Pour la fenêtre de retour sur trace cible, spécifiez le laps de temps pendant lequel vous souhaitez pouvoir faire revenir en arrière votre base de données en utilisant le retour sur trace. Aurora tente de conserver suffisamment d'enregistrements de modification pour prendre en charge cette fenêtre de temps.

Console

Vous pouvez utiliser la console pour configurer le retour en arrière lorsque vous créez un nouveau cluster de bases de données. Vous pouvez également modifier un cluster de bases de données pour modifier la fenêtre de retour en arrière d'un cluster compatible avec le retour en arrière. Si vous désactivez entièrement le retour sur trace pour un cluster en définissant la fenêtre de retour sur trace sur 0, vous ne pouvez pas activer le retour sur trace à nouveau pour ce cluster.

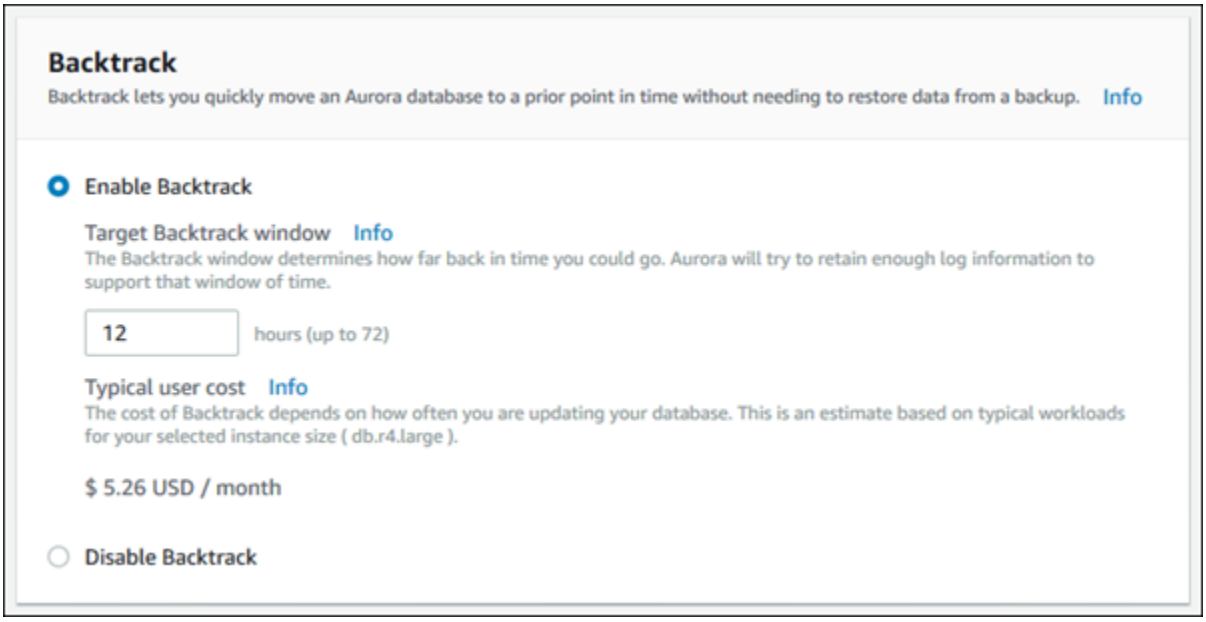
Rubriques

- Configuration du retour en arrière avec la console lors de la création d'un cluster de bases de données

Configuration du retour en arrière avec la console lors de la création d'un cluster de bases de données

Lorsque vous créez un cluster de bases de données Aurora MySQL, la configuration du retour en arrière consiste à choisir Enable Backtrack (Activer le retour en arrière) et à spécifier pour Target Backtrack window (Fenêtre de retour en arrière cible) une valeur supérieure à zéro dans la section Retour en arrière.

Pour créer un cluster de bases de données, suivez les instructions de Création d'un cluster de bases de données Amazon Aurora. L'image suivante montre la section Retour en arrière.



Lorsque vous créez un nouveau cluster de bases de données, Aurora n'a aucune donnée pour la charge de travail du cluster de bases de données. Il ne peut donc pas estimer de coût spécifique pour le nouveau cluster de bases de données. Cependant, la console présente un coût utilisateur classique pour la fenêtre de retour sur trace cible spécifiée, basé sur une charge de travail habituelle. Le coût classique permet de fournir une référence générale pour le coût de la fonction de retour sur trace.

> **Important**
>
> Votre coût réel peut être différent du coût classique, puisqu'il est basé sur la charge de travail de votre cluster de bases de données.

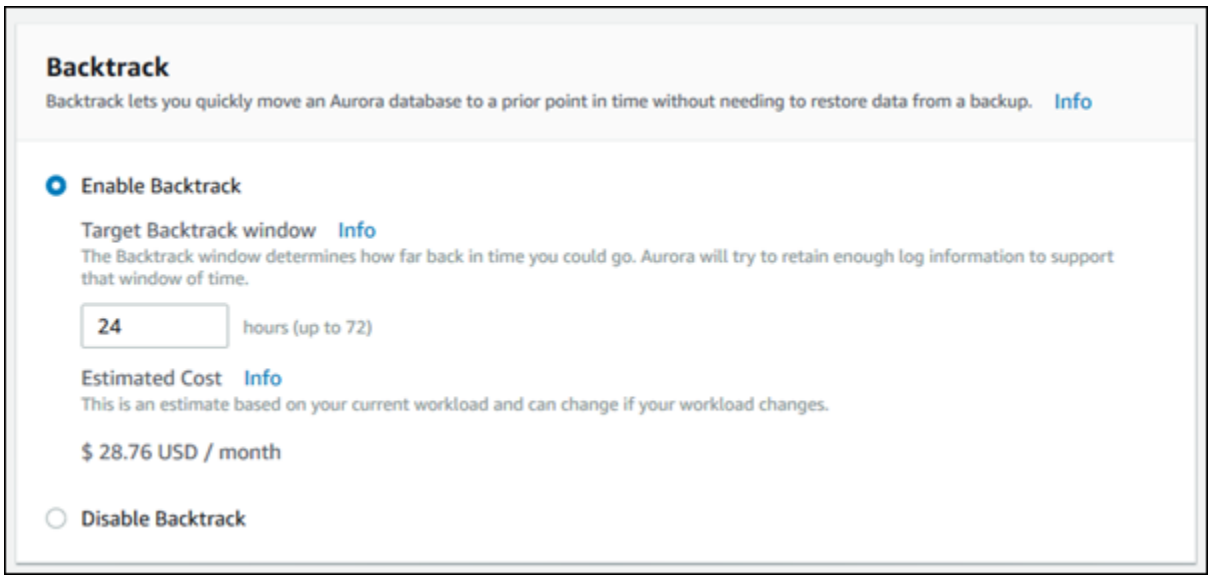
Configuration du retour en arrière avec la console lors de la modification d'un cluster de bases de données

Vous pouvez modifier le retour en arrière pour un cluster de bases de données à l'aide de la console.

> **Note**
>
> Actuellement, vous pouvez modifier le retour en arrière uniquement pour un cluster de bases de données dont la fonction de retour en arrière est activée. La section Retour en arrière n'apparaît pas pour un cluster de bases de données créé avec la fonction de retour en arrière désactivée ou si cette fonction a été désactivée pour le cluster de bases de données.

Pour modifier le retour en arrière pour un cluster de bases de données à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez le cluster que vous souhaitez modifier, puis choisissez Modifier.
4. Pour Target Backtrack window (Fenêtre de retour sur trace cible), modifiez le laps de temps pendant lequel vous souhaitez pouvoir effectuer un retour sur trace. La limite est de 72 heures.



La console indique le coût estimé pour le laps de temps que vous avez spécifié, basé sur la charge de travail précédente du cluster de bases de données :

- Si le retour en arrière était désactivé sur le cluster de bases de données, le coût estimé est basé sur la métrique `VolumeWriteIOPS` pour le cluster de bases de données dans Amazon CloudWatch.
- Si le retour en arrière était activé précédemment sur le cluster de bases de données, le coût estimé est basé sur la métrique `BacktrackChangeRecordsCreationRate` pour le cluster de bases de données dans Amazon CloudWatch.

5. Choisissez Continuer.
6. Pour Scheduling of Modifications (Planification des modifications), choisissez une des options suivantes :

   - Apply during the next scheduled maintenance window (Appliquer lors de la prochaine fenêtre de maintenance planifiée) – Attend la prochaine fenêtre de maintenance avant d'appliquer la modification de Target Backtrack window (Fenêtre de retour en arrière cible).
   - Apply immediately (Appliquer immédiatement) – Applique la modification de Target Backtrack window (Fenêtre de retour sur trace cible) dès que possible.

7. Choisissez Modifier le cluster.

AWS CLI

Lorsque vous créez un cluster de bases de données Aurora MySQL à l'aide de la commande CLI de l'AWS create-db-cluster, le retour en arrière est configuré lorsque vous spécifiez une valeur `--backtrack-window` supérieure à zéro. La valeur `--backtrack-window` spécifie la fenêtre de retour sur trace cible. Pour plus d'informations, consultez Création d'un cluster de bases de données Amazon Aurora.

Vous pouvez également spécifier la valeur `--backtrack-window` à l'aide des commandes de l'AWS CLI suivantes :

- modify-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-snapshot
- restore-db-cluster-to-point-in-time

La procédure suivante explique comment modifier la fenêtre de retour en arrière cible pour un cluster de bases de données à l'aide de l AWS CLI.

Pour modifier la fenêtre de retour en arrière cible pour un cluster de bases de données à l'aide de l AWS CLI

- Appelez la commande modify-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :
  - `--db-cluster-identifier` : nom du cluster de bases de données.
  - `--backtrack-window` – Nombre maximal de secondes pendant lesquelles vous souhaitez pouvoir faire un retour en arrière de cluster de bases de données.

  L'exemple suivant définit la fenêtre de retour en arrière cible pour `sample-cluster` à une journée (86 400 secondes).

  Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-window 86400
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-window 86400
```

> **Note**
>
> Actuellement, vous pouvez activer le retour en arrière uniquement pour un cluster de bases de données créé avec la fonction de retour en arrière activée.

API RDS

Lorsque vous créez un cluster de bases de données Aurora MySQL en utilisant l'opération CreateDBCluster de l'API Amazon RDS, le retour en arrière est configuré lorsque vous spécifiez une valeur supérieure à zéro pour `BacktrackWindow`. La valeur `BacktrackWindow` spécifie la fenêtre de retour en arrière cible pour le cluster de bases de données spécifié dans la valeur `DBClusterIdentifier`. Pour plus d'informations, consultez Création d'un cluster de bases de données Amazon Aurora.

Vous pouvez également spécifier la valeur `BacktrackWindow` à l'aide des opérations d'API suivantes :

- ModifyDBCluster
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime

> **Note**
>
> Actuellement, vous pouvez activer le retour en arrière uniquement pour un cluster de bases de données créé avec la fonction de retour en arrière activée.

Exécution d’un retour en arrière pour un cluster de bases de données MySQL

Vous pouvez effectuer un retour en arrière pour un cluster de bases de données à un horodatage de retour en arrière spécifié. Si l’horodatage de retour en arrière n’est pas antérieur à l’heure de retour en arrière la plus ancienne possible et ne se situe pas dans le futur, le retour en arrière du cluster de bases de données est effectué à cet horodatage.

Dans le cas contraire, une erreur se produit généralement. Par ailleurs, si vous essayez d’effectuer un retour en arrière sur un cluster de bases de données pour lequel la journalisation binaire est activée, une erreur se produit généralement, sauf si vous avez choisi de forcer l’exécution du retour en arrière. Forcer un retour en arrière peut interférer avec d’autres opérations utilisant la journalisation binaire.

Important

Le retour en arrière ne génère aucune entrée binlog pour les modifications qu’il effectue. Si la journalisation binaire est activée pour le cluster de bases de données, il est possible que le retour en arrière ne soit pas compatible avec votre implémentation binlog.

Note

Pour les clones de base de données, vous ne pouvez pas effectuer un retour en arrière du cluster de bases de données à une heure antérieure à l’heure à laquelle le clone a été créé. Pour plus d’informations sur le clonage de base de données, consultez Clonage d’un volume pour un cluster de bases de données Amazon Aurora.

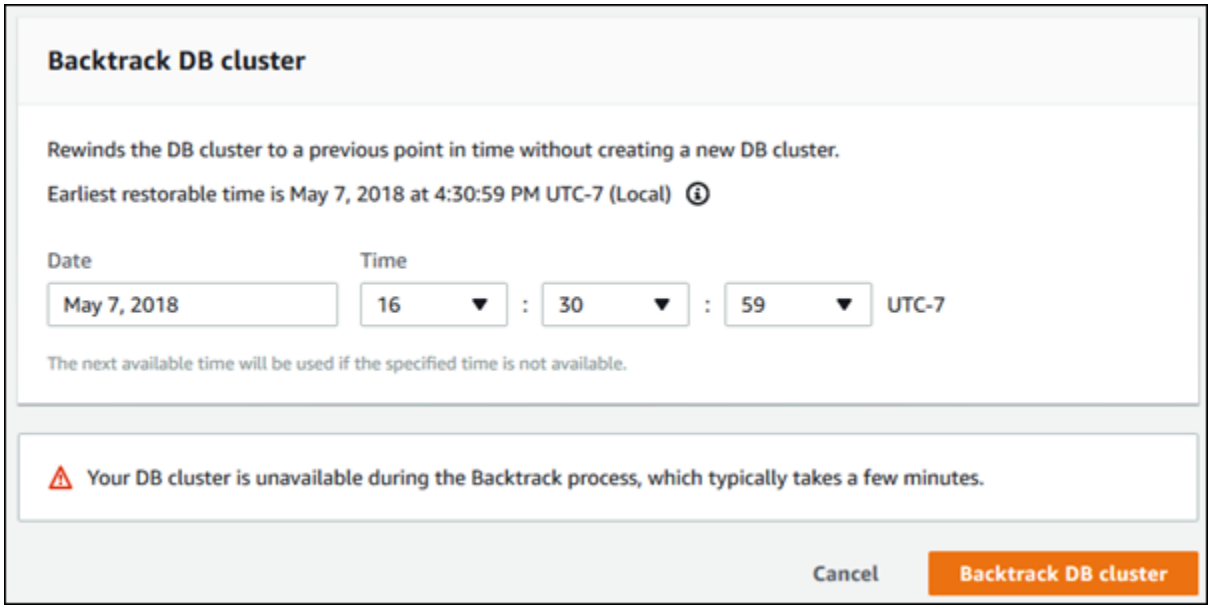
Console

La procédure suivante explique comment effectuer une opération de retour en arrière pour un cluster de bases de données à l’aide de la console.

Pour effectuer une opération de retour en arrière à l’aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Instances.

3. Choisissez l'instance principale du cluster de bases de données pour lequel vous souhaitez effectuer un retour en arrière.
4. Pour Actions, choisissez Backtrack DB cluster (Retour en arrière de cluster de bases de données).
5. Sur la page Backtrack DB cluster (Retour en arrière de cluster de bases de données), entrez l'horodatage de retour en arrière à appliquer au retour en arrière de cluster de bases de données.



6. Choisissez Backtrack DB cluster (Retour en arrière de cluster de bases de données).

AWS CLI

La procédure suivante explique comment effectuer un retour en arrière de cluster de bases de données à l'aide de l AWS CLI.

Pour effectuer un retour en arrière de cluster de bases de données à l'aide de l AWS CLI

- Appelez la commande backtrack-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :
  - `--db-cluster-identifier` : nom du cluster de bases de données.
  - `--backtrack-to` – Horodatage de retour en arrière de cluster de bases de données, spécifié au format ISO 8601.

L'exemple suivant effectue un retour en arrière du cluster de bases de données `sample-cluster` à 10 h, le 19 mars 2018.

Pour Linux, macOS ou Unix :

```
aws rds backtrack-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-to 2018-03-19T10:00:00+00:00
```

Pour Windows :

```
aws rds backtrack-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-to 2018-03-19T10:00:00+00:00
```

API RDS

Pour effectuer un retour en arrière de cluster de bases de données à l'aide de l'API Amazon RDS, utilisez l'opération BacktrackDBCluster. Cette opération effectue un retour en arrière du cluster de bases de données spécifié dans la valeur `DBClusterIdentifier` à l'heure spécifiée.

Surveillance du retour en arrière pour un cluster de bases de données Aurora MySQL

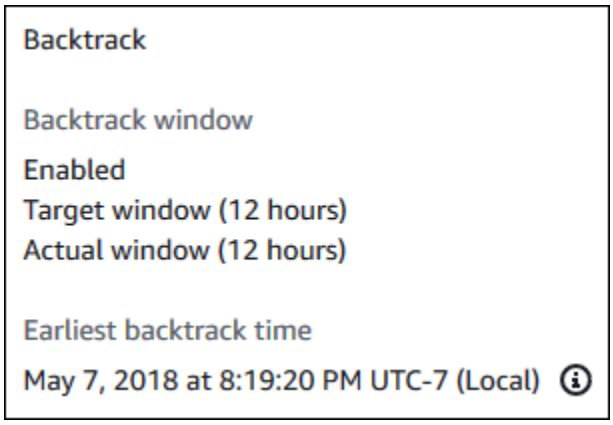
Vous pouvez afficher les informations et surveiller les métriques de retour en arrière pour un cluster de bases de données.

Console

Pour afficher les informations et surveiller les métriques de retour en arrière à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez le nom du cluster de bases de données pour lequel vous souhaitez afficher les informations.

Les informations de retour sur trace se trouvent dans la section Retour sur trace.



Lorsque le retour sur trace est activé, les informations suivantes sont disponibles :

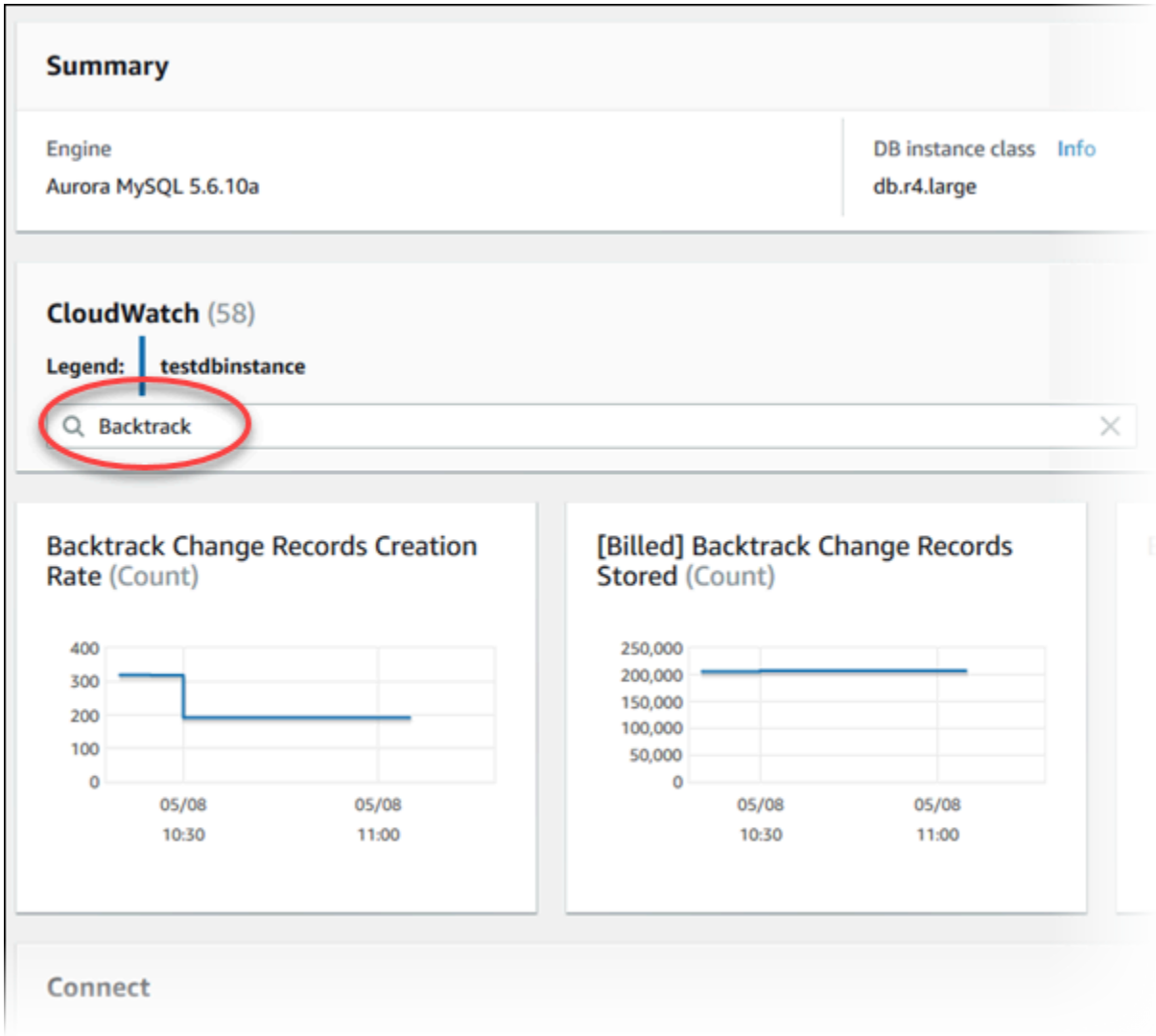
- Target window (Fenêtre cible) – Laps de temps actuel spécifié pour la fenêtre de retour sur trace cible. La cible correspond au laps de temps maximal pendant lequel vous pouvez effectuer un retour sur trace si le stockage est suffisant.
- Actual window (Fenêtre réelle) – Laps de temps réel pendant lequel vous pouvez effectuer un retour sur trace, qui peut être inférieur à celui de la fenêtre de retour sur trace cible. La fenêtre de retour sur trace réelle est basée sur votre charge de travail et sur le stockage disponible pour conserver les enregistrements de modification du retour sur trace.
- Date du retour en arrière le plus ancien – Date de retour en arrière le plus ancien possible pour le cluster de bases de données. Vous ne pouvez pas effectuer un retour en arrière du cluster de bases de données à un horodatage antérieure à l'heure affichée.

4. Procédez comme suit pour afficher les métriques de retour en arrière pour le cluster de bases de données :

   a. Dans le panneau de navigation, choisissez Instances.

   b. Choisissez le nom de l'instance principale pour le cluster de bases de données dont vous voulez afficher les détails.

   c. Dans la section CloudWatch, entrez **`Backtrack`** dans la zone CloudWatch pour afficher uniquement les métriques de retour sur trace.



Les métriques suivantes sont affichées :

- Backtrack Change Records Creation Rate (Count) (Taux de création d'enregistrements de modification de retour en arrière (nombre)) – Cette métrique affiche le nombre d'enregistrements de modification de retour en arrière créés en 5 minutes pour votre cluster de bases de données. Vous pouvez utiliser cette métrique pour estimer le coût du retour sur trace pour votre fenêtre de retour sur trace cible.
- [Billed] Backtrack Change Records Stored (Count) ([Facturé] Enregistrements de modification de retour en arrière stockés (nombre)) – Cette métrique affiche le nombre réel d'enregistrements de modification de retour en arrière utilisés par votre cluster de bases de données.
- Backtrack Window Actual (Minutes) (Fenêtre de retour en arrière réelle (minutes)) – Cette métrique indique s'il y a une différence entre la fenêtre de retour en arrière cible et la

fenêtre de retour en arrière réelle. Par exemple, si votre fenêtre de retour sur trace cible est de 2 heures (120 minutes) et que cette métrique indique 100 minutes pour la fenêtre de retour sur trace réelle, la fenêtre de retour sur trace réelle est plus petite que la fenêtre de retour sur trace cible.

- Backtrack Window Alert (Count) (Alerte de fenêtre de retour sur trace (nombre)) – Cette métrique indique le nombre de fois où la fenêtre de retour sur trace réelle est plus petite que la fenêtre de retour sur trace cible pour un laps de temps donné.

> **Note**
>
> Les métriques suivantes peuvent avoir du retard par rapport à l'heure réelle :
>
> - Backtrack Change Records Creation Rate (Count) (Taux de création d'enregistrements de modification de retour en arrière (nombre))
> - [Billed] Backtrack Change Records Stored (Count) ([Facturé] Enregistrements de modification de retour sur trace stockés (nombre))

AWS CLI

La procédure suivante explique comment afficher des informations de retour en arrière pour un cluster de bases de données à l'aide de l AWS CLI.

Pour afficher des informations de retour en arrière pour un cluster de bases de données à l'aide de l AWS CLI

- Appelez la commande describe-db-clusters de l'AWS CLI et fournissez les valeurs suivantes :
  - `--db-cluster-identifier` : nom du cluster de bases de données.

  L'exemple suivant affiche les informations de retour en arrière pour `sample-cluster`.

  Pour Linux, macOS ou Unix :

```
aws rds describe-db-clusters \
    --db-cluster-identifier sample-cluster
```

Pour Windows :

```
aws rds describe-db-clusters ^
    --db-cluster-identifier sample-cluster
```

API RDS

Pour afficher des informations de retour en arrière pour un cluster de bases de données à l'aide de l'API Amazon RDS, utilisez l'opération DescribeDBClusters. Cette opération renvoie des informations sur les retours en arrière pour le cluster de bases de données spécifié dans la valeur `DBClusterIdentifier`.

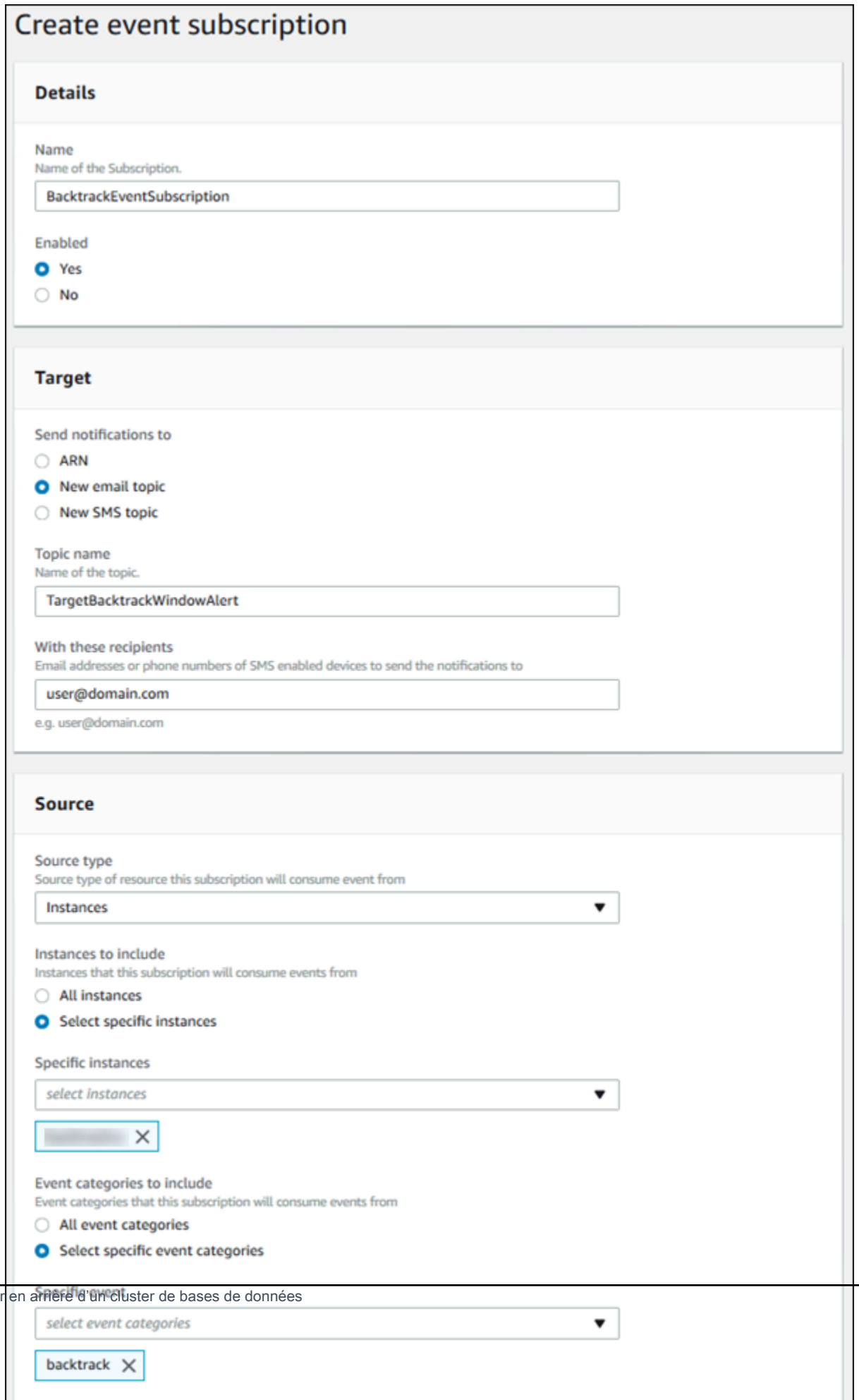
Abonnement à un événement de retour en arrière avec la console

La procédure suivante explique comment s'abonner à un événement de retour en arrière à l'aide de la console. L'événement vous envoie un e-mail ou une notification lorsque votre fenêtre de retour en arrière réelle est plus petite que votre fenêtre de retour en arrière cible.

Pour afficher des informations de retour en arrière à l'aide de la console

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Choisissez Abonnements aux événements.
3. Choisissez Créer un abonnement aux événements.
4. Dans la zone Nom, attribuez un nom à l'abonnement aux événements et vérifiez que Oui est sélectionné pour Activé.
5. Dans la section Target (Cible), choisissez New email topic (Nouvelle rubrique d'e-mail).
6. Dans Nom de la rubrique, attribuez un nom à la rubrique, puis indiquez les adresses e-mail ou les numéros de téléphone qui recevront les notifications dans Avec ces destinataires.
7. Dans la section Source, choisissez Instances pour Type de source.
8. Pour Instances to include (Instances à inclure), choisissez Select specific instances (Sélectionner des instances spécifiques), puis sélectionnez votre instance de base de données.
9. Pour Event categories to include (Catégories d'événement à inclure), choisissez Select specific event categories (Sélectionner des catégories d'événement spécifiques), puis sélectionnez backtrack (retour en arrière).

Votre page doit ressembler à la page suivante.


Create event subscription
Details
Name
Name of the Subscription.
BacktrackEventSubscription
Enabled
Yes
No
Target
Send notifications to
ARN
New email topic
New SMS topic
Topic name
Name of the topic.
TargetBacktrackWindowAlert
With these recipients
Email addresses or phone numbers of SMS enabled devices to send the notifications to
user@domain.com
e.g. user@domain.com
Source
Source type
Source type of resource this subscription will consume event from
Instances
Instances to include
Instances that this subscription will consume events from
All instances
Select specific instances
Specific instances
select instances
Event categories to include
Event categories that this subscription will consume events from
All event categories
Select specific event categories
select event categories
backtrack

10. Sélectionnez Créer.

Extraction de retours sur trace existants

Vous pouvez extraire des informations sur des retours en arrière existants pour un cluster de bases de données. Ces informations incluent l'identifiant unique du retour en arrière, la date et l'heure de destination et d'origine du retour en arrière, la date et l'heure de la demande de retour en arrière et l'état actuel du retour en arrière.

> **Note**
>
> Actuellement, vous ne pouvez pas extraire des retours en arrière existants à l'aide de la console.

AWS CLI

La procédure suivante explique comment extraire des retours en arrière existants pour un cluster de bases de données à l'aide de l AWS CLI.

Pour récupérer des backtracks existants à l'aide du AWS CLI

- Appelez la commande describe-db-cluster-backtracks AWS CLI et fournissez les valeurs suivantes :
  - `--db-cluster-identifier` : nom du cluster de bases de données.

  L'exemple suivant extrait les retours en arrière existants pour `sample-cluster`.

  Pour Linux, macOS ou Unix :

```
aws rds describe-db-cluster-backtracks \
    --db-cluster-identifier sample-cluster
```

  Pour Windows :

```
aws rds describe-db-cluster-backtracks ^
```

```
    --db-cluster-identifier sample-cluster
```

API RDS

Pour récupérer des informations sur les backtracks d'un cluster de bases de données à l'aide de l'API Amazon RDS, utilisez l'opération Describe DBCluster Backtracks. Cette opération renvoie des informations sur les retours en arrière pour le cluster de bases de données spécifié dans la valeur `DBClusterIdentifier`.

Désactivation du retour en arrière pour un cluster de bases de données Aurora MySQL

Vous pouvez désactiver la fonction de retour en arrière pour un cluster de bases de données.

Console

Vous pouvez désactiver le retour en arrière pour un cluster de bases de données à l'aide de la console. Après avoir entièrement désactivé le retour en arrière pour un cluster, vous ne pouvez pas l'activer à nouveau pour ce cluster.

Pour désactiver la fonction de retour en arrière pour un cluster de bases de données à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez le cluster que vous souhaitez modifier, puis choisissez Modifier.
4. Dans la section Retour sur trace, choisissez Disable Backtrack (Désactiver le retour sur trace).
5. Choisissez Continuer.
6. Pour Scheduling of Modifications (Planification des modifications), choisissez une des options suivantes :
   - Apply during the next scheduled maintenance window (Appliquer lors de la prochaine fenêtre de maintenance planifiée) – Attend la prochaine fenêtre de maintenance avant d'appliquer la modification.
   - Appliquer immédiatement – Applique la modification dès que possible.
7. Choisissez Modifier le cluster.

AWS CLI

Vous pouvez désactiver la fonction de retour en arrière pour un cluster de bases de données à partir de l'AWS CLIen définissant la fenêtre de retour en arrière cible sur `0` (zéro). Après avoir entièrement désactivé le retour en arrière pour un cluster, vous ne pouvez pas l'activer à nouveau pour ce cluster.

Pour modifier la fenêtre de retour en arrière cible pour un cluster de bases de données à l'aide de l AWS CLI

- Appelez la commande modify-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :
  - `--db-cluster-identifier` : nom du cluster de bases de données.
  - `--backtrack-window` – spécifier `0` pour désactiver le retour sur trace.

  L'exemple suivant désactive la fonction de retour en arrière cible pour `sample-cluster` en configurant `--backtrack-window` à `0`.

  Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --backtrack-window 0
```

  Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --backtrack-window 0
```

API RDS

Pour désactiver la fonction de retour en arrière pour un cluster de bases de données à partir de l'API Amazon RDS, utilisez l'opération ModifyDBCluster. Définissez la valeur de `BacktrackWindow` à `0` (zéro), et spécifiez le cluster de bases de données dans la valeur `DBClusterIdentifier`. Après avoir entièrement désactivé le retour en arrière pour un cluster, vous ne pouvez pas l'activer à nouveau pour ce cluster.

Test d'Amazon Aurora MySQL à l'aide de requêtes d'injection d'erreurs

Vous pouvez tester la tolérance aux pannes de votre cluster de bases de données Aurora MySQL à l'aide des requêtes d'injection d'erreurs. Les requêtes d'injection d'erreurs sont émises sous forme de commandes SQL à une instance Amazon Aurora. Elles vous permettent de programmer une simulation de l'un des événements suivants :

- Un incident de l'instance de base de données du dispositif d'écriture ou de lecture
- Un échec d'un réplica Aurora
- Une défaillance disque
- Une surcharge disque

Quand une requête d'injection d'erreurs spécifie un incident, elle provoque un incident de l'instance de base de données Aurora MySQL. Les autres requêtes d'injection d'erreurs se traduisent par des simulations d'événements d'erreur, mais n'entraînent pas la manifestation de l'événement. Lorsque vous soumettez une requête d'injection d'erreurs, vous pouvez aussi spécifier la durée pendant laquelle la simulation de l'événement d'erreur peut se produire.

Vous pouvez soumettre une requête d'injection d'erreurs à l'une de vos instances de réplica Aurora en vous connectant au point de terminaison du réplica Aurora. Pour plus d'informations, consultez Connexions de point de terminaison Amazon Aurora.

L'exécution de requêtes d'injection d'erreurs nécessite tous les privilèges d'utilisateur principal. Pour plus d'informations, consultez Privilèges du compte utilisateur principal.

Test d'un incident d'instance

Vous pouvez forcer un incident d'instance Amazon Aurora à l'aide de la requête d'injection d'erreurs `ALTER SYSTEM CRASH`.

Pour cette requête d'injection d'erreurs, un basculement ne se produira pas. Si vous souhaitez tester un basculement, vous pouvez choisir l'action d'instance Failover (Basculement) pour votre cluster de bases de données dans la console RDS, ou utiliser la commande de l'AWS CLI failover-db-cluster ou l'opération d'API RDS FailoverDBCluster.

Syntaxe

```
ALTER SYSTEM CRASH [ INSTANCE | DISPATCHER | NODE ];
```

Options

Cette requête accepte l'un des types d'incident suivants :

- **`INSTANCE`** — Simulation d'un incident de la base de données compatible MySQL pour l'instance Amazon Aurora.
- **`DISPATCHER`** — Simulation d'un incident lié au répartiteur sur l'instance de scripteur pour le cluster de bases de données Aurora. Le répartiteur écrit les mises à jour sur le volume de cluster d'un cluster de bases de données Amazon Aurora.
- **`NODE`** — Simulation d'un incident de la base de données compatible MySQL et du répartiteur pour l'instance Amazon Aurora. Pour cette simulation d'injection d'erreurs, le cache est également supprimé.

Le type d'incident par défaut est `INSTANCE`.

Test d'une défaillance d'un réplica Aurora

Vous pouvez simuler l'échec d'un réplica Aurora à l'aide de la fonction de requête d'injection d'erreurs `ALTER SYSTEM SIMULATE READ REPLICA FAILURE`.

L'échec d'un réplica Aurora bloque toutes les demandes provenant de l'instance d'enregistreur et adressées à un réplica Aurora ou à tous les réplicas Aurora du cluster de bases de données pendant un intervalle de temps spécifié. Une fois l'intervalle écoulé, les réplicas Aurora affectés sont automatiquement synchronisés avec l'instance d'enregistreur.

Syntaxe

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT READ REPLICA FAILURE
    [ TO ALL | TO "replica name" ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Options

La requête d'injection d'erreurs accepte les paramètres suivants :

- **`percentage_of_failure`** — Pourcentage de demandes de blocage pendant l'événement d'échec. La valeur peut être un nombre double compris entre 0 et 100. Si vous spécifiez 0, aucune demande n'est bloquée. Si vous spécifiez 100, toutes les demandes sont bloquées.

- Failure type (Type d'échec) — Type d'échec à simuler. Spécifiez `TO ALL` pour simuler des échecs pour tous les réplicas Aurora dans le cluster de bases de données. Spécifiez `TO` et le nom du réplica Aurora pour simuler l'échec d'un réplica Aurora unique. Le type d'incident par défaut est `TO ALL`.
- **`quantity`** — Durée pendant laquelle simuler l'échec du réplica Aurora. L'intervalle est une durée suivie d'une unité de temps. La simulation intervient pendant la durée spécifiée par l'unité. Par exemple, `20 MINUTE` entraîne l'exécution de la simulation pendant 20 minutes.

> **Note**
>
> Soyez vigilant lorsque vous spécifiez l'intervalle de l'événement d'erreur du réplica Aurora. Si vous spécifiez un intervalle trop long et que votre instance d'enregistreur écrit une importante quantité de données pendant l'échec, votre cluster Aurora DB peut considérer que votre réplica Aurora s'est bloqué et le remplacer.

Test d'une défaillance disque

Vous pouvez simuler l'échec d'un disque pour un cluster de bases de données Aurora à l'aide de la requête d'injection d'erreurs `ALTER SYSTEM SIMULATE DISK FAILURE`.

Pendant la simulation d'un échec du disque, le cluster de bases de données Aurora marque de façon aléatoire des segments de disque comme défectueux. Les demandes adressées à ces segments seront bloquées pendant la durée de la simulation.

Syntaxe

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK FAILURE
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Options

La requête d'injection d'erreurs accepte les paramètres suivants :

- **`percentage_of_failure`** — Pourcentage du disque à marquer comme défaillant pendant l'événement d'échec. La valeur peut être un nombre double compris entre 0 et 100. Si vous

spécifiez 0, aucune partie du disque n'est marquée comme défaillante. Si vous spécifiez 100, la totalité du disque est marquée comme défaillante.

- **`DISK index`** — Bloc de données logique spécifique pour lequel simuler l'événement d'échec. Si vous dépassez la plage de blocs de données logiques disponibles, vous recevez une erreur qui vous indique la valeur d'index maximale que vous pouvez spécifier. Pour plus d'informations, consultez Affichage du statut du volume pour un cluster de base de données Aurora MySQL.
- **`NODE index`** — Nœud de stockage spécifique pour lequel simuler l'événement d'échec. Si vous dépassez la plage de nœuds de stockage disponibles, vous recevez une erreur qui vous indique la valeur d'index maximale que vous pouvez spécifier. Pour plus d'informations, consultez Affichage du statut du volume pour un cluster de base de données Aurora MySQL.
- **`quantity`** — Durée pendant laquelle l'échec de disque est simulé. L'intervalle est une durée suivie d'une unité de temps. La simulation intervient pendant la durée spécifiée par l'unité. Par exemple, `20 MINUTE` entraîne l'exécution de la simulation pendant 20 minutes.

Test d'une surcharge disque

Vous pouvez simuler l'échec d'un disque pour un cluster de bases de données Aurora à l'aide de la requête d'injection d'erreurs `ALTER SYSTEM SIMULATE DISK CONGESTION`.

Pendant la simulation d'une surcharge du disque, le cluster de bases de données Aurora marque de façon aléatoire les segments disque comme surchargés. Les demandes adressées à ces segments sont retardées entre le délai minimal et le délai maximal spécifiés de la durée de la simulation.

Syntaxe

```
ALTER SYSTEM SIMULATE percentage_of_failure PERCENT DISK CONGESTION
    BETWEEN minimum AND maximum MILLISECONDS
    [ IN DISK index | NODE index ]
    FOR INTERVAL quantity { YEAR | QUARTER | MONTH | WEEK | DAY | HOUR | MINUTE |
 SECOND };
```

Options

La requête d'injection d'erreurs accepte les paramètres suivants :

- **`percentage_of_failure`** — Pourcentage du disque à marquer comme surchargé pendant l'événement d'échec. La valeur peut être un nombre double compris entre 0 et 100. Si vous

spécifiez 0, aucune partie du disque n'est marquée comme surchargée. Si vous spécifiez 100, la totalité du disque est marquée comme surchargée.

- **`DISK index`** ou **`NODE index`** — Disque ou nœud spécifique pour lequel l'événement d'échec est simulé. Si vous dépassez la plage d'index du disque ou du nœud, vous recevez une erreur qui vous indique la valeur d'index maximale que vous pouvez spécifier.
- **`minimum`** et **`maximum`** — Durées minimale et maximale du délai de surcharge, en millisecondes. Les segments de disque marqués comme surchargés sont retardés pendant une durée aléatoire comprise entre la durée minimale et la durée maximale en millisecondes de la simulation.
- **`quantity`** — Durée pendant laquelle la surcharge du disque est simulée. L'intervalle est une durée suivie d'une unité de temps. La simulation intervient pendant la durée spécifiée par l'unité de temps. Par exemple, `20 MINUTE` entraîne l'exécution de la simulation pendant 20 minutes.

Modification de tables dans Amazon Aurora à l'aide de Fast DDL

Amazon Aurora inclut des optimisations pour exécuter une opération `ALTER TABLE` en place, presque instantanément. L'opération s'effectue sans nécessiter la copie de la table et sans impact matériel sur les autres instructions DML. Puisque l'opération ne consomme pas de stockage temporaire pour une copie de table, les instructions DDL sont pratiques même pour des tables volumineuses sur des classes d'instance Small.

Aurora MySQL version 3 est compatible avec la fonction MySQL 8.0 appelée Instant DDL. Aurora MySQL version 2 utilise une implémentation différente appelée Fast DDL.

Rubriques

- Instant DDL (Aurora MySQL version 3)
- Fast DDL (Aurora MySQL version 2)

Instant DDL (Aurora MySQL version 3)

L'optimisation effectuée par Aurora MySQL version 3 pour améliorer l'efficacité de certaines opérations DDL est appelée DDL Instant DDL.

Aurora MySQL version 3 est compatible avec la fonction Instant DDL de MySQL 8.0 version communautaire. Vous effectuez une opération Instant DDL à l'aide de la clause `ALGORITHM=INSTANT` avec l'instruction `ALTER TABLE`. Pour plus de détails sur la syntaxe et

l'utilisation d'Instant DDL, consultez ALTER TABLE et Online DDL Operations dans la documentation MySQL.

Les exemples suivants illustrent la fonction Instant DDL. Les instructions `ALTER TABLE` ajoutent des colonnes et modifient les valeurs par défaut des colonnes. Les exemples incluent des colonnes régulières et virtuelles, ainsi que des tables régulières et partitionnées. À chaque étape, vous pouvez consulter les résultats en émettant les instructions `SHOW CREATE TABLE` et `DESCRIBE`.

```
mysql> CREATE TABLE t1 (a INT, b INT, KEY(b)) PARTITION BY KEY(b) PARTITIONS 6;
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE t1 RENAME TO t2, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ALTER COLUMN b SET DEFAULT 100, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE t2 ALTER COLUMN b DROP DEFAULT, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ADD COLUMN c ENUM('a', 'b', 'c'), ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 MODIFY COLUMN c ENUM('a', 'b', 'c', 'd', 'e'), ALGORITHM =
 INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> ALTER TABLE t2 ADD COLUMN (d INT GENERATED ALWAYS AS (a + 1) VIRTUAL), ALGORITHM
 = INSTANT;
Query OK, 0 rows affected (0.02 sec)

mysql> ALTER TABLE t2 ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t3 (a INT, b INT) PARTITION BY LIST(a)(
    ->   PARTITION mypart1 VALUES IN (1,3,5),
    ->   PARTITION MyPart2 VALUES IN (2,4,6)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> ALTER TABLE t3 ALTER COLUMN a SET DEFAULT 20, ALTER COLUMN b SET DEFAULT 200,
 ALGORITHM = INSTANT;
```

```
Query OK, 0 rows affected (0.01 sec)

mysql> CREATE TABLE t4 (a INT, b INT) PARTITION BY RANGE(a)
    ->   (PARTITION p0 VALUES LESS THAN(100), PARTITION p1 VALUES LESS THAN(1000),
    ->   PARTITION p2 VALUES LESS THAN MAXVALUE);
Query OK, 0 rows affected (0.05 sec)

mysql> ALTER TABLE t4 ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)

/* Sub-partitioning example */
mysql> CREATE TABLE ts (id INT, purchased DATE, a INT, b INT)
    ->   PARTITION BY RANGE( YEAR(purchased) )
    ->     SUBPARTITION BY HASH( TO_DAYS(purchased) )
    ->     SUBPARTITIONS 2 (
    ->       PARTITION p0 VALUES LESS THAN (1990),
    ->       PARTITION p1 VALUES LESS THAN (2000),
    ->       PARTITION p2 VALUES LESS THAN MAXVALUE
    ->    );
Query OK, 0 rows affected (0.10 sec)

mysql> ALTER TABLE ts ALTER COLUMN a SET DEFAULT 20,
    ->   ALTER COLUMN b SET DEFAULT 200, ALGORITHM = INSTANT;
Query OK, 0 rows affected (0.01 sec)
```

Fast DDL (Aurora MySQL version 2)

Fast DDL dans Aurora MySQL est une optimisation conçue pour améliorer les performances de certaines modifications de schéma, telles que l'ajout ou la suppression de colonnes, en réduisant la durée d'indisponibilité et l'utilisation des ressources. Elle permet d'effectuer ces opérations de manière plus efficace par rapport aux méthodes DDL traditionnelles.

> **Important**
>
> Pour l'instant, vous devez activer le mode lab d'Aurora pour pouvoir utiliser Fast DDL. Pour en savoir plus sur l'activation du mode lab, consultez Mode Lab Amazon Aurora MySQL. L'optimisation Fast DDL a été initialement introduite en mode lab sur Aurora MySQL version 2 pour améliorer l'efficacité de certaines opérations DDL. Dans la version 3 d'Aurora MySQL, le

mode lab a été abandonné et Fast DDL a été remplacé par la fonctionnalité Instant DDL de MySQL 8.0.

Dans MySQL, de nombreuses opérations de langage de définition de données (DDL) ont un impact important sur les performances.

Par exemple, supposons que vous utilisiez une opération `ALTER TABLE` pour ajouter une colonne à une table. En fonction de l'algorithme spécifié pour l'opération, cette dernière peut impliquer :

- la création d'une copie intégrale de la table,
- la création d'une table temporaire pour traiter les opérations DML (Data Manipulation Language) simultanées,
- la reconstruction de tous les index pour la table,
- l'application de verrous de table lors de l'application de modifications DML simultanées,
- le ralentissement du débit DML simultané.

Cet impact sur les performances peut être particulièrement difficile dans les environnements impliquant de grandes tables ou des volumes de transactions élevés. Fast DDL contribue à atténuer ces difficultés en optimisant les modifications de schéma, ce qui permet des opérations plus rapides et moins gourmandes en ressources.

Limitations FAST DDL

Fast DDL présente actuellement les limitations suivantes :

- FAST DLL prend uniquement en charge l'ajout de colonnes acceptant la valeur null, sans valeurs par défaut, à la fin d'une table existante.
- Fast DDL ne fonctionne pas pour les tables partitionnées.
- Fast DDL ne fonctionne pas pour des tables InnoDB qui utilisent le format de ligne REDUNDANT.
- Fast DDL ne fonctionne pas pour les tables avec des index de recherche en texte intégral.
- Si la taille maximale d'enregistrement possible pour l'opération DDL est trop importante, Fast DDL n'est pas utilisé. Une taille d'enregistrement est trop importante si elle est supérieure à la moitié de la taille de la page. La taille maximale d'un enregistrement est calculée en ajoutant les tailles maximales de toutes les colonnes. Pour les colonnes de taille variable, conformément aux normes InnoDB, les octets externes ne sont pas compris dans le calcul.

Syntaxe FAST DDL

```
ALTER TABLE tbl_name ADD COLUMN col_name column_definition
```

Cette instruction accepte les options suivantes :

- **`tbl_name`** — Nom de la table à modifier.
- **`col_name`** — Nom de la colonne à ajouter.
- **`col_definition`** — Définition de la colonne à ajouter.

> **Note**
>
> Vous devez spécifier une définition de colonne acceptant la valeur null sans valeur par défaut. Sinon, Fast DLL n'est pas utilisé.

Exemples FAST DDL

Les exemples suivants illustrent l'accélération due aux opérations Fast DDL. Le premier exemple SQL exécute des instructions `ALTER TABLE` sur une grande table sans utiliser Fast DDL. Cette opération prend beaucoup de temps. Un exemple d'interface CLI montre comment activer Fast DDL pour le cluster. Ensuite, un autre exemple SQL exécute les mêmes instructions `ALTER TABLE` sur une table identique. Avec Fast DDL activé, l'opération est très rapide.

Cet exemple utilise la table `ORDERS` du benchmark TPC-H, qui contient 150 millions de lignes. Ce cluster utilise volontairement une classe d'instance relativement petite afin de montrer combien de temps les instructions `ALTER TABLE` peuvent prendre lorsque vous ne pouvez pas utiliser Fast DDL. L'exemple crée un clone de la table d'origine contenant des données identiques. La vérification du paramètre `aurora_lab_mode` confirme que le cluster ne peut pas utiliser Fast DDL, car le mode Lab n'est pas activé. Ensuite, les instructions `ALTER TABLE ADD COLUMN` prennent beaucoup de temps pour ajouter des colonnes à la fin de la table.

```
mysql> create table orders_regular_ddl like orders;
Query OK, 0 rows affected (0.06 sec)

mysql> insert into orders_regular_ddl select * from orders;
Query OK, 150000000 rows affected (1 hour 1 min 25.46 sec)

mysql> select @@aurora_lab_mode;
```

```
+-------------------+
| @@aurora_lab_mode |
+-------------------+
|                 0 |
+-------------------+

mysql> ALTER TABLE orders_regular_ddl ADD COLUMN o_refunded boolean;
Query OK, 0 rows affected (40 min 31.41 sec)

mysql> ALTER TABLE orders_regular_ddl ADD COLUMN o_coverletter varchar(512);
Query OK, 0 rows affected (40 min 44.45 sec)
```

Cet exemple effectue la même préparation d'une grande table que l'exemple précédent. Cependant, vous ne pouvez pas simplement activer le mode Lab dans une séance SQL interactive. Ce paramètre doit être activé dans un groupe de paramètres personnalisé. Pour ce faire, il faut sortir de la session `mysql` et exécuter quelques commandes de la CLI AWS ou utiliser la AWS Management Console.

```
mysql> create table orders_fast_ddl like orders;
Query OK, 0 rows affected (0.02 sec)

mysql> insert into orders_fast_ddl select * from orders;
Query OK, 150000000 rows affected (58 min 3.25 sec)

mysql> set aurora_lab_mode=1;
ERROR 1238 (HY000): Variable 'aurora_lab_mode' is a read only variable
```

L'activation du mode Lab pour le cluster nécessite d'utiliser un groupe de paramètres. Cet exemple d'AWS CLI utilise un groupe de paramètres de cluster afin de garantir que toutes les instances de base de données dans le cluster utilisent la même valeur pour le paramètre de mode Lab.

```
$ aws rds create-db-cluster-parameter-group \
  --db-parameter-group-family aurora5.7 \
    --db-cluster-parameter-group-name lab-mode-enabled-57 --description 'TBD'
$ aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --query '*[*].[ParameterName,ParameterValue]' \
      --output text | grep aurora_lab_mode
aurora_lab_mode 0
$ aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --parameters ParameterName=aurora_lab_mode,ParameterValue=1,ApplyMethod=pending-
reboot
```

```
{
    "DBClusterParameterGroupName": "lab-mode-enabled-57"
}

# Assign the custom parameter group to the cluster that's going to use Fast DDL.
$ aws rds modify-db-cluster --db-cluster-identifier tpch100g \
  --db-cluster-parameter-group-name lab-mode-enabled-57
{
  "DBClusterIdentifier": "tpch100g",
  "DBClusterParameterGroup": "lab-mode-enabled-57",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.10.2",
  "Status": "available"
}

# Reboot the primary instance for the cluster tpch100g:
$ aws rds reboot-db-instance --db-instance-identifier instance-2020-12-22-5208
{
  "DBInstanceIdentifier": "instance-2020-12-22-5208",
  "DBInstanceStatus": "rebooting"
}

$ aws rds describe-db-clusters --db-cluster-identifier tpch100g \
  --query '*[].[DBClusterParameterGroup]' --output text
lab-mode-enabled-57

$ aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name lab-mode-enabled-57 \
    --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
      --output text | grep aurora_lab_mode
aurora_lab_mode 1
```

L'exemple suivant montre les étapes restantes une fois que les modifications du groupe de paramètres ont pris effet. Il teste le paramètre `aurora_lab_mode` pour s'assurer que le cluster peut utiliser Fast DDL. Ensuite, il exécute des instructions `ALTER TABLE` pour ajouter des colonnes à la fin d'une autre grande table. Cette fois, les instructions se terminent très rapidement.

```
mysql> select @@aurora_lab_mode;
+-------------------+
| @@aurora_lab_mode |
+-------------------+
|                 1 |
+-------------------+
```

```
mysql> ALTER TABLE orders_fast_ddl ADD COLUMN o_refunded boolean;
Query OK, 0 rows affected (1.51 sec)

mysql> ALTER TABLE orders_fast_ddl ADD COLUMN o_coverletter varchar(512);
Query OK, 0 rows affected (0.40 sec)
```

Affichage du statut du volume pour un cluster de base de données Aurora MySQL

Dans Amazon Aurora, un volume de cluster de base de données se compose d'un ensemble de blocs logiques. Chacun d'eux représente 10 gigaoctets de stockage alloué. Ces blocs sont appelés groupes de protection.

Les données figurant dans chaque groupe de protection sont répliquées sur six périphériques de stockage physiques, appelés nœuds de stockage. Ces nœuds de stockage sont alloués dans trois zones de disponibilité dans la région AWS où se trouve le cluster de base de données. Chaque nœud de stockage contient à son tour un ou plusieurs blocs de données logiques pour le volume de cluster de base de données. Pour plus d'informations sur les groupes de protection et les nœuds de stockage, consultez Introducing the Aurora Storage Engine sur le blog AWS Database.

Vous pouvez simuler la panne d'un nœud de stockage entier ou d'un seul bloc de données logique au sein d'un nœud de stockage. Pour ce faire, utilisez l'instruction d'injection d'erreurs `ALTER SYSTEM SIMULATE DISK FAILURE`. Pour l'instruction, vous devez spécifier la valeur d'index d'un nœud de stockage ou d'un bloc de données logique spécifique. Toutefois, si vous spécifiez une valeur d'index supérieure au nombre de nœuds de stockage ou de blocs de données logiques utilisés par le volume de cluster de base de données, l'instruction renvoie une erreur. Pour en savoir plus sur les requêtes d'injection d'erreurs, consultez Test d'Amazon Aurora MySQL à l'aide de requêtes d'injection d'erreurs.

Vous pouvez éviter cette erreur en utilisant l'instruction `SHOW VOLUME STATUS`. L'instruction renvoie deux variables de statut de serveur, `Disks` et `Nodes`. Ces variables représentent respectivement le nombre total de blocs de données logiques et de nœuds de stockage pour le volume de cluster de base de données.

Syntaxe

```
SHOW VOLUME STATUS
```

exemple

L'exemple suivant illustre un résultat SHOW VOLUME STATUS classique.

```
mysql> SHOW VOLUME STATUS;
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Disks         | 96    |
| Nodes         | 74    |
+---------------+-------+
```

Réglage d'Aurora MySQL

Les événements d'attente et les états de thread constituent des outils de réglage importants pour Aurora MySQL. Si vous parvenez à déterminer pourquoi les sessions sont en attente de ressources et ce qu'elles font, vous serez plus à même de réduire les goulots d'étranglement. Vous pouvez utiliser les informations de cette section pour déterminer les causes possibles et les actions correctives à mettre en œuvre.

Amazon DevOps Guru pour RDS peut déterminer de manière proactive si vos bases de données Aurora MySQL rencontrent des problèmes susceptibles de provoquer des problèmes plus graves ultérieurement. Amazon DevOps Guru pour RDS publie une explication et des recommandations concernant les actions correctives dans un insight proactif. Cette section contient des insights sur les problèmes courants.

> **Important**
>
> Les événements d'attente et les états de thread présentés dans cette section sont spécifiques à Aurora MySQL. Utilisez les informations de cette section pour régler uniquement Amazon Aurora, et non Amazon RDS for MySQL.
> Certains événements d'attente mentionnés dans cette section n'ont pas leur équivalent dans les versions open source de ces moteurs de base de données. D'autres événements d'attente portent le même nom que des événements des moteurs open source, mais se comportent différemment. Par exemple, le stockage Amazon Aurora fonctionne différemment du stockage open source et dès lors, les événements d'attente liés au stockage indiquent des conditions de ressources différentes.

Rubriques

Concepts essentiels à connaître pour le réglage d'Aurora MySQL

Avant de procéder au réglage de votre base de données Aurora MySQL, vous devez savoir ce que sont les événements d'attente ainsi que les états de thread, et pourquoi ils se produisent. Examinez

également l'architecture de base d'Aurora MySQL en termes de mémoire et de disque lorsque vous utilisez le moteur de stockage InnoDB. Un diagramme d'architecture très utile est disponible dans le Manuel de référence MySQL.

Rubriques

Événements d'attente Aurora MySQL

Un événement d'attente désigne une ressource pour laquelle une session est en attente. Par exemple, l'événement d'attente `io/socket/sql/client_connection` indique qu'un thread gère une nouvelle connexion. Les ressources généralement attendues par une session sont les suivantes :

- Accès monothread à une mémoire tampon, par exemple lorsqu'une session tente de modifier une mémoire tampon
- Ligne verrouillée par une autre session
- Lecture d'un fichier de données
- Écriture de fichier journal

Par exemple, pour répondre à une requête, la session peut effectuer une analyse complète de la table. Si les données ne sont pas déjà en mémoire, la session attend la fin des opérations d'I/O disque. Lorsque les mémoires tampons sont lues en mémoire, la session peut être contrainte d'attendre parce que d'autres sessions accèdent aux mêmes mémoires tampons. La base de données enregistre les attentes à l'aide d'un événement d'attente prédéfini. Ces événements sont regroupés en catégories.

En soi, un événement d'attente n'indique pas un problème de performances. Par exemple, si les données demandées ne sont pas en mémoire, il est nécessaire de les lire sur le disque. Si une session verrouille une ligne pour une mise à jour, une autre session attend que la ligne soit déverrouillée pour pouvoir la mettre à jour. Une validation nécessite d'attendre la fin de l'écriture dans un fichier journal. Les attentes font partie intégrante du fonctionnement normal d'une base de données.

Un grand nombre d'événements d'attente indique généralement un problème de performances. Dans ce cas, vous pouvez utiliser les données des événements d'attente pour déterminer où les sessions passent du temps. Par exemple, si plusieurs heures sont désormais nécessaires à l'exécution d'un rapport qui ne prend habituellement que quelques minutes, vous pouvez identifier les événements d'attente qui contribuent le plus au temps d'attente total. La détermination des causes des principaux événements d'attente peut vous permettre d'apporter des modifications qui auront pour effet d'améliorer les performances. Par exemple, si votre session est en attente sur une ligne qui a été verrouillée par une autre session, vous pouvez mettre fin à la session à l'origine du verrouillage.

États de thread Aurora MySQL

Un état général de thread est une valeur `State` associée au traitement général des requêtes. Par exemple, l'état de thread `sending data` indique qu'un thread lit et filtre les lignes d'une requête afin de déterminer l'ensemble de résultats qui convient.

Vous pouvez utiliser les états de thread pour régler Aurora MySQL de la même manière que vous utilisez les événements d'attente. Par exemple, des occurrences fréquentes de `sending data` indiquent généralement qu'une requête n'utilise pas d'index. Pour plus d'informations sur les états de thread, consultezGeneral Thread States dans le Manuel de référence MySQL.

Lorsque vous utilisez Performance Insights, l'une des conditions suivantes est vraie :

- Le schéma de performance est activé : Aurora MySQL affiche les événements d'attente plutôt que l'état de thread.
- Le schéma de performance n'est pas activé : Aurora MySQL affiche l'état de thread.

Nous vous recommandons de configurer le schéma de performance pour une gestion automatique. Le schéma de performance fournit des informations supplémentaires et de meilleurs outils afin d'examiner les possibles problèmes de performances. Pour en savoir plus, consultez Présentation du schéma de performance pour Performance Insights sur Aurora MySQL.

Mémoire Aurora MySQL

Dans Aurora MySQL, les zones mémoire les plus importantes sont le groupe de mémoires tampons et la mémoire tampon de journal.

Rubriques

- Groupe de mémoires tampons

Groupe de mémoires tampons

Le groupe de mémoires tampons correspond à la zone de mémoire partagée dans laquelle Aurora MySQL met en cache les données de table et d'index. Les requêtes peuvent accéder aux données fréquemment utilisées directement depuis la mémoire, sans lecture à partir du disque.

Le groupe de mémoires tampons est structuré sous forme de liste liée de pages. Une page peut contenir plusieurs lignes. Aurora MySQL utilise un algorithme LRU (Last Recently Used) pour faire vieillir les pages du groupe.

Pour en savoir plus, veuillez consulter Buffer Pool dans le Manuel de référence MySQL.

Processus Aurora MySQL

Aurora MySQL utilise un modèle de processus très différent d'Aurora PostgreSQL.

Rubriques

- Serveur MySQL (mysqld)
- Threads
- Groupe de threads

Serveur MySQL (mysqld)

Le serveur MySQL est un processus de système d'exploitation unique nommé mysqld. Le serveur MySQL ne génère pas de processus supplémentaires. Ainsi, une base de données Aurora MySQL utilise mysqld pour effectuer la majeure partie de ses tâches.

Lorsque le serveur MySQL démarre, il écoute les connexions réseau des clients MySQL. Lorsqu'un client se connecte à la base de données, mysqld ouvre un thread.

Threads

Les threads du gestionnaire de connexion associent chaque connexion client à un thread dédié. Ce thread gère l'authentification, exécute des instructions et renvoie les résultats au client. Si nécessaire, le gestionnaire de connexion crée de nouveaux threads.

Le cache de threads correspond à l'ensemble de threads disponibles. Lorsqu'une connexion se termine, MySQL renvoie le thread dans le cache de threads si ce dernier n'est pas plein. La variable système `thread_cache_size` détermine la taille du cache de threads.

Groupe de threads

Le groupe de threads se compose de plusieurs groupes de threads. Chaque groupe gère un ensemble de connexions client. Lorsqu'un client se connecte à la base de données, le groupe de threads attribue les connexions aux groupes de threads de manière circulaire. Le groupe de threads sépare les connexions et les threads. Il n'existe aucune relation fixe entre les connexions et les threads exécutant les instructions reçues de ces connexions.

Réglage d'Aurora MySQL avec des événements d'attente

Le tableau suivant récapitule les événements d'attente Aurora MySQL indiquant le plus souvent des problèmes de performances. Les événements d'attente suivants sont un sous-ensemble de la liste présente dans Événements d’attente Aurora MySQL.

Événement d'attente	Description
cpu	Cet événement se produit lorsqu'un thread est actif dans le processeur ou attend le processeur.
io/aurora_redo_log_flush	Cet événement se produit lorsqu'une session écrit des données persistantes dans le stockage Aurora.
io/aurora_respond_to_client	Cet événement se produit lorsqu'un thread attend de renvoyer un ensemble de résultats à un client.
io/redo_log_flush	Cet événement se produit lorsqu'une session écrit des données persistantes dans le stockage Aurora.
io/socket/sql/client_connection	Cet événement se produit lorsqu'un thread gère une nouvelle connexion.
io/table/sql/handler	Cet événement se produit lorsque la tâche a été déléguée à un moteur de stockage.
synch/cond/innodb/row_lock_wait	Cet événement se produit lorsqu'une session a verrouillé une ligne pour une mise à jour et

Événement d'attente	Description
	qu'une autre session tente de mettre à jour cette même ligne.
synch/cond/innodb/row_lock_wait_cond	Cet événement se produit lorsqu'une session a verrouillé une ligne pour une mise à jour et qu'une autre session tente de mettre à jour cette même ligne.
synch/cond/sql/MDL_context::COND_wait_status	Cet événement se produit lorsque des threads sont en attente de verrouillage des métadonnées de table.
synch/mutex/innodb/aurora_lock_thread_slot_futex	Cet événement se produit lorsqu'une session a verrouillé une ligne pour une mise à jour et qu'une autre session tente de mettre à jour cette même ligne.
synch/mutex/innodb/buf_pool_mutex	Cet événement se produit lorsqu'un thread a acquis un verrouillage sur le groupe de mémoires tampons InnoDB afin d'accéder à une page en mémoire.
synch/mutex/innodb/fil_system_mutex	Cet événement se produit lorsqu'une session attend d'accéder à la mémoire cache de l'espace disque logique.
synch/mutex/innodb/trx_sys_mutex	Cet événement se produit lorsque l'activité de base de données est élevée et présente un grand nombre de transactions.
synch/sxlock/innodb/hash_table_locks	Cet événement se produit lorsque des pages introuvables dans le groupe de mémoires tampons doivent être lues à partir d'un fichier.
synch/mutex/innodb/temp_pool_manager_mutex	Cet événement se produit lorsqu'une session attend d'acquérir un mutex pour gérer le pool de tablespaces temporaires de session.

cpu

L'événement d'attente `cpu` se produit lorsqu'un thread est actif dans le processeur ou attend le processeur.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL versions 2 et 3

Contexte

Pour chaque vCPU, une connexion peut exécuter des tâches sur ce processeur. Dans certains cas, le nombre de connexions actives prêtes à être exécutées est supérieur au nombre de vCPU. Ce déséquilibre entraîne des connexions en attente de ressources de processeur. Si le nombre de connexions actives est constamment supérieur au nombre de vCPU, votre instance est confrontée à une contention de processeur. En cas de contention, l'événement d'attente `cpu` se produit.

> **Note**
>
> La métrique Performance Insights pour le processeur est `DBLoadCPU`. La valeur de `DBLoadCPU` peut différer de la valeur de la métrique CloudWatch `CPUUtilization`. Cette dernière métrique est collectée à partir de l'hyperviseur pour une instance de base de données.

Les métriques du système d'exploitation Performance Insights fournissent des informations détaillées sur l'utilisation du processeur. Par exemple, vous pouvez afficher les métriques suivantes :

- `os.cpuUtilization.nice.avg`

- `os.cpuUtilization.total.avg`
- `os.cpuUtilization.wait.avg`
- `os.cpuUtilization.idle.avg`

Performance Insights signale l'utilisation du processeur par le moteur de base de données sous la forme `os.cpuUtilization.nice.avg`.

Causes probables de l'augmentation du nombre d'événements d'attente

Lorsque cet événement se produit plus que la normale, indiquant un possible problème de performances, les causes types sont les suivantes :

- Requêtes analytiques
- Transactions hautement simultanées
- Transactions de longue durée
- Augmentation soudaine du nombre de connexions (tempête de connexions)
- Augmentation du changement de contexte

Actions

Si l'événement d'attente `cpu` domine l'activité de la base de données, cela n'indique pas nécessairement un problème de performance. Ne réagissez à cet événement qu'en cas de dégradation des performances.

Selon la cause de l'augmentation de l'utilisation du processeur, envisagez les stratégies suivantes :

- Augmentez la capacité du processeur de l'hôte. En règle générale, une telle approche n'apporte qu'un soulagement temporaire.
- Identifiez les requêtes les plus importantes à des fins d'optimisation potentielle.
- Redirigez certaines charges de travail en lecture seule vers les nœuds de lecture, le cas échéant.

Rubriques

- Identifiez les sessions ou les requêtes à l'origine du problème.
- Analyser et optimiser une charge de travail élevée de l'UC

Identifiez les sessions ou les requêtes à l'origine du problème.

Pour trouver les sessions et les requêtes, consultez la table Top SQL (Principaux éléments SQL) dans Performance Insights et découvrez les instructions SQL dotées de la charge d'UC la plus élevée. Pour de plus amples informations, consultez Analyse des métriques à l'aide du tableau de bord de Performance Insights.

En règle générale, une ou deux instructions SQL consomment la majorité des cycles d'UC. Concentrez vos efforts sur ces instructions. Supposons que votre instance de base de données dispose de 2 vCPU avec une charge de base de données de 3,1 sessions actives en moyenne (AAS), le tout dans l'état d'UC. Dans ce cas, votre instance est liée à l'UC. Envisagez les stratégies suivantes :

- Procédez à un niveau vers une plus grande classe d'instance avec plus de vCPU.
- Réglez vos requêtes pour réduire la charge de l'UC.

Dans cet exemple, les principales requêtes SQL présentent une charge de base de données de 1,5 AAS, toutes à l'état d'UC. Une autre instruction SQL présente une charge de 0,1 à l'état d'UC. Dans cet exemple, si vous arrêtez l'instruction SQL de la charge la plus faible, vous ne réduisez pas la charge de la base de données de manière significative. Toutefois, si vous optimisez les deux requêtes à charge élevée pour qu'elles soient deux fois plus efficaces, vous éliminez le goulet d'étranglement de l'UC. Si vous réduisez la charge de l'UC de 1,5 AAS à hauteur de 50 %, l'AAS de chaque instruction diminue de 0,75. La charge de base de données totale dépensée sur l'UC est désormais de 1,6 AAS. Cette valeur est inférieure à la ligne de vCPU maximale de 2.0.

Pour une présentation de la résolution des problèmes à l'aide de Performance Insights, consultez le billet de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights. Consultez également l'AWSarticle de supportHow can I troubleshoot and resolve high CPU utilization on my Amazon RDS for MySQL instances?.

Analyser et optimiser une charge de travail élevée de l'UC

Après avoir identifié la ou les requêtes augmentant l'utilisation de l'UC, vous pouvez les optimiser ou mettre fin à la connexion. L'exemple suivant montre comment mettre fin à une connexion.

```
CALL mysql.rds_kill(processID);
```

Pour de plus amples informations, consultez mysql.rds_kill.

Si vous mettez fin à une session, l'action peut déclencher une longue restauration.

Suivre les recommandations relatives à l'optimisation des requêtes

Pour optimiser les requêtes, suivez les recommandations ci-dessous :

- Exécutez l'instruction `EXPLAIN`.

  Cette commande affiche les étapes individuelles associées à l'exécution d'une requête. Pour en savoir plus, consultez Optimizing Queries with EXPLAIN dans la documentation MySQL.
- Exécutez l'instruction `SHOW PROFILE`.

  Utilisez cette instruction pour consulter les détails du profil pouvant indiquer l'utilisation des ressources pour les instructions exécutées lors de la session en cours. Pour en savoir plus, consultez SHOW PROFILE Statement dans la documentation MySQL.
- Exécutez l'instruction `ANALYZE TABLE`.

  Utilisez cette instruction pour actualiser les statistiques d'index des tables accessibles par la requête sollicitant considérablement l'UC. En analysant l'instruction, vous pouvez aider l'optimiseur à choisir un plan d'exécution approprié. Pour en savoir plus, consultez ANALYZE TABLE Statement dans la documentation MySQL.

Suivez les recommandations pour améliorer l'utilisation de l'UC

Pour améliorer l'utilisation de l'UC dans une instance de base de données, suivez ces recommandations :

- Assurez-vous que toutes les requêtes utilisent des index appropriés.
- Voyez si vous pouvez utiliser des requêtes parallèles Aurora. Vous pouvez utiliser cette technique pour réduire l'utilisation de l'UC au niveau du nœud principal grâce au traitement de la fonction « pushdown », au filtrage des lignes et à la projection des colonnes pour la clause `WHERE`.
- Déterminez si le nombre d'exécutions SQL par seconde atteint les seuils attendus.
- Déterminez si la maintenance de l'index ou la création d'un nouvel index prend en charge les cycles d'UC nécessaires à votre charge de travail de production. Planifiez les activités de maintenance en dehors des heures de pointe.
- Déterminez si vous pouvez utiliser le partitionnement pour réduire l'ensemble de données de requête. Pour plus d'informations, consultez le billet de blog.How to plan and optimize Amazon Aurora with MySQL compatibility for consolidated workloads.

Vérifier la présence de tempêtes de connexions

Si la métrique `DBLoadCPU` n'est pas très élevée, mais que la métrique `CPUUtilization` l'est, la cause de l'utilisation élevée de l'UC se situe en dehors du moteur de base de données. La tempête de connexions en est l'illustration type.

Vérifiez si les conditions suivantes sont vraies :

- La métrique `CPUUtilization` Performance Insights et la métrique `DatabaseConnections` Amazon CloudWatch présente toutes deux une augmentation.
- Le nombre de threads de l'UC est supérieur au nombre de vCPU.

Si les conditions précédentes sont vraies, envisagez de diminuer le nombre de connexions à la base de données. Par exemple, vous pouvez utiliser un groupe de connexions tel que RDS Proxy. Pour connaître les bonnes pratiques en matière de gestion et de mise à l'échelle efficaces des connexions, consultez le livre blanc Manuel d'administrateur de base de données Amazon Aurora MySQL pour la gestion des connexions.

io/aurora_redo_log_flush

L’événement `io/aurora_redo_log_flush` se produit lorsqu’une session écrit des données persistantes sur un stockage Amazon Aurora.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l’augmentation du nombre d’événements d’attente
- Actions

Versions de moteur prises en charge

Ces informations relatives aux événements d’attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL version 2

Contexte

L'événement `io/aurora_redo_log_flush` est destiné à une opération d'entrée/sortie (E/S) dans Aurora MySQL.

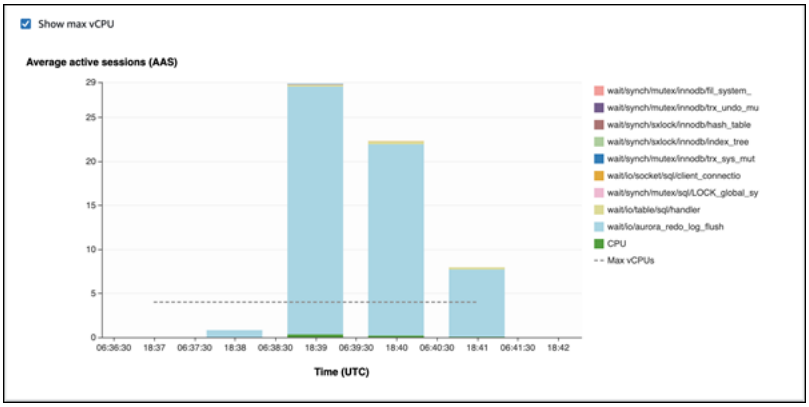
> **Note**
>
> Dans Aurora MySQL version 3, cet événement d'attente s'appelle io/redo_log_flush.

Causes probables de l'augmentation du nombre d'événements d'attente

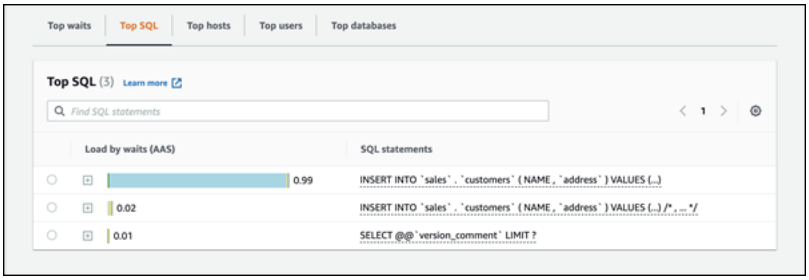
Pour assurer la persistance des données, les validations nécessitent une écriture durable dans un stockage stable. Si la base de données effectue trop de validations, un événement d'attente se produit lors de l'opération I/O en écriture, l'événement d'attente `io/aurora_redo_log_flush`.

Dans les exemples suivants, 50 000 enregistrements sont insérés dans un cluster de bases de données Aurora MySQL à l'aide de la classe d'instance de base de données db.r5.xlarge :

- Dans le premier exemple, chaque session insère 10 000 enregistrements ligne par ligne. Par défaut, en l'absence de commande de langage de manipulation de données (DML) dans une transaction, Aurora MySQL utilise des validations implicites. La validation automatique est activée. Cela signifie qu'une validation accompagne chaque insertion de ligne. Performance Insights montre que les connexions passent l'essentiel de leur temps à attendre sur l'événement d'attente `io/aurora_redo_log_flush`.

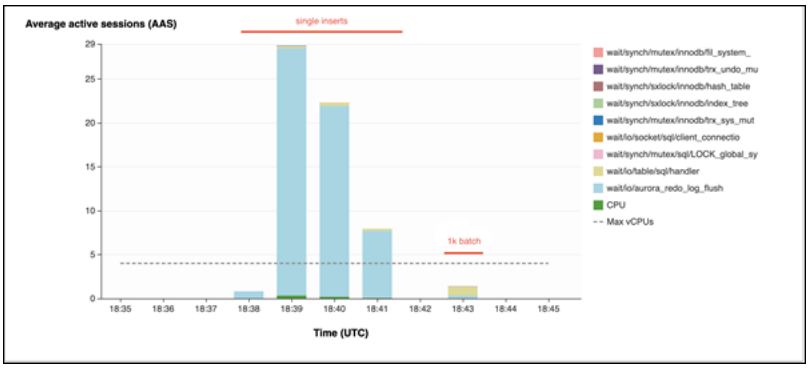


  Cela est dû aux simples instructions d'insertion utilisées.



L'insertion des 50 000 enregistrements requiert 3,5 minutes.

- Dans le deuxième exemple, les insertions sont réalisées par lots de 1 000 lots, ce qui signifie que chaque connexion effectue 10 validations au lieu de 10 000. Performance Insights montre que les connexions ne passent pas l'essentiel de leur temps à attendre sur l'événement d'attente `io/aurora_redo_log_flush`.



L'insertion des 50 000 enregistrements requiert 4 secondes.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Identifier les sessions et requêtes problématiques
- Regrouper vos opérations d'écriture
- Désactiver la validation automatique
- Utiliser des transactions
- Utiliser des lots

Identifier les sessions et requêtes problématiques

Si votre instance de base de données se heure à un goulet d'étranglement, votre première tâche consiste à rechercher les sessions et les requêtes qui en sont à l'origine. Pour un billet de blog AWS Database particulièrement utile, consultez Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Pour identifier les sessions et les requêtes à l'origine d'un goulet d'étranglement

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Performance Insights.
3. Sélectionnez votre instance DB.
4. Dans Database load (Charge de base de données), choisissez Slice by wait (Tranche par attente).
5. Au bas de la page, choisissez Top SQL (Principaux éléments SQL).

   Les requêtes situées en haut de la liste imposent la charge la plus élevée sur la base de données.

Regrouper vos opérations d'écriture

Les exemples suivants déclenchent l'événement d'attente `io/aurora_redo_log_flush`. (La validation automatique est activée.)

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

```
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

Pour réduire le temps passé à attendre sur l'événement d'attente `io/aurora_redo_log_flush`, regroupez logiquement vos opérations d'écriture dans une seule validation et limitez ainsi les appels persistants vers le stockage.

Désactiver la validation automatique

Désactivez la validation automatique avant d'effectuer d'importantes modifications en dehors d'une transaction, comme le montre l'exemple suivant.

```
SET SESSION AUTOCOMMIT=OFF;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
-- Other DML statements here
COMMIT;

SET SESSION AUTOCOMMIT=ON;
```

Utiliser des transactions

Vous pouvez utiliser des transactions comme le montre l'exemple suivant.

```
BEGIN
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;

-- Other DML statements here
```

```
END
```

Utiliser des lots

Vous pouvez apporter des modifications par lots, comme le montre l'exemple suivant. Cela étant, l'utilisation de lots trop volumineux peut entraîner des problèmes de performance, notamment dans les réplicas en lecture ou lors d'une reprise ponctuelle (PITR).

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES
('xxxx','xxxxx'),('xxxx','xxxxx'),...,('xxxx','xxxxx'),('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1 BETWEEN xx AND
 xxx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1<xx;
```

io/aurora_respond_to_client

L'événement `io/aurora_respond_to_client` se produit lorsqu'un thread attend de renvoyer un ensemble de résultats à un client.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL version 2

Contexte

L'événement `io/aurora_respond_to_client` indique qu'un thread attend de renvoyer un ensemble de résultats à un client.

Le traitement des requêtes est terminé et les résultats sont renvoyés au client de l'application. Toutefois, la bande passante réseau sur le cluster de bases de données étant insuffisante, un thread attend de renvoyer l'ensemble de résultats.

Causes probables de l'augmentation du nombre d'événements d'attente

Lorsque l'événement `io/aurora_respond_to_client` se produit plus souvent qu'à l'accoutumée, indiquant un possible problème de performance, les causes sont généralement les suivantes :

Classe d'instance de base de données insuffisante pour la charge de travail

La classe d'instance de base de données utilisée par le cluster de bases de données ne dispose pas de suffisamment de bande passante réseau pour traiter efficacement la charge de travail.

Ensembles de résultats volumineux

La taille de l'ensemble de résultats renvoyé a augmenté, car la requête renvoie un nombre plus élevé de lignes. L'ensemble de résultats plus volumineux consomme davantage de bande passante réseau.

Charge accrue sur le client

Le client peut être soumis à une pression exercée sur l'UC, la mémoire ou à une saturation du réseau. Une augmentation de la charge exercée sur le client retarde la réception des données du cluster de bases de données Aurora MySQL.

Latence réseau accrue

La latence réseau peut être accrue entre le cluster de bases de données Aurora MySQL et le client. Une latence réseau plus élevée augmente le temps nécessaire au client pour recevoir les données.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Identifier les sessions et les requêtes à l'origine des événements
- Mettre à l'échelle la classe d'instance de base de données
- Vérifier la charge de travail pour trouver des résultats inattendus

- Distribuer la charge de travail entre les instances de lecture
- Utiliser le modificateur SQL_BUFFER_RESULT

Identifier les sessions et les requêtes à l'origine des événements

Vous pouvez utiliser Performance Insights pour afficher les requêtes bloquées par l'événement d'attente `io/aurora_respond_to_client`. En règle générale, les bases de données à charge modérée à importante présentent des événements d'attente. Les événements d'attente peuvent être acceptables si les performances sont optimales. Si les performances ne sont pas optimales, voyez où la base de données passe le plus de temps. Examinez les événements d'attente qui contribuent à la charge la plus élevée et voyez si vous pouvez optimiser la base de données et l'application afin de réduire ces événements.

Pour rechercher les requêtes SQL responsables d'une charge élevée

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le volet de navigation, choisissez Performance Insights.
3. Choisissez une instance de base de données. Le tableau de bord Performance Insights s'affiche pour cette instance de base de données.
4. Dans le graphique Database load (Charge de base de données), choisissez Slice by wait (Tranche par attente).
5. Au bas de la page, choisissez Top SQL (Principaux éléments SQL).

   Le graphique répertorie les requêtes SQL responsables de la charge. Les requêtes situées en haut de la liste sont les plus responsables. Pour résoudre un goulet d'étranglement, concentrez-vous sur ces instructions.

Pour une présentation de la résolution des problèmes à l'aide de Performance Insights, consultez le billet de blog AWS Database Amazon Aurora MySQL Workloads with Performance Insights.

Mettre à l'échelle la classe d'instance de base de données

Vérifiez toute augmentation des métriques Amazon CloudWatch liées au débit réseau, notamment `NetworkReceiveThroughput` et `NetworkTransmitThroughput`. Si la bande passante réseau de la classe d'instance de base de données est atteinte, vous pouvez mettre à l'échelle la classe d'instance de base de données utilisée par le cluster de bases de données. Une classe d'instance

de base de données dotée d'une bande passante réseau plus importante renvoie les données aux clients plus efficacement.

Pour plus d'informations sur la surveillance des métriques Amazon CloudWatch, consultez Affichage des métriques dans la console Amazon RDS. Pour plus d'informations sur les classes d'instances de base de données, consultez Classes d'instance de base de données Amazon Aurora. Pour plus d'informations sur la modification d'un cluster de base de données , consultez Modification d'un cluster de bases de données Amazon Aurora.

Vérifier la charge de travail pour trouver des résultats inattendus

Vérifiez la charge de travail sur le cluster de bases de données et assurez-vous qu'elle ne produit pas de résultats inattendus. Par exemple, certaines requêtes peuvent renvoyer un nombre de lignes plus élevé que prévu. Si tel est le cas, vous pouvez utiliser des métriques de compteur Performance Insights telles que `Innodb_rows_read`. Pour de plus amples informations, consultez Métrique de compteur de Performance Insights.

Distribuer la charge de travail entre les instances de lecture

Vous pouvez distribuer la charge de travail en lecture seule entre les réplicas Aurora. Vous pouvez procéder à une mise à l'échelle horizontale en ajoutant d'autres réplicas Aurora. Cela peut entraîner une augmentation des limites de bande passante réseau. Pour de plus amples informations, consultez Clusters de bases de données Amazon Aurora.

Utiliser le modificateur SQL_BUFFER_RESULT

Vous pouvez ajouter le modificateur `SQL_BUFFER_RESULT` aux instructions `SELECT` pour forcer les résultats dans une table temporaire avant qu'ils ne soient renvoyés au client. Ce modificateur facilite la résolution des problèmes de performances lorsque les verrous InnoDB ne sont pas libérés, car des requêtes se trouvent dans l'état d'attente `io/aurora_respond_to_client`. Pour en savoir plus, consultez SELECT Statement dans la documentation MySQL.

io/redo_log_flush

L'événement `io/redo_log_flush` se produit lorsqu'une session écrit des données persistantes sur un stockage Amazon Aurora.

Rubriques

- Versions de moteur prises en charge
- Contexte

Versions de moteur prises en charge

Ces informations relatives aux événements d’attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL version 3

Contexte

L’événement `io/redo_log_flush` est destiné à une opération d’entrée/sortie (E/S) dans Aurora MySQL.

> **Note**
>
> Dans Aurora MySQL version 2, cet événement d’attente s’appelle io/aurora_redo_log_flush.

Causes probables de l’augmentation du nombre d’événements d’attente

Pour assurer la persistance des données, les validations nécessitent une écriture durable dans un stockage stable. Si la base de données effectue trop de validations, un événement d’attente se produit lors de l’opération I/O en écriture, l’événement d’attente `io/redo_log_flush`.

Pour des exemples du comportement de cet événement d’attente, consultez io/aurora_redo_log_flush.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d’attente.

Rubriques

- Utiliser des lots

Identifier les sessions et requêtes problématiques

Si votre instance de base de données se heure à un goulet d'étranglement, votre première tâche consiste à rechercher les sessions et les requêtes qui en sont à l'origine. Pour un billet de blog AWS Database particulièrement utile, consultez Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Pour identifier les sessions et les requêtes à l'origine d'un goulet d'étranglement

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Performance Insights.
3. Sélectionnez votre instance DB.
4. Dans Database load (Charge de base de données), choisissez Slice by wait (Tranche par attente).
5. Au bas de la page, choisissez Top SQL (Principaux éléments SQL).

   Les requêtes situées en haut de la liste imposent la charge la plus élevée sur la base de données.

Regrouper vos opérations d'écriture

Les exemples suivants déclenchent l'événement d'attente `io/redo_log_flush`. (La validation automatique est activée.)

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE id=xx;
```

```
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

Pour réduire le temps passé à attendre sur l'événement d'attente `io/redo_log_flush`, regroupez logiquement vos opérations d'écriture dans une seule validation et limitez ainsi les appels persistants vers le stockage.

Désactiver la validation automatique

Désactivez la validation automatique avant d'effectuer d'importantes modifications en dehors d'une transaction, comme le montre l'exemple suivant.

```
SET SESSION AUTOCOMMIT=OFF;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
....
UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1=xx;
-- Other DML statements here
COMMIT;

SET SESSION AUTOCOMMIT=ON;
```

Utiliser des transactions

Vous pouvez utiliser des transactions comme le montre l'exemple suivant.

```
BEGIN
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');
....
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES ('xxxx','xxxxx');

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
....
DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1=xx;
```

```
-- Other DML statements here
END
```

Utiliser des lots

Vous pouvez apporter des modifications par lots, comme le montre l'exemple suivant. Cela étant, l'utilisation de lots trop volumineux peut entraîner des problèmes de performance, notamment dans les réplicas en lecture ou lors d'une reprise ponctuelle (PITR).

```
INSERT INTO `sampleDB`.`sampleTable` (sampleCol2, sampleCol3) VALUES
('xxxx','xxxxx'),('xxxx','xxxxx'),...,('xxxx','xxxxx'),('xxxx','xxxxx');

UPDATE `sampleDB`.`sampleTable` SET sampleCol3='xxxxx' WHERE sampleCol1 BETWEEN xx AND
 xxx;

DELETE FROM `sampleDB`.`sampleTable` WHERE sampleCol1<xx;
```

io/socket/sql/client_connection

L'événement `io/socket/sql/client_connection` se produit lorsqu'un thread gère une nouvelle connexion.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL versions 2 et 3

Contexte

L'événement `io/socket/sql/client_connection` indique que mysqld est occupé à créer des threads pour gérer les nouvelles connexions client entrantes. Dans ce scénario, le traitement de la

maintenance des nouvelles demandes de connexion client ralentit alors que les connexions attendent l'attribution du thread. Pour de plus amples informations, consultez Serveur MySQL (mysqld).

Causes probables de l'augmentation du nombre d'événements d'attente

Lorsque cet événement se produit plus souvent qu'à l'accoutumée, indiquant un possible problème de performances, les causes sont généralement les suivantes :

- Le nombre de nouvelles connexions utilisateur entre l'application et votre instance Amazon RDS augmente soudainement.
- Votre instance de base de données n'est pas en mesure de traiter de nouvelles connexions, car le réseau, le processeur ou la mémoire sont limités.

Actions

Si `io/socket/sql/client_connection` domine l'activité de la base de données, cela n'indique pas nécessairement un problème de performances. Dans une base de données active, un événement d'attente est toujours en tête. N'intervenez qu'en cas de dégradation des performances. Nous recommandons différentes actions selon les causes de votre événement d'attente.

Rubriques

- Identifier les sessions et requêtes problématiques
- Suivre les bonnes pratiques relatives à la gestion des connexions
- Augmenter l'échelle de votre instance en cas de limitation des ressources
- Vérifier les hôtes et les utilisateurs principaux
- Interroger les tables performance_schema
- Vérifiez l'état des threads de vos requêtes
- Auditer vos demandes et requêtes
- Grouper vos connexions de base de données

Identifier les sessions et requêtes problématiques

Si votre instance de base de données se heure à un goulet d'étranglement, votre première tâche consiste à rechercher les sessions et les requêtes qui en sont à l'origine. Pour un billet de blog particulièrement utile, consultez Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Pour identifier les sessions et les requêtes à l'origine d'un goulet d'étranglement

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le volet de navigation, choisissez Performance Insights.
3. Sélectionnez votre instance DB.
4. Dans Database load (Charge de base de données), choisissez Slice by wait (Tranche par attente).
5. Au bas de la page, choisissez Top SQL (Principaux éléments SQL).

   Les requêtes situées en haut de la liste imposent la charge la plus élevée sur la base de données.

Suivre les bonnes pratiques relatives à la gestion des connexions

Pour gérer vos connexions, envisagez les stratégies suivantes :

- Utiliser le regroupement des connexions

  Vous pouvez augmenter progressivement le nombre de connexions au fil de vos besoins. Pour plus d'informations, consultez le livre blanc Manuel de l'administrateur de base de données Amazon Aurora MySQL.
- Utilisez un nœud de lecteur pour redistribuer le trafic en lecture seule.

  Pour plus d’informations, consultez Réplicas Aurora et Connexions de point de terminaison Amazon Aurora.

Augmenter l'échelle de votre instance en cas de limitation des ressources

Recherchez des exemples de limitation dans les ressources suivantes :

- CPU

  Vérifiez vos métriques Amazon CloudWatch en termes d'utilisation élevée de l'UC.
- Réseau

  Vérifiez toute augmentation des métriques CloudWatch `network receive throughput` et `network transmit throughput`. Si votre instance a atteint la limite de bande passante réseau

pour votre classe d'instance, pensez à augmenter l'échelle de votre instance RDS vers un type de classe d'instance supérieur. Pour de plus amples informations, consultez Classes d'instance de base de données Amazon Aurora.

- Mémoire libérable

  Vérifiez une éventuelle baisse de la métrique CloudWatch `FreeableMemory`. Pensez également à activer la surveillance améliorée. Pour de plus amples informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.

Vérifier les hôtes et les utilisateurs principaux

Utilisez Performance Insights pour vérifier les hôtes et les utilisateurs principaux. Pour de plus amples informations, consultez Analyse des métriques à l'aide du tableau de bord de Performance Insights.

Interroger les tables performance_schema

Pour obtenir un comptage précis des connexions actuelles et totales, interrogez les tables `performance_schema`. Cette technique vous permet d'identifier l'utilisateur ou l'hôte source responsable de la création d'un grand nombre de connexions. Par exemple, interrogez les tables `performance_schema` comme suit.

```
SELECT * FROM performance_schema.accounts;
SELECT * FROM performance_schema.users;
SELECT * FROM performance_schema.hosts;
```

Vérifiez l'état des threads de vos requêtes

Si votre problème de performances persiste, vérifiez l'état des threads de vos requêtes. Dans le client `mysql`, exécutez la commande suivante.

```
show processlist;
```

Auditer vos demandes et requêtes

Pour vérifier la nature des demandes et des requêtes provenant de comptes utilisateurs, utilisez Aurora MySQL Advanced Auditing. Pour en savoir plus sur l'activation de l'audit, consultez Utilisation de l'Audit avancé avec un cluster de bases de données Amazon Aurora MySQL.

Grouper vos connexions de base de données

Pensez à utiliser Amazon RDS Proxy pour gérer les connexions. RDS Proxy vous permet d'autoriser vos applications à grouper et à partager des connexions de bases de données pour améliorer leur capacité de mise à l'échelle. RDS Proxy rend les applications plus résistantes aux échecs de base de données en les connectant automatiquement à une instance de base de données de secours tout en préservant les connexions des applications. Pour de plus amples informations, consultez Proxy Amazon RDS pour Aurora.

io/table/sql/handler

L'événement `io/table/sql/handler` se produit lorsque la tâche a été déléguée à un moteur de stockage.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL versions 2 et 3

Contexte

L'événement `io/table` indique une attente avant d'accéder à une table. Cet événement se produit indépendamment du fait que les données soient mises en cache dans le groupe de mémoires tampons ou accessibles sur disque. L'événement `io/table/sql/handler` indique une augmentation de l'activité de la charge de travail.

Un gestionnaire est une routine spécialisée dans un certain type de données ou axée sur certaines tâches spécifiques. Par exemple, un gestionnaire d'événements reçoit et effectue la synthèse des événements et des signaux issus du système d'exploitation ou d'une interface utilisateur. Un gestionnaire de mémoire effectue des tâches liées à la mémoire. Un gestionnaire d'entrée de fichier

est une fonction qui reçoit l'entrée de fichier et effectue des tâches spécifiques sur les données, en fonction du contexte.

Des vues telles que `performance_schema.events_waits_current` indiquent souvent `io/table/sql/handler` lorsque l'attente réelle est un événement d'attente imbriqué tel qu'un verrou. Lorsque l'attente réelle n'est pas `io/table/sql/handler`, Performance Insights signale l'événement d'attente imbriqué. Lorsque Performance Insights signale `io/table/sql/handler`, il représente le traitement InnoDB de la requête d'I/O, et non un événement d'attente imbriqué masqué. Pour plus d'informations, consultez Performance Schema Atom and Molecule Events dans le Manuel de référence MySQL.

L'événement `io/table/sql/handler` apparaît souvent dans les principaux événements d'attente avec des attentes I/O telles que `io/aurora_redo_log_flush`.

Causes probables de l'augmentation du nombre d'événements d'attente

Dans Performance Insights, des pics soudains de l'événement `io/table/sql/handler` indiquent une augmentation de l'activité de la charge de travail. Une activité accrue traduit une augmentation des I/O.

Performance Insights filtre les ID d'événements imbriqués et ne signale pas d'attente `io/table/sql/handler` lorsque l'événement imbriqué sous-jacent correspond à une attente de verrouillage. Par exemple, si l'événement de cause racine est synch/mutex/innodb/aurora_lock_thread_slot_futex, Performance Insights affiche cette attente dans les principaux événements d'attente, et non `io/table/sql/handler`.

Dans des vues telles que `performance_schema.events_waits_current`, les attentes pour `io/table/sql/handler` apparaissent souvent lorsque l'attente réelle est un événement d'attente imbriqué tel qu'un verrou. Lorsque l'attente réelle diffère de `io/table/sql/handler`, Performance Insights recherche l'attente imbriquée et signale l'attente réelle plutôt que `io/table/sql/handler`. Lorsque Performance Insights signale `io/table/sql/handler`, l'attente réelle est `io/table/sql/handler`, et non un événement d'attente imbriqué masqué. Pour plus d'informations, consultez Performance Schema Atom and Molecule Events dans le Manuel de référence MySQL 5.7.

Actions

Si cet événement d'attente domine l'activité de la base de données, cela n'indique pas nécessairement un problème de performances. Un événement d'attente est toujours en tête lorsque la base de données est active. Vous ne devez intervenir qu'en cas de dégradation des performances.

Nous recommandons différentes actions en fonction des autres événement d'attente que vous voyez.

Rubriques

Identifier les sessions et les requêtes à l'origine des événements

En règle générale, les bases de données à charge modérée à importante présentent des événements d'attente. Les événements d'attente peuvent être acceptables si les performances sont optimales. Si les performances ne sont pas optimales, voyez où la base de données passe le plus de temps. Examinez les événements d'attente qui contribuent à la charge la plus élevée et voyez si vous pouvez optimiser la base de données et l'application afin de réduire ces événements.

Pour rechercher les requêtes SQL responsables d'une charge élevée

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Performance Insights.
3. Choisissez une instance de base de données. Le tableau de bord Performance Insights s'affiche pour cette instance de base de données.
4. Dans le graphique Database load (Charge de base de données), choisissez Slice by wait (Tranche par attente).
5. Au bas de la page, choisissez Top SQL (Principaux éléments SQL).

   Le graphique répertorie les requêtes SQL responsables de la charge. Les requêtes situées en haut de la liste sont les plus responsables. Pour résoudre un goulet d'étranglement, concentrez-vous sur ces instructions.

Pour une présentation de la résolution des problèmes à l'aide de Performance Insights, consultez le billet de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Vérifier une éventuelle corrélation avec les métriques de compteur Performance Insights

Vérifiez les métriques de compteur Performance Insights telles que `Innodb_rows_changed`. Si les métriques de compteur sont corrélées avec `io/table/sql/handler`, procédez comme suit :

1. Dans Performance Insights, recherchez les instructions SQL prenant en compte le principal événement d'attente `io/table/sql/handler`. Si possible, optimisez cette instruction de manière à ce qu'elle renvoie moins de lignes.
2. Récupérez les tables principales des vues `schema_table_statistics` et `x$schema_table_statistics`. Ces vues indiquent le temps passé par table. Pour plus d'informations, consultez The schema_table_statistics and x$schema_table_statistics Views dans le Manuel de référence MySQL.

   Par défaut, les lignes sont triées en fonction du temps d'attente total décroissant. Les tables présentant le plus de contention apparaissent en premier. La sortie indique si le temps concerne des lectures, écritures, extractions, insertions, mises à jour ou suppressions.

```
mysql> select * from sys.schema_table_statistics limit 1\G

*************************** 1. row ***************************
     table_schema: read_only_db
       table_name: sbtest41
    total_latency: 54.11 m
     rows_fetched: 6001557
    fetch_latency: 39.14 m
    rows_inserted: 14833
   insert_latency: 5.78 m
     rows_updated: 30470
   update_latency: 5.39 m
     rows_deleted: 14833
   delete_latency: 3.81 m
 io_read_requests: NULL
          io_read: NULL
  io_read_latency: NULL
io_write_requests: NULL
         io_write: NULL
 io_write_latency: NULL
 io_misc_requests: NULL
  io_misc_latency: NULL
1 row in set (0.11 sec)
```

Rechercher d'autres événements d'attente corrélés

Si `synch/sxlock/innodb/btr_search_latch` et `io/table/sql/handler` contribuent le plus à l'anomalie de charge de base de données, vérifiez si la variable

`innodb_adaptive_hash_index` est activée. Si tel est le cas, pensez à augmenter la valeur du paramètre `innodb_adaptive_hash_index_parts`.

Si l'index de hachage adaptatif est désactivé, pensez à l'activer. Pour en savoir plus sur l'index de hachage adaptatif MySQL, consultez les ressources suivantes :

- Article Is Adaptive Hash Index in InnoDB right for my workload? sur le site Percona
- Adaptive Hash Index dans leManuel de référence MySQL
- Article Contention in MySQL InnoDB: Useful Info From the Semaphores Section sur le site Percona

> **Note**
>
> L'index de hachage adaptatif n'est pas pris en charge par les instances de base de données du lecteur Aurora.
> Dans certains cas, les performances peuvent être médiocres sur une instance en lecture lorsque `synch/sxlock/innodb/btr_search_latch` et `io/table/sql/handler` sont dominants. Si tel est le cas, pensez à rediriger temporairement la charge de travail vers l'instance de base de données d'enregistreur et à activer l'index de hachage adaptatif.

synch/cond/innodb/row_lock_wait

L'événement `synch/cond/innodb/row_lock_wait` se produit lorsqu'une session a verrouillé une ligne pour une mise à jour et qu'une autre session tente de mettre à jour cette même ligne. Pour plus d'informations, consultez InnoDB Locking dans la documentation MySQL.

Versions de moteur prises en charge

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL version 3

Causes probables de l'augmentation du nombre d'événements d'attente

Plusieurs instructions en langage de manipulation de données (DML) accèdent simultanément aux mêmes lignes.

Actions

Nous recommandons différentes actions en fonction des autres événement d'attente que vous voyez.

Rubriques

- Rechercher et répondre aux instructions SQL responsables de cet événement d'attente
- Rechercher et répondre à la session de blocage

Rechercher et répondre aux instructions SQL responsables de cet événement d'attente

Utilisez Performance Insights pour identifier les instructions SQL responsables de cet événement d'attente. Envisagez les stratégies suivantes :

- Si les verrous de ligne posent un problème persistant, pensez à réécrire l'application de manière à utiliser un verrouillage optimiste.
- Utiliser plusieurs instructions
- Répartissez la charge de travail sur différents objets de base de données. Vous pouvez le faire via le partitionnement.
- Vérifiez la valeur du paramètre `innodb_lock_wait_timeout`. Il contrôle le délai d'attente des transactions avant de générer une erreur de dépassement.

Pour une présentation de la résolution des problèmes à l'aide de Performance Insights, consultez le billet de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Rechercher et répondre à la session de blocage

Déterminez si la session de blocage est active ou inactive. Vérifiez également si la session provient d'une application ou d'un utilisateur actif.

Pour identifier la session maintenant le verrou, vous pouvez exécuter `SHOW ENGINE INNODB STATUS`. L'exemple suivant illustre une sortie.

```
mysql> SHOW ENGINE INNODB STATUS;

---TRANSACTION 1688153, ACTIVE 82 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 2 row lock(s)
MySQL thread id 4244, OS thread handle 70369524330224, query id 4020834 172.31.14.179
 reinvent executing
```

```
select id1 from test.t1 where id1=1 for update
------- TRX HAS BEEN WAITING 24 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 11 page no 4 n bits 72 index GEN_CLUST_INDEX of table test.t1 trx
 id 1688153 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
```

Vous pouvez également utiliser la requête suivante pour extraire des détails sur les verrous en cours.

```
mysql> SELECT p1.id waiting_thread,
    p1.user waiting_user,
    p1.host waiting_host,
    it1.trx_query waiting_query,
    ilw.requesting_engine_transaction_id waiting_transaction,
    ilw.blocking_engine_lock_id blocking_lock,
    il.lock_mode blocking_mode,
    il.lock_type blocking_type,
    ilw.blocking_engine_transaction_id blocking_transaction,
    CASE it.trx_state
        WHEN 'LOCK WAIT'
        THEN it.trx_state
        ELSE p.state end blocker_state,
    concat(il.object_schema,'.', il.object_name) as locked_table,
    it.trx_mysql_thread_id blocker_thread,
    p.user blocker_user,
    p.host blocker_host
FROM performance_schema.data_lock_waits ilw
JOIN performance_schema.data_locks il
ON ilw.blocking_engine_lock_id = il.engine_lock_id
AND ilw.blocking_engine_transaction_id = il.engine_transaction_id
JOIN information_schema.innodb_trx it
ON ilw.blocking_engine_transaction_id = it.trx_id join information_schema.processlist p
ON it.trx_mysql_thread_id = p.id join information_schema.innodb_trx it1
ON ilw.requesting_engine_transaction_id = it1.trx_id join
 information_schema.processlist p1
ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
waiting_thread: 4244
waiting_user: reinvent
waiting_host: 123.456.789.012:18158
waiting_query: select id1 from test.t1 where id1=1 for update
waiting_transaction: 1688153
blocking_lock: 70369562074216:11:4:2:70369549808672
```

```
blocking_mode: X
blocking_type: RECORD
blocking_transaction: 1688142
blocker_state: User sleep
locked_table: test.t1
blocker_thread: 4243
blocker_user: reinvent
blocker_host: 123.456.789.012:18156
1 row in set (0.00 sec)
```

Après avoir identifié la session, vous pouvez procédez comme suit :

- Contactez le propriétaire de l'application ou l'utilisateur.
- Si la session de blocage est inactive, pensez à y mettre fin. Une telle action peut déclencher une longue restauration. Pour savoir comment mettre fin à une session, consultez Mettre fin à une session ou à une requête.

Pour plus d'informations sur l'identification des transactions de blocage, consultez Using InnoDB Transaction and Locking Information dans la documentation MySQL.

synch/cond/innodb/row_lock_wait_cond

L'événement `synch/cond/innodb/row_lock_wait_cond` se produit lorsqu'une session a verrouillé une ligne pour une mise à jour et qu'une autre session tente de mettre à jour cette même ligne. Pour plus d'informations, consultez InnoDB Locking dans la documentation MySQL.

Versions de moteur prises en charge

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL version 2

Causes probables de l'augmentation du nombre d'événements d'attente

Plusieurs instructions en langage de manipulation de données (DML) accèdent simultanément aux mêmes lignes.

Actions

Nous recommandons différentes actions en fonction des autres événement d'attente que vous voyez.

Rubriques

- Rechercher et répondre aux instructions SQL responsables de cet événement d'attente
- Rechercher et répondre à la session de blocage

Rechercher et répondre aux instructions SQL responsables de cet événement d'attente

Utilisez Performance Insights pour identifier les instructions SQL responsables de cet événement d'attente. Envisagez les stratégies suivantes :

- Si les verrous de ligne posent un problème persistant, pensez à réécrire l'application de manière à utiliser un verrouillage optimiste.
- Utiliser plusieurs instructions
- Répartissez la charge de travail sur différents objets de base de données. Vous pouvez le faire via le partitionnement.
- Vérifiez la valeur du paramètre `innodb_lock_wait_timeout`. Il contrôle le délai d'attente des transactions avant de générer une erreur de dépassement.

Pour une présentation de la résolution des problèmes à l'aide de Performance Insights, consultez le billet de blog Analyze Amazon Aurora MySQL Workloads with Performance Insights.

Rechercher et répondre à la session de blocage

Déterminez si la session de blocage est active ou inactive. Vérifiez également si la session provient d'une application ou d'un utilisateur actif.

Pour identifier la session maintenant le verrou, vous pouvez exécuter `SHOW ENGINE INNODB STATUS`. L'exemple suivant illustre une sortie.

```
mysql> SHOW ENGINE INNODB STATUS;

---TRANSACTION 2771110, ACTIVE 112 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 1136, 1 row lock(s)
MySQL thread id 24, OS thread handle 70369573642160, query id 13271336 172.31.14.179
 reinvent Sending data
```

```
select id1 from test.t1 where id1=1 for update
------- TRX HAS BEEN WAITING 43 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 11 page no 3 n bits 0 index GEN_CLUST_INDEX of table test.t1 trx
 id 2771110 lock_mode X waiting
Record lock, heap no 2 PHYSICAL RECORD: n_fields 5; compact format; info bits 0
```

Vous pouvez également utiliser la requête suivante pour extraire des détails sur les verrous en cours.

```
mysql> SELECT p1.id waiting_thread,
              p1.user waiting_user,
              p1.host waiting_host,
              it1.trx_query waiting_query,
              ilw.requesting_trx_id waiting_transaction,
              ilw.blocking_lock_id blocking_lock,
              il.lock_mode blocking_mode,
              il.lock_type blocking_type,
              ilw.blocking_trx_id blocking_transaction,
              CASE it.trx_state
                WHEN 'LOCK WAIT'
                THEN it.trx_state
                ELSE p.state
              END blocker_state,
              il.lock_table locked_table,
              it.trx_mysql_thread_id blocker_thread,
              p.user blocker_user,
              p.host blocker_host
       FROM information_schema.innodb_lock_waits ilw
       JOIN information_schema.innodb_locks il
         ON ilw.blocking_lock_id = il.lock_id
        AND ilw.blocking_trx_id = il.lock_trx_id
       JOIN information_schema.innodb_trx it
         ON ilw.blocking_trx_id = it.trx_id
       JOIN information_schema.processlist p
         ON it.trx_mysql_thread_id = p.id
       JOIN information_schema.innodb_trx it1
         ON ilw.requesting_trx_id = it1.trx_id
       JOIN information_schema.processlist p1
         ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
      waiting_thread: 3561959471
        waiting_user: reinvent
        waiting_host: 123.456.789.012:20485
```

```
       waiting_query: select id1 from test.t1 where id1=1 for update
 waiting_transaction: 312337314
       blocking_lock: 312337287:261:3:2
       blocking_mode: X
       blocking_type: RECORD
blocking_transaction: 312337287
       blocker_state: User sleep
        locked_table: `test`.`t1`
      blocker_thread: 3561223876
        blocker_user: reinvent
        blocker_host: 123.456.789.012:17746
1 row in set (0.04 sec)
```

Après avoir identifié la session, vous pouvez procédez comme suit :

- Contactez le propriétaire de l'application ou l'utilisateur.
- Si la session de blocage est inactive, pensez à y mettre fin. Une telle action peut déclencher une longue restauration. Pour savoir comment mettre fin à une session, consultez Mettre fin à une session ou à une requête.

Pour plus d'informations sur l'identification des transactions de blocage, consultez Using InnoDB Transaction and Locking Information dans la documentation MySQL.

synch/cond/sql/MDL_context::COND_wait_status

L'événement `synch/cond/sql/MDL_context::COND_wait_status` se produit lorsque des threads sont en attente de verrouillage des métadonnées de table.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL versions 2 et 3

Contexte

L'événement `synch/cond/sql/MDL_context::COND_wait_status` indique que des threads sont en attente de verrouillage des métadonnées de table. Dans certains cas, une session maintient un verrou de métadonnées sur une table et une autre tente d'obtenir le même verrou sur la même table. Si tel est le cas, la seconde session attend l'événement d'attente `synch/cond/sql/MDL_context::COND_wait_status`.

MySQL utilise le verrouillage des métadonnées pour gérer l'accès simultané aux objets de base de données et assurer la cohérence des données. Le verrouillage des métadonnées s'applique aux tables, schémas, événements planifiés, espaces disque logiques et verrous utilisateur acquis avec la fonction `get_lock` et les programmes stockés. Les programmes stockés incluent des procédures, fonctions et déclencheurs. Pour plus d'informations, consultez Metadata Locking dans la documentation MySQL.

La liste du processus MySQL affiche cette session dans l'état `waiting for metadata lock`. Dans Performance Insights, si `Performance_schema` est activé, l'événement `synch/cond/sql/MDL_context::COND_wait_status`apparaît.

Le délai d'expiration par défaut d'une requête en attente d'un verrouillage des métadonnées est basé sur la valeur du paramètre `lock_wait_timeout`, qui s'élève par défaut à 31 536 000 secondes (365 jours).

Pour en savoir plus sur les différents verrous InnoDB et les types de verrous susceptibles d'entraîner des conflits, consultez InnoDB Locking dans la documentation MySQL.

Causes probables de l'augmentation du nombre d'événements d'attente

Lorsque l'événement `synch/cond/sql/MDL_context::COND_wait_status` se produit plus souvent qu'à l'accoutumée, indiquant un possible problème de performance, les causes sont généralement les suivantes :

Transactions de longue durée

Une ou plusieurs transactions modifient une grande quantité de données et maintiennent des verrous sur les tables pendant très longtemps.

Transactions inactives

Une ou plusieurs transactions restent ouvertes pendant longtemps, sans être validées ou annulées.

Instructions DDL sur de grandes tables

Une ou plusieurs instructions en langage de définition de données (DDL), telles que `ALTER TABLE`, ont été exécutées sur de grandes tables.

Verrous de table explicites

Certains verrous de table explicites ne sont pas libérés en temps opportun. Par exemple, une application peut exécuter des instructions `LOCK TABLE` de manière incorrecte.

Actions

Nous vous recommandons différentes actions en fonction des causes d'apparition de votre événement d'attente et de la version du cluster de bases de données Aurora MySQL.

Rubriques

- Identifier les sessions et les requêtes à l'origine des événements
- Rechercher des événements passés
- Exécuter des requêtes sur Aurora MySQL version 2
- Répondre à la session de blocage

Identifier les sessions et les requêtes à l'origine des événements

Vous pouvez utiliser Performance Insights pour afficher les requêtes bloquées par l'événement d'attente `synch/cond/sql/MDL_context::COND_wait_status`. Toutefois, pour identifier la session de blocage, interrogez les tables de métadonnées depuis `performance_schema` et `information_schema` sur le cluster de bases de données.

En règle générale, les bases de données à charge modérée à importante présentent des événements d'attente. Les événements d'attente peuvent être acceptables si les performances sont optimales. Si les performances ne sont pas optimales, voyez où la base de données passe le plus de temps. Examinez les événements d'attente qui contribuent à la charge la plus élevée et voyez si vous pouvez optimiser la base de données et l'application afin de réduire ces événements.

Pour rechercher les requêtes SQL responsables d'une charge élevée

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le volet de navigation, choisissez Performance Insights.
3. Choisissez une instance de base de données. Le tableau de bord Performance Insights correspondant à cette instance de base de données s'affiche.
4. Dans le graphique Database load (Charge de base de données), choisissez Slice by wait (Tranche par attente).
5. Au bas de la page, choisissez Top SQL (Principaux éléments SQL).

   Le graphique répertorie les requêtes SQL responsables de la charge. Les requêtes situées en haut de la liste sont les plus responsables. Pour résoudre un goulet d'étranglement, concentrez-vous sur ces instructions.

Pour une présentation de la résolution des problèmes à l'aide de Performance Insights, consultez le billet de blog AWS Database Amazon Aurora MySQL Workloads with Performance Insights.

Rechercher des événements passés

Pour obtenir des informations sur cet événement d'attente, vous pouvez rechercher ses précédentes occurrences. Pour ce faire, effectuez les actions suivantes :

- Vérifiez le langage de manipulation de données (DML), le débit DDL et la latence pour savoir si la charge de travail a changé.

  Vous pouvez utiliser Performance Insights pour rechercher des requêtes en attente sur cet événement au moment du problème. Vous pouvez également afficher le résumé des requêtes exécutées à un moment proche du problème.
- Si les journaux d'audit ou les journaux généraux sont activés pour le cluster de bases de données, vous pouvez rechercher toutes les requêtes exécutées sur les objets (schema.table) impliqués dans la transaction en attente. Vous pouvez également rechercher les requêtes dont l'exécution a pris fin avant la transaction.

Les informations disponibles pour résoudre les problèmes liés aux événements passés sont limitées. L'exécution de ces vérifications n'indique pas quel objet est en attente d'informations. Cela étant, vous pouvez identifier les tables ayant présenté une charge importante au moment de l'événement

et l'ensemble de lignes fréquemment utilisées ayant provoqué des conflits au moment du problème. Vous pouvez ensuite utiliser ces informations pour reproduire le problème dans un environnement de test et fournir des informations sur sa cause.

Exécuter des requêtes sur Aurora MySQL version 2

Dans Aurora MySQL version 2, vous pouvez identifier directement la session bloquée en interrogeant les tables `performance_schema` ou les vues de schéma `sys`. Un exemple permet d'illustrer comment interroger des tables afin d'identifier les requêtes et les sessions de blocage.

Dans la sortie de la liste de processus suivante, l'ID de connexion 89 attend sur un verrou de métadonnées et exécute une commande `TRUNCATE TABLE`. Dans une requête portant sur les tables `performance_schema` ou les vues de schéma `sys`, la sortie indique que la session de blocage est 76.

```
MySQL [(none)]> select @@version, @@aurora_version;
+-----------+------------------+
| @@version | @@aurora_version |
+-----------+------------------+
| 5.7.12    | 2.11.5           |
+-----------+------------------+
1 row in set (0.01 sec)

MySQL [(none)]> show processlist;
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+------------------------------+
| Id | User            | Host               | db        | Command | Time | State
                    | Info                           |
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+------------------------------+
|  2 | rdsadmin        | localhost          | NULL      | Sleep   |    0 | NULL
                    | NULL                           |
|  4 | rdsadmin        | localhost          | NULL      | Sleep   |    2 | NULL
                    | NULL                           |
|  5 | rdsadmin        | localhost          | NULL      | Sleep   |    1 | NULL
                    | NULL                           |
| 20 | rdsadmin        | localhost          | NULL      | Sleep   |    0 | NULL
                    | NULL                           |
| 21 | rdsadmin        | localhost          | NULL      | Sleep   |  261 | NULL
                    | NULL                           |
| 66 | auroramysql5712 | 172.31.21.51:52154 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                           |
```

```
| 67 | auroramysql5712 | 172.31.21.51:52158 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 68 | auroramysql5712 | 172.31.21.51:52150 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 69 | auroramysql5712 | 172.31.21.51:52162 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 70 | auroramysql5712 | 172.31.21.51:52160 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 71 | auroramysql5712 | 172.31.21.51:52152 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 72 | auroramysql5712 | 172.31.21.51:52156 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 73 | auroramysql5712 | 172.31.21.51:52164 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 74 | auroramysql5712 | 172.31.21.51:52166 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 75 | auroramysql5712 | 172.31.21.51:52168 | sbtest123 | Sleep   |    0 | NULL
                    | NULL                          |
| 76 | auroramysql5712 | 172.31.21.51:52170 | NULL      | Query   |    0 | starting
                    | show processlist              |
| 88 | auroramysql5712 | 172.31.21.51:52194 | NULL      | Query   |   22 | User sleep
                    | select sleep(10000)           |
| 89 | auroramysql5712 | 172.31.21.51:52196 | NULL      | Query   |    5 | Waiting for
 table metadata lock | truncate table sbtest.sbtest1 |
+----+-----------------+--------------------+-----------+---------+------
+---------------------------------+-------------------------------+
18 rows in set (0.00 sec)
```

Ensuite, une requête portant sur les tables `performance_schema` ou les vues de schéma `sys` indique que la session de blocage est 76.

```
MySQL [(none)]> select * from sys.schema_table_lock_waits;

+---------------+-------------+-------------------+-------------
+-------------------------------+-------------------+-----------------------
+-------------------------------+--------------------+-----------------------------
+-------------------------------+--------------------+--------------
+-------------------------------+--------------------+-----------------------
+--------------------------+------------------------------+
| object_schema | object_name | waiting_thread_id | waiting_pid | waiting_account
           | waiting_lock_type | waiting_lock_duration | waiting_query
   | waiting_query_secs | waiting_query_rows_affected | waiting_query_rows_examined |
```

```
 blocking_thread_id | blocking_pid | blocking_account             | blocking_lock_type
 | blocking_lock_duration | sql_kill_blocking_query | sql_kill_blocking_connection |
+---------------+-------------+-------------------+-------------
+------------------------------+--------------------+-----------------------
+------------------------------+--------------------+-----------------------------
+------------------------------+--------------------+-------------
+------------------------------+--------------------+-----------------------
+--------------------------+------------------------------+
| sbtest        | sbtest1     |               121 |          89 |
 auroramysql5712@192.0.2.0    | EXCLUSIVE         | TRANSACTION           | truncate
 table sbtest.sbtest1 |                 10 |                           0 |
               0 |                108 |           76 | auroramysql5712@192.0.2.0    |
 SHARED_READ        | TRANSACTION            | KILL QUERY 76           | KILL 76
              |
+---------------+-------------+-------------------+-------------
+------------------------------+--------------------+-----------------------
+------------------------------+--------------------+-----------------------------
+------------------------------+--------------------+-------------
+------------------------------+--------------------+-----------------------
+--------------------------+------------------------------+
1 row in set (0.00 sec)
```

Répondre à la session de blocage

Après avoir identifié la session, vous pouvez procédez comme suit :

- Contactez le propriétaire de l'application ou l'utilisateur.
- Si la session de blocage est inactive, pensez à y mettre fin. Une telle action peut déclencher une longue restauration. Pour savoir comment mettre fin à une session, consultez Mettre fin à une session ou à une requête.

Pour plus d'informations sur l'identification des transactions de blocage, consultez Using InnoDB Transaction and Locking Information dans la documentation MySQL.

synch/mutex/innodb/aurora_lock_thread_slot_futex

L'événement `synch/mutex/innodb/aurora_lock_thread_slot_futex` se produit lorsqu'une session a verrouillé une ligne pour une mise à jour et qu'une autre session tente de mettre à jour cette même ligne. Pour plus d'informations, consultez InnoDB locking dans la Référence MySQL.

Versions de moteur prises en charge

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL version 2

Causes probables de l'allongement des temps d'attente

Plusieurs instructions en langage de manipulation de données (DML) accèdent simultanément aux mêmes lignes.

Actions

Nous recommandons différentes actions en fonction des autres événement d'attente que vous voyez.

Rubriques

- Rechercher et répondre aux instructions SQL responsables de cet événement d'attente
- Rechercher et répondre à la session de blocage

Rechercher et répondre aux instructions SQL responsables de cet événement d'attente

Utilisez Performance Insights pour identifier les instructions SQL responsables de cet événement d'attente. Envisagez les stratégies suivantes :

- Si les verrous de ligne posent un problème persistant, pensez à réécrire l'application de manière à utiliser un verrouillage optimiste.
- Utiliser plusieurs instructions
- Répartissez la charge de travail sur différents objets de base de données. Vous pouvez le faire via le partitionnement.
- Vérifiez la valeur du paramètre `innodb_lock_wait_timeout`. Il contrôle le délai d'attente des transactions avant de générer une erreur de dépassement.

Pour une vue d'ensemble de la résolution des problème à l'aide de Performance Insights, consultez le billet de blogAnalyze Amazon Aurora MySQL Workloads with Performance Insights.

Rechercher et répondre à la session de blocage

Déterminez si la session de blocage est active ou inactive. Vérifiez également si la session provient d'une application ou d'un utilisateur actif.

Pour identifier la session maintenant le verrou, vous pouvez exécuter `SHOW ENGINE INNODB STATUS`. L'exemple suivant illustre une sortie.

```
mysql> SHOW ENGINE INNODB STATUS;

----------------------TRANSACTION 302631452, ACTIVE 2 sec starting index read
mysql tables in use 1, locked 1
LOCK WAIT 2 lock struct(s), heap size 376, 1 row lock(s)
MySQL thread id 80109, OS thread handle 0x2ae915060700, query id 938819 10.0.4.12
 reinvent updating
UPDATE sbtest1 SET k=k+1 WHERE id=503
------- TRX HAS BEEN WAITING 2 SEC FOR THIS LOCK TO BE GRANTED:
RECORD LOCKS space id 148 page no 11 n bits 30 index `PRIMARY` of table
 `sysbench2`.`sbtest1` trx id 302631452 lock_mode X locks rec but not gap waiting
Record lock, heap no 30 PHYSICAL RECORD: n_fields 6; compact format; info bits 0
```

Vous pouvez également utiliser la requête suivante pour extraire des détails sur les verrous en cours.

```
mysql> SELECT p1.id waiting_thread,
              p1.user waiting_user,
              p1.host waiting_host,
              it1.trx_query waiting_query,
              ilw.requesting_trx_id waiting_transaction,
              ilw.blocking_lock_id blocking_lock,
              il.lock_mode blocking_mode,
              il.lock_type blocking_type,
              ilw.blocking_trx_id blocking_transaction,
              CASE it.trx_state
                WHEN 'LOCK WAIT'
                THEN it.trx_state
                ELSE p.state
              END blocker_state,
              il.lock_table locked_table,
              it.trx_mysql_thread_id blocker_thread,
              p.user blocker_user,
              p.host blocker_host
       FROM information_schema.innodb_lock_waits ilw
       JOIN information_schema.innodb_locks il
```

```
         ON ilw.blocking_lock_id = il.lock_id
        AND ilw.blocking_trx_id = il.lock_trx_id
       JOIN information_schema.innodb_trx it
         ON ilw.blocking_trx_id = it.trx_id
       JOIN information_schema.processlist p
         ON it.trx_mysql_thread_id = p.id
       JOIN information_schema.innodb_trx it1
         ON ilw.requesting_trx_id = it1.trx_id
       JOIN information_schema.processlist p1
         ON it1.trx_mysql_thread_id = p1.id\G

*************************** 1. row ***************************
      waiting_thread: 3561959471
        waiting_user: reinvent
        waiting_host: 123.456.789.012:20485
       waiting_query: select id1 from test.t1 where id1=1 for update
 waiting_transaction: 312337314
       blocking_lock: 312337287:261:3:2
       blocking_mode: X
       blocking_type: RECORD
blocking_transaction: 312337287
       blocker_state: User sleep
        locked_table: `test`.`t1`
      blocker_thread: 3561223876
        blocker_user: reinvent
        blocker_host: 123.456.789.012:17746
1 row in set (0.04 sec)
```

Après avoir identifié la session, vous pouvez procédez comme suit :

- Contactez le propriétaire de l'application ou l'utilisateur.
- Si la session de blocage est inactive, pensez à y mettre fin. Une telle action peut déclencher une longue restauration. Pour savoir comment mettre fin à une session, consultez Mettre fin à une session ou à une requête.

Pour en savoir plus sur l'identification des transactions de blocage, consultez Using InnoDB Transaction and Locking Information dans le Manuel de référence MySQL.

synch/mutex/innodb/buf_pool_mutex

L'événement `synch/mutex/innodb/buf_pool_mutex` se produit lorsqu'un thread a acquis un verrouillage sur le groupe de mémoires tampons InnoDB afin d'accéder à une page en mémoire.

Rubriques

Versions de moteur pertinentes

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL version 2

Contexte

Le mutex `buf_pool` est un mutex unique qui protège les structures de données de contrôle du groupe de mémoires tampons.

Pour plus d'informations, consultez Monitoring InnoDB Mutex Waits Using Performance Schema dans la documentation MySQL.

Causes probables de l'augmentation du nombre d'événements d'attente

Il s'agit d'un événement d'attente spécifique à la charge de travail. Les principales causes liées à l'apparition de `synch/mutex/innodb/buf_pool_mutex` sont les suivantes :

- La taille du groupe de mémoires de tampons n'est pas suffisante pour contenir l'ensemble de données de travail.
- La charge de travail est plus spécifique à certaines pages d'une table spécifique de la base de données, ce qui entraîne une contention dans le groupe de mémoires tampons.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Créer des réplicas Aurora
- Examiner la taille du groupe de mémoires tampons
- Surveiller l'historique global des statuts

Identifier les sessions et les requêtes à l'origine des événements

En règle générale, les bases de données à charge modérée à importante présentent des événements d'attente. Les événements d'attente peuvent être acceptables si les performances sont optimales. Si les performances ne sont pas optimales, voyez où la base de données passe le plus de temps. Examinez les événements d'attente qui contribuent à la charge la plus élevée et voyez si vous pouvez optimiser la base de données et l'application afin de réduire ces événements.

Pour afficher le graphique Top SQL (Principaux éléments SQL) dans la console de gestion AWS

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le volet de navigation, choisissez Performance Insights.
3. Choisissez une instance de base de données. Le tableau de bord Performance Insights s'affiche pour cette instance de base de données.
4. Dans le graphique Database load (Charge de base de données), choisissez Slice by wait (Tranche par attente).
5. Sous le graphique Data load (Charge de base de données), choisissezTop SQL (Principaux éléments SQL).

   Le graphique répertorie les requêtes SQL responsables de la charge. Les requêtes situées en haut de la liste sont les plus responsables. Pour résoudre un goulet d'étranglement, concentrez-vous sur ces instructions.

Pour une vue d'ensemble de la résolution des problème à l'aide de Performance Insights, consultez le billet de blogAnalyze Amazon Aurora MySQL Workloads with Performance Insights.

Utiliser Performance Insights

Cet événement est lié à la charge de travail. Vous pouvez utiliser Performance Insights pour effectuer les opérations suivantes :

- Identifier le moment où les événements d'attente démarrent et si la charge de travail change à ce moment-là à partir des journaux d'application ou des sources associées.

- Identifier les instructions SQL responsables de cet événement d'attente. Examiner le plan d'exécution des requêtes pour vous assurer que ces requêtes sont optimisées et utilisent des index appropriés.

  Si les principales requêtes responsables de l'événement d'attente sont liées au même objet ou table de base de données, pensez à partitionner cet objet ou cette table.

Créer des réplicas Aurora

Vous pouvez créer des réplicas Aurora pour gérer le trafic en lecture seule. Vous pouvez également utiliser Aurora Auto Scaling pour gérer les pics de trafic en lecture. Assurez-vous d'exécuter des tâches planifiées en lecture seule et des sauvegardes logiques sur les réplicas Aurora.

Pour de plus amples informations, consultez Amazon Aurora Auto Scaling avec des réplicas Aurora.

Examiner la taille du groupe de mémoires tampons

Vérifiez si la taille du groupe de mémoires tampons est suffisante pour la charge de travail en examinant la métrique `innodb_buffer_pool_wait_free`. Si la valeur de cette métrique est élevée et augmente continuellement, cela indique que la taille du groupe de mémoires tampons n'est pas suffisante pour gérer la charge de travail. Si `innodb_buffer_pool_size` a été correctement défini, la valeur de `innodb_buffer_pool_wait_free` doit être devrait être faible. Pour plus d'informations, consultez Innodb_buffer_pool_wait_free dans la documentation MySQL.

Augmentez la taille du groupe de mémoires tampons si l'instance de base de données dispose de suffisamment de mémoire pour les tampons de session et les tâches du système d'exploitation. Dans le cas contraire, remplacez l'instance de base de données par une classe d'instance de base de données plus grande pour obtenir plus de mémoire à allouer au groupe de mémoires tampons.

> **Note**
>
> Aurora MySQL ajuste automatiquement la valeur `innodb_buffer_pool_instances` en fonction du paramètre `innodb_buffer_pool_size` configuré.

Surveiller l'historique global des statuts

La surveillance des taux de modification des variables de statut vous permet de détecter les problèmes de verrouillage ou de mémoire sur votre instance de base de données. Si ce n'est pas

déjà fait, activez l'historique global des statuts (GoSH). Pour en savoir plus sur GoSH, consultez Gestion de l'historique global des statuts.

Vous pouvez également créer des métriques Amazon CloudWatch personnalisées afin de surveiller les variables de statut. Pour plus d'informations, consultez Publication de métriques personnalisées.

synch/mutex/innodb/fil_system_mutex

L'événement `synch/mutex/innodb/fil_system_mutex` se produit lorsqu'une session attend d'accéder au cache mémoire de l'espace disque logique.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL versions 2 et 3

Contexte

InnoDB utilise des espaces de table pour gérer la zone de stockage des tables et des fichiers journaux. Le cache mémoire de l'espace disque logique est une structure de mémoire globale qui tient à jour les informations relatives aux espaces de table. MySQL utilise les attentes `synch/mutex/innodb/fil_system_mutex` pour contrôler l'accès simultané au cache mémoire de l'espace disque logique.

L'événement `synch/mutex/innodb/fil_system_mutex` indique qu'il existe actuellement plusieurs opérations qui doivent récupérer et manipuler des informations dans le cache mémoire de l'espace disque logique pour le même espace de table.

Causes probables de l'augmentation du nombre d'événements d'attente

Lorsque l'événement `synch/mutex/innodb/fil_system_mutex` se produit plus souvent qu'à l'accoutumée, indiquant un possible problème de performance, toutes les conditions suivantes sont généralement réunies :

- Augmentation des opérations en langage de manipulation de données (DML) simultanées mettant à jour ou supprimant des données dans la même table.
- L'espace disque logique de cette table est très volumineux et contient de nombreuses pages de données.
- Le facteur de remplissage de ces pages de données est faible.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Identifier les sessions et les requêtes à l'origine des événements
- Réorganiser les tables volumineuses pendant les heures creuses

Identifier les sessions et les requêtes à l'origine des événements

En règle générale, les bases de données à charge modérée à importante présentent des événements d'attente. Les événements d'attente peuvent être acceptables si les performances sont optimales. Si les performances ne sont pas optimales, voyez où la base de données passe le plus de temps. Examinez les événements d'attente qui contribuent à la charge la plus élevée et voyez si vous pouvez optimiser la base de données et l'application afin de réduire ces événements.

Pour rechercher les requêtes SQL responsables d'une charge élevée

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le volet de navigation, choisissez Performance Insights.
3. Choisissez une instance de base de données. Le tableau de bord Performance Insights correspondant à cette instance de base de données s'affiche.
4. Dans le graphique Database load (Charge de base de données), choisissez Slice by wait (Tranche par attente).

5. Au bas de la page, choisissez Top SQL (Principaux éléments SQL).

   Le graphique répertorie les requêtes SQL responsables de la charge. Les requêtes situées en haut de la liste sont les plus responsables. Pour résoudre un goulet d'étranglement, concentrez-vous sur ces instructions.

Pour une vue d'ensemble de la résolution des problème à l'aide de Performance Insights, consultez le billet de blogAnalyze Amazon Aurora MySQL Workloads with Performance Insights.

Pour savoir quelles requêtes entraînent un grand nombre d'attentes `synch/mutex/innodb/fil_system_mutex`, il est également possible de consulter `performance_schema`, comme dans l'exemple suivant.

```
mysql> select * from performance_schema.events_waits_current where EVENT_NAME='wait/
synch/mutex/innodb/fil_system_mutex'\G
*************************** 1. row ***************************
            THREAD_ID: 19
             EVENT_ID: 195057
         END_EVENT_ID: 195057
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:6700
          TIMER_START: 1010146190118400
            TIMER_END: 1010146196524000
           TIMER_WAIT: 6405600
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
*************************** 2. row ***************************
            THREAD_ID: 23
             EVENT_ID: 5480
         END_EVENT_ID: 5480
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:5906
          TIMER_START: 995269979908800
```

```
            TIMER_END: 995269980159200
          TIMER_WAIT: 250400
               SPINS: NULL
       OBJECT_SCHEMA: NULL
         OBJECT_NAME: NULL
          INDEX_NAME: NULL
         OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
*************************** 3. row ***************************
            THREAD_ID: 55
             EVENT_ID: 23233794
         END_EVENT_ID: NULL
           EVENT_NAME: wait/synch/mutex/innodb/fil_system_mutex
               SOURCE: fil0fil.cc:449
          TIMER_START: 1010492125341600
            TIMER_END: 1010494304900000
           TIMER_WAIT: 2179558400
                SPINS: NULL
       OBJECT_SCHEMA: NULL
         OBJECT_NAME: NULL
          INDEX_NAME: NULL
         OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 47285552262176
     NESTING_EVENT_ID: 23233786
   NESTING_EVENT_TYPE: WAIT
            OPERATION: lock
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
```

Réorganiser les tables volumineuses pendant les heures creuses

Réorganisez les tables volumineuses que vous identifiez en tant que source d'un nombre élevé d'événements d'attente `synch/mutex/innodb/fil_system_mutex` pendant une fenêtre de maintenance en dehors des heures de production. Ainsi, le nettoyage des mappages d'espace disque logique interne n'intervient pas lorsqu'un accès rapide à la table est essentiel. Pour plus d'informations sur la réorganisation des tables, consultez OPTIMIZE TABLE Statement dans la Référence MySQL.

synch/mutex/innodb/trx_sys_mutex

L'événement `synch/mutex/innodb/trx_sys_mutex` se produit lorsque l'activité de base de données est élevée et présente un grand nombre de transactions.

Rubriques

Versions de moteur pertinentes

Ces informations relatives aux événements d'attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL versions 2 et 3

Contexte

En interne, le moteur de base de données InnoDB utilise le niveau d'isolement de lecture renouvelée avec instantanés pour assurer la cohérence en lecture. Vous disposez ainsi d'une vue à un instant dans le passé de la base de données lors de la création de l'instantané.

Dans InnoDB, toutes les modifications sont appliquées à la base de données dès leur arrivée, qu'elles soient validées ou non. Une telle approche signifie que sans contrôle de simultanéité multiversion (MVCC), tous les utilisateurs connectés à la base de données voient toutes les modifications et les dernières lignes. Par conséquent, InnoDB doit pouvoir suivre les modifications pour comprendre les éléments à annuler, si nécessaire.

Pour ce faire, InnoDB utilise un système de transactions (`trx_sys`) lui permettant de suivre les instantanés. Le système de transactions procède comme suit :

- Il suit l'ID de transaction de chaque ligne dans les journaux d'annulation.
- Il utilise une structure InnoDB interne appelée `ReadView` qui permet d'identifier les ID de transaction visibles pour un instantané.

Causes probables de l'augmentation du nombre d'événements d'attente

Toute opération de base de données nécessitant une gestion cohérente et contrôlée (création, lecture, mise à jour et suppression) des ID de transaction génère un appel entre `trx_sys` et le mutex.

Ces appels se produisent dans trois fonctions :

- `trx_sys_mutex_enter` – Crée le mutex.
- `trx_sys_mutex_exit` – Libère le mutex.
- `trx_sys_mutex_own` – Teste si le mutex a un propriétaire.

L'instrumentation du schéma de performance InnoDB suit tous les appels du mutex `trx_sys`. Le suivi inclut, sans s'y limiter, les opérations suivantes : gestion de `trx_sys` lorsque la base de données démarre ou s'arrête, restaurations, nettoyages après annulation, accès en lecture de ligne et charges de groupe de mémoires tampons. Une activité de base de données élevée avec un grand nombre de transactions entraîne l'apparition de `synch/mutex/innodb/trx_sys_mutex` parmi les principaux événements d'attente.

Pour plus d'informations, consultez Monitoring InnoDB Mutex Waits Using Performance Schema dans la documentation MySQL.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Identifier les sessions et les requêtes à l'origine des événements
- Examiner d'autres événements d'attente

Identifier les sessions et les requêtes à l'origine des événements

En règle générale, les bases de données à charge modérée à importante présentent des événements d'attente. Les événements d'attente peuvent être acceptables si les performances sont optimales. Si les performances ne sont pas optimales, voyez où la base de données passe le plus de temps. Examinez les événements d'attente qui contribuent à la charge la plus élevée. Voyez si vous pouvez optimiser la base de données et l'application de manière à réduire ces événements.

Pour afficher le graphique Top SQL (Principaux éléments SQL) dans la AWS Management Console

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le volet de navigation, choisissez Performance Insights.
3. Choisissez une instance de base de données. Le tableau de bord Performance Insights s'affiche pour cette instance de base de données.
4. Dans le graphique Database load (Charge de base de données), choisissez Slice by wait (Tranche par attente).
5. Sous le graphique Data load (Charge de base de données), choisissezTop SQL (Principaux éléments SQL).

   Le graphique répertorie les requêtes SQL responsables de la charge. Les requêtes situées en haut de la liste sont les plus responsables. Pour résoudre un goulet d'étranglement, concentrez-vous sur ces instructions.

Pour une vue d'ensemble de la résolution des problème à l'aide de Performance Insights, consultez le billet de blogAnalyze Amazon Aurora MySQL Workloads with Performance Insights.

Examiner d'autres événements d'attente

Examinez les autres événements d'attente associés à l'événement d'attente `synch/mutex/innodb/trx_sys_mutex`. Ils peuvent vous fournir plus d'informations sur la nature de la charge de travail. Un grand nombre de transactions peuvent réduire le débit, mais la charge de travail peut également l'imposer.

Pour en savoir plus sur l'optimisation des transactions, consultez Optimizing InnoDB Transaction Managementdans la documentation MySQL.

synch/sxlock/innodb/hash_table_locks

L'événement `synch/sxlock/innodb/hash_table_locks` se produit lorsque des pages introuvables dans le groupe de mémoires tampons doivent être lues à partir d'un fichier.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente

- Actions

Versions de moteur prises en charge

Ces informations relatives aux événements d'attente sont prises en charge dans les versions suivantes :

- Aurora MySQL versions 2 et 3

Contexte

L'événement `synch/sxlock/innodb/hash_table_locks` indique qu'une charge de travail accède fréquemment à des données qui ne sont pas stockées dans le groupe de mémoires tampons. Cet événement d'attente est associé à des ajouts de nouvelles pages et expulsions d'anciennes données du groupe de mémoires tampons. Les données stockées dans le groupe de mémoires tampons correspondant aux anciennes et nouvelles données doivent être mises en cache pour permettre l'expulsion des anciennes pages et la mise en cache des nouvelles pages. MySQL utilise un algorithme LRU (Last Recently Used) pour expulser les pages du groupe de mémoires tampons. La charge de travail tente d'accéder aux données qui n'ont pas été chargées dans le groupe de mémoires tampons ou aux données qui ont été expulsées de ce dernier.

Cet événement d'attente se produit lorsque la charge de travail doit accéder aux données de fichiers sur disque ou lorsque des blocs sont libérés ou ajoutés à la liste LRU du groupe de mémoires tampons. Ces opérations attendent d'obtenir un verrou partagé exclu (SX-Lock). Ce verrou SX-Lock est utilisé pour la synchronisation sur la table de hachage, qui est une table en mémoire conçue pour améliorer les performances d'accès au groupe de mémoires tampons.

Pour plus d'informations, consultez Buffering and Caching dans la documentation MySQL.

Causes probables de l'augmentation du nombre d'événements d'attente

Lorsque l'événement d'attente `synch/sxlock/innodb/hash_table_locks` se produit plus souvent qu'à l'accoutumée, indiquant un possible problème de performance, les causes sont généralement les suivantes :

Groupe de mémoires tampons sous-dimensionné

La taille du groupe de mémoires tampons est insuffisante pour conserver en mémoire toutes les pages fréquemment consultées.

Charge de travail importante

La charge de travail entraîne de fréquentes expulsions et le rechargement de pages de données dans le cache de tampon.

Erreurs de lecture des pages

Le groupe de mémoires tampons présente des erreurs de lectures de pages, ce qui peut indiquer une corruption des données.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Augmentez la taille du groupe de mémoires tampons
- Améliorer les modèles d'accès aux données
- Réduire ou éviter les analyses de table entière
- Rechercher une éventuelle corruption des pages dans les journaux d'erreurs

Augmentez la taille du groupe de mémoires tampons

Assurez-vous que le groupe de mémoires tampons est correctement dimensionné pour la charge de travail. Pour ce faire, vous pouvez vérifier le taux d'accès au cache du groupe de mémoires tampons. En règle générale, si la valeur est inférieure à 95 %, envisagez d'augmenter la taille du groupe de mémoires tampons. Un groupe de mémoires tampons plus important peut conserver les pages fréquemment consultées en mémoire plus longtemps. Pour augmenter la taille du groupe de mémoires tampons, modifiez la valeur du paramètre `innodb_buffer_pool_size`. La valeur par défaut de ce paramètre dépend de la taille de la classe d'instance de base de données. Pour plus d'informations, consultez Bonnes pratiques relatives à la configuration de base de données Amazon Aurora MySQL.

Améliorer les modèles d'accès aux données

Vérifiez les requêtes affectées par cette attente ainsi que leurs plans d'exécution. Pensez à améliorer les modèles d'accès aux données. Par exemple, si vous utilisez mysqli_result::fetch_array, vous pouvez essayer d'augmenter la taille d'extraction du tableau.

Vous pouvez utiliser Performance Insights pour afficher les requêtes et les sessions susceptibles d'entraîner l'événement d'attente `synch/sxlock/innodb/hash_table_locks`.

Pour rechercher les requêtes SQL responsables d'une charge élevée

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Performance Insights.
3. Choisissez une instance de base de données. Le tableau de bord Performance Insights s'affiche pour cette instance de base de données.
4. Dans le graphique Database load (Charge de base de données), choisissez Slice by wait (Tranche par attente).
5. Au bas de la page, choisissez Top SQL (Principaux éléments SQL).

   Le graphique répertorie les requêtes SQL responsables de la charge. Les requêtes situées en haut de la liste sont les plus responsables. Pour résoudre un goulet d'étranglement, concentrez-vous sur ces instructions.

Pour une présentation de la résolution des problèmes à l'aide de Performance Insights, consultez le billet de blog AWS Database Amazon Aurora MySQL Workloads with Performance Insights.

Réduire ou éviter les analyses de table entière

Surveillez votre charge de travail pour voir si elle exécute des analyses de table entière et, si tel est le cas, les réduire ou les éviter. Par exemple, vous pouvez surveiller des variables d'état telles que `Handler_read_rnd_next`. Pour plus d'informations, consultez Server Status Variables dans la documentation MySQL.

Rechercher une éventuelle corruption des pages dans les journaux d'erreurs

Vous pouvez consulter le journal mysql-error.log afin d'y détecter d'éventuels messages de corruption au moment du problème. Les messages à utiliser pour résoudre le problème se trouvent dans le journal des erreurs. Vous devrez peut-être recréer les objets signalés comme corrompus.

synch/mutex/innodb/temp_pool_manager_mutex

L'événement `synch/mutex/innodb/temp_pool_manager_mutex` d'attente se produit lorsqu'une session attend d'acquérir un mutex pour gérer le pool de tablespaces temporaires de session.

Rubriques

Versions de moteur prises en charge

Ces informations relatives aux événements d’attente sont prises en charge pour les versions de moteur suivantes :

- Aurora MySQL version 3

Contexte

Aurora MySQL version 3.x et versions ultérieures permet `temp_pool_manager_mutex` de contrôler plusieurs sessions accédant simultanément au pool d'espaces de table temporaires. Aurora MySQL gère le stockage via un volume de cluster Aurora pour les données persistantes et le stockage local pour les fichiers temporaires. Un tablespace temporaire est nécessaire lorsqu'une session crée une table temporaire sur le volume du cluster Aurora.

Lorsqu'une session demande pour la première fois un tablespace temporaire, MySQL alloue des tablespaces temporaires de session depuis le pool partagé. Une session peut contenir jusqu'à 2 tablespaces temporaires à la fois pour les types de tables suivants :

- Tables temporaires créées par l'utilisateur
- Tables temporaires internes générées par l'optimiseur

Le `TempTable` moteur par défaut utilise le mécanisme de débordement suivant pour gérer les tables temporaires :

- Stocke les tables dans la RAM jusqu'à la `temptable_max_ram`limite.
- Se déplace vers les fichiers mappés en mémoire sur le stockage local lorsque la RAM est pleine.
- Utilise le volume de cluster partagé lorsque les fichiers mappés en mémoire atteignent leur limite. `temptable_max_mmap`

Lorsque les tables temporaires dépassent à la fois les limites de RAM et de stockage local, MySQL les gère à l'aide d'un espace disque logique.

Lorsqu'une session nécessite une table temporaire sur disque, MySQL :

- Recherche les `INACTIVE` tablespaces disponibles dans le pool à réutiliser.
- Crée 10 nouveaux tablespaces s'il n'en existe aucun`INACTIVE`.

Lorsqu'une session se déconnecte, MySQL :

- Tronque les tablespaces temporaires de la session.
- Les marque comme INACTIFS dans le pool en vue de leur réutilisation.
- Maintient la taille actuelle du pool jusqu'au redémarrage du serveur.
- Revient à la taille du pool par défaut (10 tablespaces) après le redémarrage.

Causes probables de l’augmentation du nombre d’événements d’attente

Situations courantes à l'origine de cet événement d'attente :

- Sessions simultanées créant des tables temporaires internes sur le volume du cluster.
- Sessions simultanées créant des tables temporaires utilisateur sur le volume du cluster.
- Fin soudaine de sessions utilisant des tablespaces actifs.
- Expansion du pool de tablespaces lors de lourdes charges d'écriture.
- Accès aux requêtes simultanées `INFORMATION_SCHEMA.`

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d’attente.

Rubriques

- Surveillez et optimisez l'utilisation des tables temporaires
- Révision des requêtes à l'aide de INFORMATION_SCHEMA
- Augmenter le paramètre innodb_sync_array_size
- Implémenter un groupe de connexions

Surveillez et optimisez l'utilisation des tables temporaires

Pour surveiller et optimiser l'utilisation des tables temporaires, appliquez l'une des méthodes suivantes :

- Consultez le `Created_tmp_disk_tables` compteur dans Performance Insights pour suivre la création de tables temporaires sur disque dans votre cluster Aurora.
- Exécutez cette commande dans votre base de données pour surveiller directement la création de tables temporaires :`mysql> show status like '%created_tmp_disk%'`.

> **Note**
>
> Le comportement des tables temporaires diffère entre les nœuds de lecture et d'écriture Aurora MySQL. Pour de plus amples informations, veuillez consulter Nouveau comportement de table temporaire dans Aurora MySQL version 3.

Après avoir identifié les requêtes créant des tables temporaires, suivez les étapes d'optimisation suivantes :

- `EXPLAIN`À utiliser pour examiner les plans d'exécution des requêtes et identifier où et pourquoi les tables temporaires sont créées.
- Modifiez les requêtes pour réduire l'utilisation des tables temporaires dans la mesure du possible.

Si l'optimisation des requêtes ne résout pas à elle seule les problèmes de performances, pensez à ajuster les paramètres de configuration suivants :

- `temptable_max_ram`- Contrôle l'utilisation maximale de la RAM pour les tables temporaires.
- `temptable_max_mmap`- Définit la limite pour le stockage de fichiers mappé en mémoire.
- `tmp_table_size`- S'applique lorsqu'`aurora_tmptable_enable_per_table_limit`il est activé (désactivé par défaut).

> **Important**
>
> Notez que certaines conditions de requête nécessiteront toujours des tables temporaires sur disque, quels que soient les paramètres de configuration. Pour plus d'informations sur le

moteur de `TempTable` stockage, consultez Utiliser le moteur TempTable de stockage sur Amazon RDS for MySQL et Amazon Aurora MySQL.

Révision des requêtes à l'aide de INFORMATION_SCHEMA

Lorsque vous interrogez `INFORMATION_SCHEMA` des tables, MySQL crée des tables temporaires InnoDB sur le volume du cluster. Chaque session a besoin d'un tablespace temporaire pour ces tables, ce qui peut entraîner des problèmes de performances en cas d'accès simultané élevé.

Pour améliorer les performances :

- À utiliser `PERFORMANCE_SCHEMA` au lieu de `INFORMATION_SCHEMA` là où c'est possible.
- Si vous devez en utiliser`INFORMATION_SCHEMA`, réduisez la fréquence à laquelle vous exécutez ces requêtes.

Augmenter le paramètre innodb_sync_array_size

Le `innodb_sync_array_size` paramètre contrôle la taille du tableau d' mutex/lock attente dans MySQL. La valeur par défaut de 1 fonctionne pour les charges de travail générales, mais son augmentation peut réduire la contention des threads en cas de forte simultanéité.

Lorsque votre charge de travail affiche un nombre croissant de threads en attente :

- Surveillez le nombre de threads en attente dans votre charge de travail.
- Définissez un nombre `innodb_sync_array_size` égal ou supérieur au nombre de vCPU de votre instance pour diviser la structure de coordination des threads et réduire les conflits.

> **Note**
>
> Pour déterminer le nombre de v CPUs disponibles sur votre instance RDS, consultez les spécifications des vCPU dans les types d'instances Amazon RDS.

Implémenter un groupe de connexions

MySQL attribue un tablespace dédié à chaque session qui crée une table temporaire. Ce tablespace reste actif jusqu'à la fin de la connexion à la base de données. Pour gérer vos ressources de manière plus efficace :

- Implémentez le regroupement de connexions pour limiter le nombre de tablespaces temporaires actifs.
- Réutilisez les connexions existantes au lieu d'en créer de nouvelles pour chaque opération.

Réglage d'Aurora MySQL avec des états de thread

Le table suivant récapitule les états généraux de thread les plus courants pour Aurora MySQL.

État général de thread	Description
???	Cet état de thread indique qu'un thread traite une instruction `SELECT` qui requiert l'utilisation d'une table temporaire interne pour trier les données.
???	Cet état de thread indique qu'un thread lit et filtre les lignes d'une requête afin de déterminer l'ensemble de résultats qui convient.

creating sort index

L'état de thread `creating sort index` pour indique qu'un thread traite une instruction `SELECT` qui requiert l'utilisation d'une table temporaire interne pour trier les données.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations relatives aux états de thread sont prises en charge dans les versions suivantes :

- Aurora MySQL versions 2 à 2.09.2

Contexte

L'état `creating sort index` apparaît lorsqu'une requête avec une clause `ORDER BY` ou `GROUP BY` ne peut pas utiliser un index existant pour effectuer l'opération. Dans ce cas, MySQL doit effectuer

une opération `filesort` plus coûteuse. Cette opération s'effectue généralement en mémoire si l'ensemble de résultats n'est pas trop volumineux. Sinon, elle implique la création d'un fichier sur disque.

Causes probables de l'augmentation du nombre d'événements d'attente

L'apparition de `creating sort index` n'indique pas en soi un problème. Si les performances sont médiocres et que les instances de `creating sort index` se multiplient, il y a fort à parier que cela soit dû à la lenteur des requêtes avec les opérateurs `ORDER BY` ou `GROUP BY`.

Actions

De manière générale, nous vous conseillons de déterminer les requêtes avec des clauses `ORDER BY` ou `GROUP BY` associées aux augmentations de l'état `creating sort index`. Voyez ensuite si l'ajout d'un index ou l'augmentation de la taille du tampon de tri permet de résoudre le problème.

Rubriques

- Activer le schéma de performance, le cas échéant
- Identifier les requêtes problématiques
- Examiner les plans d'explication de l'utilisation de filesort
- Augmenter la taille du tampon de tri

Activer le schéma de performance, le cas échéant

Performance Insights signale les états de thread uniquement si les instruments du schéma de performance ne sont pas activés. Lorsque les instruments du schéma de performance sont activés, Performance Insights signale plutôt les événements d'attente. Les instruments du schéma de performance fournissent des informations supplémentaires et de meilleurs outils pour examiner les possibles problèmes de performances. Par conséquent, nous vous recommandons d'activer le schéma de performance. Pour plus d'informations, consultez Présentation du schéma de performance pour Performance Insights sur Aurora MySQL.

Identifier les requêtes problématiques

Pour identifier les requêtes actuelles entraînant des augmentations de l'état `creating sort index`, exécutez `show processlist` et vérifiez si l'une des requêtes est accompagnées de `ORDER BY` ou `GROUP BY`. Vous pouvez également exécuter `explain for connection N`, où `N` correspond à l'ID de liste de processus de la requête avec `filesort`.

Pour identifier les requêtes antérieures à l'origine de ces augmentations, activez le journal des requêtes lentes et recherchez les requêtes avec `ORDER BY`. Exécutez `EXPLAIN` sur les requêtes lentes et recherchez « using filesort ». Pour plus d'informations, consultez Examiner les plans d'explication de l'utilisation de filesort.

Examiner les plans d'explication de l'utilisation de filesort

Identifiez les déclarations accompagnées de clauses `ORDER BY` ou `GROUP BY` aboutissant à l'état `creating sort index`.

L'exemple suivant illustre l'exécution de `explain` sur une requête. La colonne `Extra` indique que cette requête utilise `filesort`.

```
mysql> explain select * from mytable order by c1 limit 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mytable
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 2064548
     filtered: 100.00
        Extra: Using filesort
1 row in set, 1 warning (0.01 sec)
```

L'exemple suivant illustre le résultat de l'exécution de `EXPLAIN` sur la même requête après la création d'un index sur la colonne `c1`.

```
mysql> alter table mytable add index (c1);
```

```
mysql> explain select * from mytable order by c1 limit 10\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: mytable
   partitions: NULL
         type: index
possible_keys: NULL
```

```
          key: c1
      key_len: 1023
          ref: NULL
         rows: 10
     filtered: 100.00
        Extra: Using index
1 row in set, 1 warning (0.01 sec)
```

Pour plus d'informations sur l'utilisation des index à des fins d'optimisation de l'ordre de tri, consultez ORDER BY Optimization dans la documentation MySQL.

Augmenter la taille du tampon de tri

Pour savoir si une requête spécifique a nécessité un processus `filesort` qui a créé un fichier sur disque, vérifiez la valeur de la variable `sort_merge_passes` après l'exécution de la requête. Vous en trouverez un exemple ci-dessous.

```
mysql> show session status like 'sort_merge_passes';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
+-------------------+-------+
1 row in set (0.01 sec)

--- run query
mysql> select * from mytable order by u limit 10;
--- run status again:

mysql> show session status like 'sort_merge_passes';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| Sort_merge_passes | 0     |
+-------------------+-------+
1 row in set (0.01 sec)
```

Si la valeur de `sort_merge_passes` est élevée, envisagez d'augmenter la taille du tampon de tri. Appliquez l'augmentation au niveau de la session, car une augmentation globale peut considérablement accroître la quantité de RAM utilisée par MySQL. L'exemple suivant montre comment modifier la taille du tampon de tri avant d'exécuter une requête.

```
mysql> set session sort_buffer_size=10*1024*1024;
Query OK, 0 rows affected (0.00 sec)
-- run query
```

envoi de données

L'état de thread `sending data` indique qu'un thread lit et filtre les lignes d'une requête afin de déterminer l'ensemble de résultats qui convient. Le nom est trompeur car il implique que l'état transfère des données, sans collecter ni préparer les données à envoyer ultérieurement.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations relatives aux états de thread sont prises en charge dans les versions suivantes :

- Aurora MySQL versions 2 à 2.09.2

Contexte

De nombreux états de thread sont de courte durée. Les opérations intervenant pendant `sending data` ont tendance à effectuer un grand nombre de lectures de disque ou de cache. Par conséquent, `sending data` est souvent l'état le plus long au cours de la durée de vie d'une requête donnée. Cet état apparaît lorsqu'Aurora MySQL effectue les opérations suivantes :

- Lecture et traitement de lignes pour une instruction `SELECT`
- Exécution d'un grand nombre de lectures à partir d'un disque ou d'une mémoire
- Exécution d'une lecture complète de toutes les données d'une requête spécifique
- Lecture de données à partir d'une table, d'un index ou du travail d'une procédure stockée
- Tri, regroupement ou classement des données

Après que l'état `sending data` a terminé de préparer les données, l'état de thread `writing to net` indique le renvoi des données au client. En règle générale, `writing to net` est uniquement capturé lorsque l'ensemble de résultats est très volumineux ou qu'une importante latence réseau ralentit le transfert.

Causes probables de l'augmentation du nombre d'événements d'attente

L'apparition de `sending data` n'indique pas en soi un problème. Si les performances sont médiocres et que les instances de `sending data` se multiplient, les causes les plus probables sont les suivantes.

Rubriques

- Requête inefficace
- Configuration sous-optimale du serveur

Requête inefficace

Dans la plupart des cas, cet état découle d'une requête qui n'utilise pas d'index approprié pour trouver l'ensemble de résultats d'une requête spécifique. Par exemple, considérez une requête lisant une table de 10 millions d'enregistrements correspondant à l'ensemble des commandes passées en Californie, où la colonne d'état n'est pas indexée ou mal indexée. Dans ce dernier cas, l'index peut exister, mais l'optimiseur l'ignore en raison de sa faible cardinalité.

Configuration sous-optimale du serveur

Si plusieurs requêtes apparaissent dans l'état `sending data`, le serveur de base de données peut être mal configuré. Plus précisément, le serveur peut se heurter aux problèmes suivants :

- Le serveur de base de données ne dispose pas d'une capacité de calcul suffisante : I/O disque, type et vitesse de disque, processeur ou nombre de processeurs.
- Le serveur consomme beaucoup de ressources allouées, telles que le groupe de mémoires tampons InnoDB pour les tables InnoDB ou le tampon de clé pour les tables MyIsam.
- Les paramètres de mémoire par thread tels que `sort_buffer`,`read_buffer` et `join_buffer`consomment plus de RAM que nécessaire, privant le serveur physique de ressources mémoire.

Actions

De manière générale, nous vous conseillons de rechercher les requêtes qui renvoient un grand nombre de lignes en vérifiant le schéma de performance. Si les requêtes de journalisation n'utilisant pas d'index sont activées, vous pouvez également examiner les résultats des journaux lents.

Rubriques

- Activer le schéma de performance, le cas échéant
- Examiner les paramètres de mémoire
- Examiner les plans d'explication de l'utilisation d'index
- Vérifier le volume de données renvoyées
- Vérifier les problèmes de simultanéité
- Vérifier la structure de vos requêtes

Activer le schéma de performance, le cas échéant

Performance Insights signale les états de thread uniquement si les instruments du schéma de performance ne sont pas activés. Lorsque les instruments du schéma de performance sont activés, Performance Insights signale plutôt les événements d'attente. Les instruments du schéma de performance fournissent des informations supplémentaires et de meilleurs outils pour examiner les possibles problèmes de performances. Par conséquent, nous vous recommandons d'activer le schéma de performance. Pour de plus amples informations, consultez Présentation du schéma de performance pour Performance Insights sur Aurora MySQL.

Examiner les paramètres de mémoire

Examinez les paramètres de mémoire des groupes de mémoires tampons principaux. Assurez-vous que ces groupes de mémoires tampons sont correctement dimensionné pour la charge de travail. Si votre base de données utilise plusieurs instances de groupe de mémoires tampons, assurez-vous qu'elles ne sont pas divisées en petits groupes de mémoires tampons. Les threads ne peuvent utiliser qu'un seul groupe de mémoires tampons à la fois.

Assurez-vous que les paramètres de mémoire suivants utilisés pour chaque thread sont correctement dimensionnés :

- read_buffer
- read_rnd_buffer
- sort_buffer

- join_buffer
- binlog_cache

Sauf raison spécifique de les modifier, utilisez les valeurs par défaut des paramètres.

Examiner les plans d'explication de l'utilisation d'index

Pour les requêtes dans l'état de thread `sending data`, examinez le plan pour déterminer si des index appropriés sont utilisés. Si une requête n'utilise pas d'index utile, envisagez d'ajouter des indicateurs tels que `USE INDEX` ou `FORCE INDEX`. Les indicateurs peuvent considérablement augmenter ou diminuer le délai nécessaire à l'exécution d'une requête et par conséquent, ajoutez-les à bon escient.

Vérifier le volume de données renvoyées

Vérifiez les tables interrogées et la quantité de données qu'elles contiennent. Certaines de ces données peuvent-elles être archivées ? Très souvent, de piètres délai d'exécution des requêtes ne relèvent pas du plan des requête, mais du volume de données à traiter. De nombreux développeurs ajoutent facilement des données à une base de données, mais rares sont ceux à prendre en compte le cycle de vie des jeux de données lors des phases de conception et de développement.

Recherchez les requêtes qui fonctionnent bien dans les bases de données à faible volume, mais qui fonctionnent nettement moins bien dans votre système actuel. Parfois, les développeurs qui conçoivent des requêtes spécifiques ne réalisent pas que ces requêtes renvoient 350 000 lignes. Les développeurs ont peut-être développé les requêtes dans un environnement à faible volume avec des jeux de données plus petits que ceux des environnements de production.

Vérifier les problèmes de simultanéité

Vérifiez si plusieurs requêtes de même type sont exécutées en même temps. Certaines formes de requêtes s'exécutent efficacement lorsqu'elles sont exécutées seules. Toutefois, si des formes de requête similaires sont exécutées ensemble ou à un volume élevé, elles peuvent entraîner des problèmes de simultanéité. Ces problèmes surviennent souvent lorsque la base de données utilise des tables temporaires pour afficher les résultats. Un niveau d'isolement restrictif des transactions peut également entraîner des problèmes de simultanéité.

Si les tables sont lues et écrites simultanément, la base de données peut utiliser des verrous. Pour mieux identifier les périodes où les performances sont médiocres, examinez l'utilisation des bases de données via des processus par lots à grande échelle. Pour voir les verrous et restaurations récents, examinez la sortie de la commande `SHOW ENGINE INNODB STATUS`.

Vérifier la structure de vos requêtes

Vérifiez si les requêtes capturées depuis ces états utilisent des sous-requêtes. Ce type de requête entraîne souvent de mauvaises performances, car la base de données compile les résultats en interne, puis les remplace à nouveau dans la requête pour restituer les données. Ce processus relève d'une étape supplémentaire pour la base de données. Bien souvent, cette étape peut entraîner de mauvaises performances dans des conditions de chargement hautement simultanées.

Vérifiez également si vos requêtes utilisent un grand nombre de clauses `ORDER BY` et `GROUP BY`. Au cours de telles opérations, la base de données doit souvent constituer le jeu de données entier en mémoire. Elle doit ensuite classer ou regrouper ces données de manière spécifique avant de les renvoyer au client.

Réglage d'Aurora MySQL avec les insights proactifs Amazon DevOps Guru

Les insights proactifs DevOps Guru détectent les conditions problématiques connues sur vos clusters de bases de données Aurora MySQL avant qu'ils se produisent. DevOps Guru peut effectuer les opérations suivantes :

- Éviter de nombreux problèmes courants liés aux bases de données en recoupant la configuration de votre base de données par rapport aux paramètres courants recommandés.
- Vous alerter face à des problèmes critiques dans votre flotte qui, s'ils ne sont pas vérifiés, peuvent entraîner des problèmes plus importants ultérieurement.
- Vous alerter face à des problèmes nouvellement découverts.

Chaque insight proactif contient une analyse de la cause du problème et des recommandations d'actions correctives.

Rubriques

- La longueur de la liste d’historique InnoDB a considérablement augmenté
- La base de données crée des tables temporaires sur le disque

La longueur de la liste d’historique InnoDB a considérablement augmenté

À compter de cette *`date`*, votre liste d’historique des modifications de ligne a considérablement augmenté, jusqu’à sa *`longueur`* sur l’*`instance de base de données`*. Cette augmentation affecte les performances d’arrêt des requêtes et des bases de données.

Rubriques

Versions de moteur prises en charge

Ces données d'insight sont prises en charge pour toutes les versions d'Aurora MySQL.

Contexte

Le système de transaction InnoDB gère le contrôle de simultanéité multiversion (MVCC). Lorsqu'une ligne est modifiée, la version antérieure à la modification des données en cours de modification est stockée sous la forme d'un enregistrement d'annulation dans un journal d'annulation. Chaque enregistrement d'annulation comporte une référence à son enregistrement de rétablissement précédent, formant ainsi une liste liée.

La liste d'historique InnoDB est une liste globale des journaux d'annulation des transactions validées. MySQL utilise cette liste d'historique pour purger les enregistrements et les pages de journal lorsque les transactions n'ont plus besoin de l'historique. La longueur de la liste d'historique est le nombre total de journaux d'annulation contenant des modifications dans la liste d'historique. Chaque journal contient une ou plusieurs modifications. Si la liste d'historique InnoDB devient trop grande, indiquant un grand nombre d'anciennes versions de lignes, les arrêts des bases de données et des requêtes deviennent plus lents.

Causes probables de ce problème

Les causes typiques d'une longue liste d'historique sont les suivantes :

- Transactions de longue durée, de lecture ou d'écriture
- Lourde charge d'écriture

Actions

Nous vous recommandons différentes actions en fonction des causes de votre insight.

Rubriques

Ne commencer aucune opération impliquant un arrêt de base de données tant que la liste d'historique InnoDB n'a pas diminué

Étant donné qu'une longue liste d'historique InnoDB ralentit les arrêts de base de données, réduisez la taille de la liste avant de lancer des opérations impliquant un arrêt de base de données. Ces opérations incluent les mises à niveau des versions majeures de base de données.

Identifier les transactions de longue durée et y mettre fin

Vous pouvez trouver les transactions de longue durée en exécutant la requête `information_schema.innodb_trx`.

> **Note**
>
> Assurez-vous également de rechercher les transactions de longue durée sur les réplicas en lecture.

Pour identifier les transactions de longue durée et y mettre fin

1. Dans votre client SQL, exécutez la requête suivante :

```
SELECT a.trx_id,
      a.trx_state,
      a.trx_started,
      TIMESTAMPDIFF(SECOND,a.trx_started, now()) as "Seconds Transaction Has Been
 Open",
      a.trx_rows_modified,
      b.USER,
      b.host,
      b.db,
      b.command,
      b.time,
```

```
      b.state
FROM  information_schema.innodb_trx a,
      information_schema.processlist b
WHERE a.trx_mysql_thread_id=b.id
  AND TIMESTAMPDIFF(SECOND,a.trx_started, now()) > 10
ORDER BY trx_started
```

2. Terminez chaque transaction de longue durée avec la procédure stockée mysql.rds_kill.

Identifier les hôtes et les utilisateurs principaux à l'aide de l'analyse des performances

Optimisez les transactions afin qu'un grand nombre de lignes modifiées soient immédiatement validées.

Métriques pertinentes

Les métriques suivantes sont liées à cet insight :

- `trx_rseg_history_len` : cette métrique de compteur peut être consultée dans Performance Insights, ainsi que dans la table `INFORMATION_SCHEMA.INNODB_METRICS`. Pour plus d'informations, consultez Tableau des métriques InnoDB INFORMATION_SCHEMA dans la documentation MySQL.
- `RollbackSegmentHistoryListLength` : cette métrique Amazon CloudWatch mesure les journaux d'annulation qui enregistrent les transactions validées avec des enregistrements marqués pour la suppression. Ces enregistrements sont planifiés pour être traités par l'opération de purge InnoDB. La métrique `trx_rseg_history_len` présente la même valeur que `RollbackSegmentHistoryListLength`.
- `PurgeBoundary` : le numéro de transaction jusqu'auquel la purge d'InnoDB est autorisée. Si cette métrique CloudWatch n'avance pas pendant de longues périodes, cela indique que la purge d'InnoDB est bloquée par des transactions de longue durée. Pour en savoir plus, vérifiez les transactions actives sur votre cluster de bases de données Aurora MySQL. Cette métrique est prise en charge pour Aurora MySQL versions 2.11 et ultérieures, et 3.08 et ultérieures.
- `PurgeFinishedPoint` : le numéro de transaction jusqu'auquel la purge d'InnoDB est effectuée. Cette métrique CloudWatch peut vous aider à examiner la rapidité de la purge d'InnoDB. Cette métrique est prise en charge pour Aurora MySQL versions 2.11 et ultérieures, et 3.08 et ultérieures.
- `TransactionAgeMaximum` : l'âge de la transaction en cours d'exécution active la plus ancienne. Cette métrique CloudWatch est disponible pour Aurora MySQL versions 3.08 et ultérieures.

- `TruncateFinishedPoint` : le numéro de transaction jusqu'auquel la troncature d'annulation est effectuée. Cette métrique CloudWatch est prise en charge pour Aurora MySQL versions 2.11 et ultérieures, et 3.08 et ultérieures.

Pour plus d'informations sur les métriques CloudWatch, consultez Métriques de niveau instance pour Amazon Aurora.

La base de données crée des tables temporaires sur le disque

Votre utilisation récente de tables temporaires sur disque a augmenté de manière significative, jusqu'à *`pourcentage`*. La base de données crée environ le *`nombre`* de tables temporaires par seconde. Cela peut avoir un impact sur les performances et augmenter les opérations sur le disque sur l'*`instance de base de données`*.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de ce problème
- Actions
- Métriques pertinentes

Versions de moteur prises en charge

Ces données d'insight sont prises en charge pour toutes les versions d'Aurora MySQL.

Contexte

Il est parfois nécessaire que le serveur MySQL crée une table temporaire interne lors du traitement d'une requête. Aurora MySQL peut contenir une table temporaire interne en mémoire, où elle peut être traitée par le moteur de stockage TempTable ou MEMORY, ou stockée sur disque par InnoDB. Pour plus d'informations, consultez Utilisation des tables temporaires internes dans MySQL dans le manuel de référence de MySQL.

Causes probables de ce problème

Une augmentation du nombre de tables temporaires sur disque indique l'utilisation de requêtes complexes. Si la mémoire configurée est insuffisante pour stocker des tables temporaires en

mémoire, Aurora MySQL crée les tables sur disque. Cela peut avoir un impact sur les performances et augmenter les opérations sur le disque.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre insight.

- Pour Aurora MySQL version 3, nous vous recommandons d'utiliser le moteur de stockage TempTable.
- Optimisez vos requêtes pour renvoyer moins de données en sélectionnant uniquement les colonnes nécessaires.

  Si vous activez le schéma de performance alors que tous les instruments `statement` sont activés et temporisés, vous pouvez effectuer une requête `SYS.statements_with_temp_tables` pour récupérer la liste des requêtes utilisant des tables temporaires. Pour plus d'informations, consultez Prérequis pour l'utilisation du schéma sys dans la documentation sur MySQL.
- Envisagez d'indexer les colonnes impliquées dans les opérations de tri et de regroupement.
- Réécrivez vos requêtes pour éviter les colonnes `BLOB` et `TEXT`. Ces colonnes utilisent toujours un disque.
- Réglez les paramètres de base de données suivants : `tmp_table_size` et `max_heap_table_size`.

  La valeur par défaut de ces paramètres est 16 Mio. Lorsque vous utilisez le moteur de stockage MEMORY pour des tables temporaires en mémoire, leur taille maximale est définie par la plus petite des valeurs `tmp_table_size` et `max_heap_table_size`. Lorsque cette taille maximale est atteinte, MySQL convertit automatiquement la table temporaire interne en mémoire en une table temporaire interne sur disque InnoDB. Pour plus d'informations, consultez Utiliser le moteur de stockage TempTable sur Amazon RDS pour MySQL et Amazon Aurora MySQL.

  > **Note**
  >
  > Lorsque vous créez explicitement des tables MEMORY avec CREATE TABLE, seule la variable `max_heap_table_size` détermine la taille maximale qu'une table peut prendre. Il n'y a pas non plus de conversion vers un format sur disque.

Métriques pertinentes

Les métriques suivantes d'analyse des performances sont liées à cet insight :

- Created_tmp_disk_tables
- Created_tmp_tables

Pour plus d'informations, consultez Created_tmp_disk_tables dans la documentation sur MySQL.

Requêtes parallèles pour Amazon Aurora MySQL

Cette rubrique décrit l'optimisation des performances via les requêtes parallèles pour Édition compatible avec Amazon Aurora MySQL. Cette fonction utilise un chemin de traitement spécial pour certaines requêtes à usage intensif de données, en tirant parti de l'architecture de stockage partagé d'Aurora. Les requêtes parallèles sont conçues pour les clusters de bases de données Aurora MySQL dont les tables contiennent des millions de lignes et dont le traitement des requêtes analytiques dure plusieurs minutes, voire plusieurs heures.

Rubriques

Présentation des requêtes parallèles pour Aurora MySQL

Une requête parallèle Aurora MySQL est une optimisation qui met en parallèle une partie des I/O et du calcul impliqués dans le traitement des requêtes à usage intensif de données. Les tâches qui sont mises en parallèle incluent la récupération des lignes à partir du stockage, l'extraction des valeurs des colonnes et la détermination des lignes répondant aux conditions définies dans la clause `WHERE` et dans les clauses JOIN. Ces tâches de gestion des gros volumes de données sont déléguées à différents nœuds de la couche de stockage distribué Aurora. Dans le jargon, on parle également d'optimisation de base de données par pushdown. Sans la fonction de requête parallèle, chaque requête envoie toutes les données analysées à un même nœud au sein du cluster Aurora MySQL (le nœud principal) et y effectue toutes les tâches de traitement nécessaires.

> **Tip**
>
> Le moteur de base de données PostgreSQL possède également une fonction appelée « requête parallèle ». Cette fonction n'est pas liée à la requête parallèle Aurora.

Lorsque la fonction de requête parallèle est activée, le moteur Aurora MySQL détermine automatiquement quand utiliser cette dernière, sans nécessiter de modifications SQL telles que les indicateurs ou les attributs de table. Dans les sections suivantes, vous découvrirez dans quels cas une requête parallèle est appliquée. Vous découvrirez également comment vous assurer qu'une requête parallèle est appliquée là où il faut.

> **Note**
>
> L'optimisation via des requêtes parallèles est destinée aux requêtes de longue durée dont le traitement prend entre quelques minutes et plusieurs heures. Aurora MySQL n'a généralement pas recours à cette fonction pour des requêtes peu coûteuses. Il n'effectue généralement pas d'optimisation de requête parallèle si une autre technique d'optimisation est plus logique, telle que la mise en cache de requêtes, la mise en cache de pools de mémoires tampons ou les recherches d'index. Si vous constatez que la requête parallèle n'est pas déclenchée lorsqu'elle le devrait, consultez Vérification des instructions utilisant les requêtes parallèles pour Aurora MySQL.

Rubriques

Avantages

Avec la requête parallèle, vous pouvez exécuter des requêtes analytiques à grand volume de données sur des tables Aurora MySQL. Dans de nombreux cas, l'amélioration des performances est significative par rapport au traitement traditionnel des requêtes.

Voici certains des avantages des requêtes parallèles :

- Amélioration des performances des I/O grâce à la mise en parallèle des requêtes de type « physical read » sur plusieurs nœuds de stockage.
- Trafic réseau réduit. Aurora ne transmet pas de pages de données entières des nœuds de stockage au nœud principal, et ne filtre pas les lignes et colonnes superflues par la suite. Au lieu de

cela, Aurora transmet des tuples compacts contenant uniquement les valeurs de colonne requises pour le jeu de résultats.

- Réduction de l'utilisation de l'UC au niveau du nœud principal grâce au traitement de la fonction « pushdown », au filtrage des lignes et à la projection des colonnes pour la clause `WHERE`.
- Sollicitation moindre de la mémoire au niveau du pool de mémoires tampons. Les pages traitées par la requête parallèle ne sont pas ajoutées au pool de mémoires tampons. Cette approche réduit les risques qu'une analyse à grand volume de données expulse les données fréquemment utilisées du pool de mémoires tampons.
- Réduction potentielle de la duplication des données dans le pipeline ETL (extract, transform, load) grâce à l'exécution, plus pratique, des requêtes analytiques de longue durée au niveau des données existantes.

Architecture

La fonction de requêtes parallèles repose sur les principes architecturaux clés d'Aurora MySQL : découplage du moteur de base de données du sous-système de stockage et réduction du trafic via la rationalisation des protocoles de communication. Aurora MySQL utilise ces techniques pour accélérer les opérations impliquant un grand nombre d'écritures, telles que le traitement des journaux redo. Les requêtes parallèles appliquent les mêmes principes aux opérations de lecture.

> **Note**
>
> L'architecture des requêtes parallèles Aurora MySQL diffère de celle des fonctions dont le nom est similaire dans les autres systèmes de base de données. Les requêtes parallèles Aurora MySQL n'impliquent pas de multitraitement symétrique et ne dépendent donc pas de la capacité d'UC du serveur de base de données. Le traitement parallèle intervient dans la couche de stockage, indépendamment du serveur Aurora MySQL qui joue le rôle de coordinateur de requêtes.

Par défaut, sans requête parallèle, le traitement d'une requête Aurora inclut la transmission des données brutes à un même nœud au sein du cluster Aurora (nœud principal). Aurora exécute ensuite toutes les autres tâches nécessaires pour cette requête dans un seul thread sur de ce nœud unique. Avec une requête parallèle, une grande partie des tâches impliquant un usage intensif de l'UC et un grand nombre d'I/O est déléguée aux nœuds de la couche de stockage. Seules les lignes compactes du jeu de résultats sont renvoyées au nœud principal, avec les lignes déjà filtrées et les valeurs de

colonne déjà extraites et transformées. L'amélioration des performances découle de la réduction du trafic réseau, d'une utilisation moindre de l'UC au niveau du nœud principal et de la mise en parallèle des I/O entre les nœuds de stockage. Le volume d'I/O parallèles, de filtrage et de projection n'est pas lié au nombre d'instances de base de données du cluster Aurora qui exécute la requête.

Prérequis

Pour pouvoir utiliser toutes les fonctionnalités des requêtes parallèles, vous avez besoin d'un cluster de bases de données Aurora MySQL qui exécute la version 2.09 ou une version ultérieure. Si vous avez déjà un cluster que vous souhaitez utiliser avec une requête parallèle, vous pouvez le mettre à niveau vers une version compatible et activer la requête parallèle par la suite. Dans ce cas, assurez-vous de suivre la procédure de mise à niveau décrite dans Considérations relatives aux mises à niveau pour les requêtes parallèles, car les noms de paramètre de configuration et les valeurs par défaut sont différents dans ces versions plus récentes.

Les instances de base de données de votre cluster doivent utiliser les classes d'instance `db.r*`.

Assurez-vous que l'optimisation de jointure par hachage est activée pour votre cluster. Pour savoir comment procéder, consultez Activation de la jointure par hachage pour les clusters de requête parallèle.

Pour personnaliser des paramètres tels que `aurora_parallel_query` et `aurora_disable_hash_join`, vous devez disposer d'un groupe de paramètres personnalisés que vous utilisez avec votre cluster. Vous pouvez spécifier ces paramètres individuellement pour chaque instance de base de données à l'aide d'un groupe de paramètres de base de données. Toutefois, nous vous recommandons de les spécifier dans un groupe de paramètres de cluster de bases de données. De cette façon, toutes les instances de base de données de votre cluster héritent des mêmes choix pour ces paramètres.

Limitations

Les limitations suivantes s'appliquent à la fonction de requête parallèle :

- La requête parallèle n'est pas prise en charge avec la configuration de stockage du cluster de bases de données Aurora I/O-Optimized.
- Vous ne pouvez pas utiliser les requêtes parallèles avec les classes d'instance db.t2 ou db.t3. Cette limite s'applique même si vous demandez une requête parallèle en utilisant la variable de session `aurora_pq_force`.

- La requête parallèle ne s'applique pas aux tables utilisant les formats de ligne `COMPRESSED` ou `REDUNDANT`. Utilisez les formats de ligne `COMPACT` ou `DYNAMIC` pour les tables que vous prévoyez d'utiliser avec une requête parallèle.
- Aurora utilise un algorithme basé sur les coûts pour déterminer si le mécanisme de requête parallèle doit être utilisé pour chaque instruction SQL. L'utilisation de certaines constructions SQL dans une instruction peut empêcher la requête parallèle ou rendre celle-ci peu probable pour cette instruction. Pour plus d'informations sur la compatibilité des constructions SQL avec une requête parallèle, consultez Constructions SQL pour les requêtes parallèles dans Aurora MySQL.
- Chaque instance de base de données Aurora ne peut exécuter qu'un nombre spécifique de sessions de requêtes parallèles à la fois. Si une requête inclut plusieurs parties utilisant une requête parallèle, telles que des sous-requêtes, des clauses JOIN ou des opérateurs `UNION`, ces expressions sont exécutées les unes à la suite des autres. L'instruction n'est considérée que comme une seule session de requête parallèle. Vous pouvez surveiller le nombre de sessions actives via les variables d'état des requêtes parallèles. Pour vérifier le nombre limite de sessions simultanées pour une instance de base de données déterminée, interrogez la variable de statut `Aurora_pq_max_concurrent_requests`.
- La fonction de requêtes parallèles est disponible dans toutes les Régions AWS prises en charge par Aurora. Pour la plupart des Régions AWS, la version minimale requise d'Aurora MySQL pour utiliser les requêtes parallèles est 2.09.
- La requête parallèle est conçue pour améliorer les performances des requêtes gourmandes en données. Elle n'est pas conçue pour les requêtes courtes.
- Nous vous recommandons d'utiliser des nœuds de lecteur pour les instructions SELECT, en particulier pour les instructions gourmandes en données.

Coûts d'E/S liés aux requêtes parallèles

Si votre cluster Aurora MySQL utilise une requête parallèle, vous pouvez voir une augmentation des valeurs d'`VolumeReadIOPS`. Les requêtes parallèles n'utilisent pas le pool de mémoires tampons. Ainsi, bien que les requêtes soient rapides, ce traitement optimisé peut entraîner une augmentation des opérations de lecture et des frais associés.

Les coûts d'E/S des requêtes parallèles pour votre requête sont mesurés au niveau de la couche de stockage et seront identiques ou supérieurs si la requête parallèle est activée. Votre avantage réside dans l'amélioration des performances des requêtes. Les coûts d'E/S potentiellement plus élevés liés aux requêtes parallèles peuvent s'expliquer par deux raisons :

- Même si certaines données d'une table se trouvent dans le pool de mémoire tampon, la requête parallèle nécessite que toutes les données soient analysées au niveau de la couche de stockage, ce qui entraîne des coûts d'E/S.
- L'exécution d'une requête parallèle ne réchauffe pas le pool de mémoire tampon. Par conséquent, les exécutions consécutives de la même requête parallèle entraînent le coût intégral des E/S.

Création d'un cluster de bases de données de requêtes parallèles dans Aurora MySQL

Pour créer un cluster Aurora MySQL compatible avec les requêtes parallèles, pour y ajouter des instances ou pour effectuer d'autres opérations administratives, vous devez utiliser les mêmes techniques d'AWS Management Console et d'AWS CLI que pour les autres clusters Aurora MySQL. Vous pouvez créer un cluster pour qu'il soit compatible avec les requêtes parallèles. Vous pouvez également créer un cluster de bases de données pour qu'il fonctionne avec les requêtes parallèles en restaurant un instantané de cluster de bases de données Aurora compatible avec MySQL 5.6. Si vous ne savez pas comment créer un cluster Aurora MySQL, vous trouverez les prérequis et toute information utile dans Création d'un cluster de bases de données Amazon Aurora.

Lorsque vous choisissez une version de moteur Aurora MySQL, nous vous recommandons de choisir la version la plus récente disponible. Actuellement, toutes les versions disponibles d'Aurora MySQL prennent en charge les requêtes parallèles. Vous disposez d'une plus grande flexibilité pour activer et désactiver les requêtes parallèles ou pour les utiliser avec des clusters existants sur les versions les plus récentes.

Que vous choisissiez de créer un cluster ou de restaurer un instantané, les techniques d'ajout de nouvelles instances de base de données sont les mêmes techniques que pour les autres clusters Aurora MySQL.

Vous pouvez créer un cluster de requêtes parallèles à l'aide de la console Amazon RDS ou de l'AWS CLI.

Table des matières

Création d'un cluster compatible avec les requêtes parallèles à l'aide de la console

Pour créer un cluster compatible avec les requêtes parallèles à l'aide de la console, procédez comme décrit ci-dessous.

Pour créer un cluster compatible avec les requêtes parallèles via AWS Management Console

1. Suivez la procédure AWS Management Console générale décrite dans Création d'un cluster de bases de données Amazon Aurora.
2. Pour Type de moteur, choisissez Aurora MySQL.
3. Pour Configuration supplémentaire, choisissez un groupe de paramètres que vous avez créé pour Groupe de paramètres de cluster de bases de données. L'utilisation d'un groupe de paramètres personnalisés de ce type est requise pour Aurora MySQL 2.09 et versions ultérieures. Dans votre groupe de paramètres de cluster de bases de données, spécifiez les paramètres `aurora_parallel_query=ON` et `aurora_disable_hash_join=OFF`. Cela active la requête parallèle pour le cluster, ainsi que l'optimisation de jointure par hachage qui fonctionne en combinaison avec la requête parallèle.

Pour vérifier qu'un nouveau cluster est compatible avec les requêtes parallèles

1. Créez un cluster à l'aide de la technique précédente.
2. (Pour Aurora MySQL version 2 ou 3) Vérifiez que le paramètre de configuration `aurora_parallel_query` est true.

```
mysql> select @@aurora_parallel_query;
+--------------------------+
| @@aurora_parallel_query |
+--------------------------+
|                        1 |
+--------------------------+
```

3. (Pour Aurora MySQL version 2) Vérifiez que le paramètre `aurora_disable_hash_join` est false.

```
mysql> select @@aurora_disable_hash_join;
+----------------------------+
| @@aurora_disable_hash_join |
+----------------------------+
|                          0 |
```

```
+----------------------------+
```

4. Avec certaines tables volumineuses et certaines requêtes à grand volume de données, vérifiez les plans de requête pour confirmer que certaines de vos requêtes utilisent l'optimisation de requête parallèle. Pour ce faire, suivez la procédure décrite dans Vérification des instructions utilisant les requêtes parallèles pour Aurora MySQL.

Création d'un cluster compatible avec les requêtes parallèles à l'aide de l'interface de ligne de commande (CLI)

Pour créer un cluster compatible avec les requêtes parallèles à l'aide de l'interface de ligne de commande (CLI), procédez comme décrit ci-dessous.

Pour créer un cluster compatible avec les requêtes parallèles via AWS CLI

1. (Facultatif) Vérifiez quelles versions d'Aurora MySQL sont compatibles avec les clusters de requête parallèle. Pour ce faire, utilisez la commande `describe-db-engine-versions` et vérifiez la valeur du champ `SupportsParallelQuery`. Pour obtenir un exemple, consultez Vérification de la compatibilité de version Aurora MySQL pour une requête parallèle.
2. (Facultatif) Créez un groupe de paramètres de cluster de bases de données personnalisés avec les paramètres `aurora_parallel_query=ON` et `aurora_disable_hash_join=OFF`. Utilisez des commandes telles que les suivantes.

   ```
   aws rds create-db-cluster-parameter-group --db-parameter-group-family aurora-
   mysql8.0 --db-cluster-parameter-group-name pq-enabled-80-compatible
   aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name pq-
   enabled-80-compatible \
     --parameters
    ParameterName=aurora_parallel_query,ParameterValue=ON,ApplyMethod=pending-reboot
   aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name pq-
   enabled-80-compatible \
     --parameters
    ParameterName=aurora_disable_hash_join,ParameterValue=OFF,ApplyMethod=pending-
   reboot
   ```

   Si vous effectuez cette étape, spécifiez l'option `--db-cluster-parameter-group-name` *`my_cluster_parameter_group`* dans l'instruction `create-db-cluster` suivante. Indiquez le nom de votre propre groupe de paramètres. Si vous omettez cette étape, vous devrez créer

le groupe de paramètres et l'associer ultérieurement au cluster, comme décrit dans Activation et désactivation des requêtes parallèles dans Aurora MySQL.

3. Suivez la procédure AWS CLI générale décrite dans Création d'un cluster de bases de données Amazon Aurora.
4. Spécifiez les options suivantes :

   - Pour l'option `--engine`, utilisez `aurora-mysql`. Ces valeurs produisent des clusters de requête parallèle compatibles avec MySQL 5.7 ou 8.0.
   - Pour l'option `--db-cluster-parameter-group-name`, spécifiez le nom d'un groupe de paramètres de cluster de bases de données que vous avez créé et spécifiez la valeur du paramètre `aurora_parallel_query=ON`. Si vous omettez cette option, vous pouvez créer le cluster avec un groupe de paramètres par défaut et le modifier ultérieurement pour utiliser un groupe de paramètres personnalisés de ce type.
   - Pour l'option `--engine-version`, utilisez une version d'Aurora MySQL compatible avec la requête parallèle. Utilisez la procédure décrite dans Optimisation des requêtes parallèles dans Aurora MySQL pour obtenir une liste des versions si nécessaire.

     L'exemple de code suivant illustre la marche à suivre. Indiquez votre propre valeur pour chacune des variables d'environnement telles que *`$CLUSTER_ID`*. Cet exemple spécifie également l'option `--manage-master-user-password` permettant de générer le mot de passe de l'utilisateur principal et de le gérer dans Secrets Manager. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager. Vous pouvez également utiliser l'option `--master-password` pour spécifier et gérer vous-même le mot de passe.

```
aws rds create-db-cluster --db-cluster-identifier $CLUSTER_ID \
  --engine aurora-mysql --engine-version 8.0.mysql_aurora.3.04.1 \
  --master-username $MASTER_USER_ID --manage-master-user-password \
  --db-cluster-parameter-group-name $CUSTOM_CLUSTER_PARAM_GROUP

aws rds create-db-instance --db-instance-identifier ${INSTANCE_ID}-1 \
  --engine same_value_as_in_create_cluster_command \
  --db-cluster-identifier $CLUSTER_ID --db-instance-class $INSTANCE_CLASS
```

5. Vérifiez que la fonction de requête parallèle est disponible pour le cluster que vous avez créé ou restauré.

Vérifiez que le paramètre de configuration `aurora_parallel_query` existe. Si ce paramètre a pour valeur 1, la requête parallèle est prête à être utilisée. Si ce paramètre a pour valeur 0, définissez-la sur 1 afin de pouvoir utiliser la requête parallèle. Dans tous les cas, le cluster est capable d'effectuer des requêtes parallèles.

```
mysql> select @@aurora_parallel_query;
+------------------------+
| @@aurora_parallel_query|
+------------------------+
|                      1 |
+------------------------+
```

Pour restaurer un instantané de cluster compatible avec les requêtes parallèles via AWS CLI

1. Vérifiez quelles sont les versions Aurora MySQL compatibles avec les clusters de requête parallèle. Pour ce faire, utilisez la commande `describe-db-engine-versions` et vérifiez la valeur du champ `SupportsParallelQuery`. Pour obtenir un exemple, consultez Vérification de la compatibilité de version Aurora MySQL pour une requête parallèle. Choisissez la version à utiliser pour le cluster restauré.
2. Recherchez un instantané de cluster compatible avec Aurora MySQL.
3. Suivez la procédure AWS CLI générale décrite dans Restauration à partir d'un instantané de cluster de bases de données.

```
aws rds restore-db-cluster-from-snapshot \
  --db-cluster-identifier mynewdbcluster \
  --snapshot-identifier mydbclustersnapshot \
  --engine aurora-mysql
```

4. Vérifiez que la fonction de requête parallèle est disponible pour le cluster que vous avez créé ou restauré. Utilisez la même procédure de vérification que dans Création d'un cluster compatible avec les requêtes parallèles à l'aide de l'interface de ligne de commande (CLI).

Activation et désactivation des requêtes parallèles dans Aurora MySQL

Lorsque la fonction de requête parallèle est activée, Aurora MySQL détermine quand l'utiliser pour chaque requête lors de l'exécution. En cas de clauses JOIN, de clauses UNION, de sous-requêtes, et ainsi de suite, Aurora MySQL détermine quand utiliser les requêtes parallèles pour chaque bloc de

requêtes lors de l'exécution. Pour plus d'informations, consultez Vérification des instructions utilisant les requêtes parallèles pour Aurora MySQL et Constructions SQL pour les requêtes parallèles dans Aurora MySQL.

Vous pouvez activer ou désactiver de manière dynamique les requêtes parallèles au niveau global et au niveau de la session pour une instance de base de données via l'option aurora_parallel_query. Vous pouvez modifier le paramètre `aurora_parallel_query` dans votre groupe de cluster de bases de données pour activer ou désactiver la requête parallèle par défaut.

```
mysql> select @@aurora_parallel_query;
+------------------------+
| @@aurora_parallel_query|
+------------------------+
|                      1 |
+------------------------+
```

Pour basculer le paramètre `aurora_parallel_query` au niveau de la session, utilisez les méthodes standard pour modifier un paramètre de configuration client. Par exemple, vous pouvez pour ce faire utiliser la ligne de commande `mysql` ou une application JDBC ou ODBC. La commande du client MySQL standard est `set session aurora_parallel_query = {'ON'/'OFF'}`. Vous pouvez également ajouter le paramètre au niveau de la session à la configuration JDBC ou dans le code de l'application pour activer ou désactiver les requêtes parallèles de manière dynamique.

Vous pouvez modifier définitivement le paramètre `aurora_parallel_query` pour une instance de base de données spécifique ou pour l'ensemble de votre cluster. Si vous spécifiez la valeur de paramètre dans un groupe de paramètres de base de données, cette valeur s'applique uniquement à une instance de base de données spécifique de votre cluster. Si vous spécifiez la valeur de paramètre dans un groupe de paramètres de cluster de bases de données, toutes les instances de base de données du cluster héritent du même paramètre. Pour activer le paramètre `aurora_parallel_query`, utilisez les techniques applicables aux groupes de paramètres, comme décrit dans Groupes de paramètres pour Amazon Aurora. Procédez comme suit :

1. Créez un groupe de paramètres de cluster personnalisés (recommandé) ou un groupe de paramètres de base de données personnalisés.
2. Dans ce groupe de paramètres, mettez à jour `parallel_query` avec la valeur souhaitée.
3. Selon que vous avez créé un groupe de paramètres de cluster de bases de données ou un groupe de paramètres de base de données, attachez le groupe de paramètres à votre cluster Aurora ou

aux instances de base de données spécifiques dans lesquelles vous prévoyez d'utiliser la fonction de requête parallèle.

> **Tip**
>
> Comme `aurora_parallel_query` est un paramètre dynamique, il n'est pas nécessaire de redémarrer le cluster après avoir modifié ce paramètre. Cependant, toutes les connexions qui utilisaient la requête parallèle avant de basculer l'option continueront à le faire jusqu'à ce que la connexion soit fermée ou que l'instance soit redémarrée.

Pour modifier le paramètre de requête parallèle, utilisez l'opération d'API ModifyDBClusterParameterGroup ou ModifyDBParameterGroup, ou bien AWS Management Console.

Vous pouvez activer la jointure par hachage pour les clusters de requêtes parallèles, activer et désactiver les requêtes parallèles à l'aide de la console Amazon RDS ou de l'AWS CLI, et remplacer l'optimiseur de requêtes parallèles.

Table des matières

Activation de la jointure par hachage pour les clusters de requête parallèle

Les requêtes parallèles sont généralement utilisées pour les types de requêtes gourmandes en ressources qui tirent parti de l'optimisation de la jointure par hachage. Ainsi, il est utile de s'assurer que les jointures par hachage sont activées pour les clusters où vous envisagez d'utiliser une requête parallèle. Pour plus d'informations sur la façon d'utiliser les jointures par hachage efficacement, consultez Optimisation des requêtes de jointure MySQL Aurora volumineuses avec des jointures de hachage.

Activation et désactivation de la requête parallèle à l'aide de la console

Vous pouvez activer ou désactiver la requête parallèle au niveau de l'instance de base de données ou du cluster de bases de données en utilisant les groupes de paramètres.

Pour activer ou désactiver la requête parallèle pour un cluster de bases de données avec la AWS Management Console

1. Créez un groupe personnalisé de paramètres, comme décrit dans Groupes de paramètres pour Amazon Aurora.
2. Mettez à jour aurora_parallel_query sur 1 (activé) ou 0 (désactivé). Pour les clusters pour lesquels la fonction de requête parallèle est activée, aurora_parallel_query est désactivé par défaut.
3. Si vous utilisez un groupe de paramètres de cluster personnalisés, attachez-le au cluster de bases de données Aurora où vous prévoyez d'utiliser la fonction de requête parallèle. Si vous utilisez un groupe de paramètres de base de données personnalisés, attachez-le à une ou plusieurs instances de base de données du cluster. Nous vous recommandons d'utiliser un groupe de paramètres de cluster. Cela garantit que toutes les instances de base de données du cluster ont les mêmes paramètres pour la requête parallèle et les fonctions associées telles que la jointure par hachage.

Activation et désactivation de la requête parallèle à l'aide de l'interface de ligne de commande (CLI)

Vous pouvez modifier le paramètre de requête parallèle via la commande `modify-db-cluster-parameter-group` ou `modify-db-parameter-group` Choisissez la commande appropriée selon que vous spécifiez la valeur de `aurora_parallel_query` via un groupe de paramètres de cluster de bases de données ou un groupe de paramètres de base de données.

Pour activer ou désactiver la requête parallèle pour un cluster de bases de données avec l'interface de ligne de commande (CLI)

- Pour modifier le paramètre de requête parallèle, utilisez la commande `modify-db-cluster-parameter-group`. Utilisez une commande telle que la suivante. Indiquez le nom approprié pour votre propre groupe de paramètres personnalisé. Indiquez `ON` ou `OFF` pour la partie `ParameterValue` de l'option `--parameters`.

```
$ aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-
name cluster_param_group_name \
  --parameters
 ParameterName=aurora_parallel_query,ParameterValue=ON,ApplyMethod=pending-reboot
{
```

```
    "DBClusterParameterGroupName": "cluster_param_group_name"
}

aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-
name cluster_param_group_name \
  --parameters ParameterName=aurora_pq,ParameterValue=ON,ApplyMethod=pending-reboot
```

Vous pouvez également activer ou désactiver les requêtes parallèles au niveau de la session, par exemple via la ligne de commande `mysql` ou au sein d'une application JDBC ou ODBC. Pour ce faire, utilisez les méthodes habituelles qui s'appliquent à la modification d'un paramètre de configuration client. Par exemple, la commande pour le client MySQL standard est `set session aurora_parallel_query = {'ON'/'OFF'}` pour Aurora MySQL.

Vous pouvez également ajouter le paramètre au niveau de la session à la configuration JDBC ou dans le code de l'application pour activer ou désactiver les requêtes parallèles de manière dynamique.

Remplacer l'optimiseur de requêtes parallèles

Vous pouvez utiliser la variable de session `aurora_pq_force` pour remplacer l'optimiseur de requête parallèle et demander une requête parallèle pour chaque requête. Nous vous recommandons de ne le faire qu'à des fins de test. L'exemple suivant montre comment utiliser `aurora_pq_force` dans une session.

```
set SESSION aurora_parallel_query = ON;
set SESSION aurora_pq_force = ON;
```

Pour désactiver le remplacement, procédez comme suit :

```
set SESSION aurora_pq_force = OFF;
```

Optimisation des requêtes parallèles dans Aurora MySQL

Pour optimiser votre cluster de bases de données pour les requêtes parallèles, identifiez quels clusters bénéficieraient de requêtes parallèles et déterminez s'il convient de procéder à une mise à niveau pour les requêtes parallèles. Ajustez ensuite votre charge de travail et créez des objets de schéma pour les requêtes parallèles.

Table des matières

Planification d’un cluster de requête parallèle

La planification d’un cluster de bases de données dont les requêtes parallèles sont activées nécessite quelques choix. Il s’agit notamment d’effectuer des étapes de configuration (création ou restauration d’un cluster Aurora MySQL complet) et de décider de l’étendue de l’activation des requêtes parallèles sur votre cluster de bases de données.

Considérez les éléments suivants dans le cadre de la planification :

- Si vous utilisez une version d’Aurora MySQL compatible avec MySQL 5.7, vous devez créer un cluster provisionné. Ensuite, vous activez la requête parallèle en utilisant le paramètre `aurora_parallel_query`.

  Si vous disposez déjà d’un cluster Aurora MySQL existant, vous n’avez pas besoin d’en créer un autre pour utiliser une requête parallèle. Vous pouvez associer votre cluster ou des instances de base de données spécifiques du cluster à un groupe de paramètres pour lequel `aurora_parallel_query` est activé. De ce fait, vous pouvez réduire le temps et les efforts nécessaires pour configurer les données pertinentes à utiliser avec une requête parallèle.
- Planifiez les tables volumineuses que vous devez réorganiser afin d’être en mesure d’utiliser la requête parallèle lors de l’accès à celles-ci. Vous devrez peut-être créer de nouvelles versions de certaines tables volumineuses où la requête parallèle est utile. Par exemple, vous devrez peut-être supprimer les index de recherche en texte intégral. Pour en savoir plus, consultez Création d’objets de schéma pour tirer parti des requêtes parallèles.

Vérification de la compatibilité de version Aurora MySQL pour une requête parallèle

Pour vérifier quelles sont les versions d’Aurora MySQL compatibles avec les clusters de requête parallèle, utilisez la commande de l’AWS CLI `describe-db-engine-versions` et vérifiez la valeur

du champ `SupportsParallelQuery`. L'exemple de code suivant montre comment vérifier les combinaisons qui sont accessibles aux clusters compatibles avec les requêtes parallèles dans une région AWS déterminée. Assurez-vous de spécifier la chaîne de paramètres `--query` complète sur une seule ligne.

```
aws rds describe-db-engine-versions --region us-east-1 --engine aurora-mysql \
--query '*[]|[?SupportsParallelQuery == `true`].[EngineVersion]' --output text
```

Les commandes précédentes génèrent une sortie similaire à la sortie suivante : La sortie peut varier en fonction des versions d'Aurora MySQL disponibles dans la région AWS spécifiée.

```
5.7.mysql_aurora.2.11.1
5.7.mysql_aurora.2.11.2
5.7.mysql_aurora.2.11.3
5.7.mysql_aurora.2.11.4
5.7.mysql_aurora.2.11.5
5.7.mysql_aurora.2.11.6
5.7.mysql_aurora.2.12.0
5.7.mysql_aurora.2.12.1
5.7.mysql_aurora.2.12.2
5.7.mysql_aurora.2.12.3
5.7.mysql_aurora.2.12.4
8.0.mysql_aurora.3.04.0
8.0.mysql_aurora.3.04.1
8.0.mysql_aurora.3.04.2
8.0.mysql_aurora.3.04.3
8.0.mysql_aurora.3.05.2
8.0.mysql_aurora.3.06.0
8.0.mysql_aurora.3.06.1
8.0.mysql_aurora.3.07.0
8.0.mysql_aurora.3.07.1
```

Une fois que vous avez commencé à utiliser une requête parallèle avec un cluster, vous pouvez surveiller les performances et supprimer les obstacles à l'utilisation de la requête parallèle. Pour obtenir ces instructions, consultez Réglage de performances pour les requêtes parallèles.

Considérations relatives aux mises à niveau pour les requêtes parallèles

Selon les versions d'origine et de destination lorsque vous mettez à niveau un cluster de requêtes parallèles, vous trouverez des améliorations dans les types de requêtes que la requête parallèle peut optimiser. Vous constaterez également que vous n'avez pas besoin de spécifier un paramètre

de mode de moteur spécial pour les requêtes parallèles. Les sections suivantes expliquent les considérations à prendre en compte lors de la mise à niveau d'un cluster sur lequel la requête parallèle est activée.

Mise à niveau des clusters de requêtes parallèles vers Aurora MySQL version 3

Plusieurs instructions, clauses et types de données SQL offrent une prise en charge des requêtes parallèles nouvelles ou améliorées à partir d'Aurora MySQL version 3. Lorsque vous effectuez une mise à niveau à partir d'une version antérieure à la version 3, vérifiez si des requêtes supplémentaires peuvent bénéficier d'optimisations des requêtes parallèles. Pour plus d'informations sur ces améliorations des requêtes parallèles, consultez Types de données de colonne, Tables partitionnées et Fonctions d'agrégation et clauses GROUP BY et HAVING.

Si vous mettez à niveau un cluster de requête parallèle à partir d'Aurora MySQL 2.08 ou version antérieure, découvrez également les modifications apportées à l'activation d'une requête parallèle. Pour ce faire, lisez Mise à niveau vers Aurora MySQL 2.09 et versions ultérieures.

Dans Aurora MySQL version 3, l'optimisation de jointure par hachage est activée par défaut. L'option de configuration `aurora_disable_hash_join` des versions antérieures n'est pas utilisée.

Mise à niveau vers Aurora MySQL 2.09 et versions ultérieures

Dans Aurora MySQL 2.09 et versions ultérieures, la requête parallèle fonctionne pour les clusters alloués et ne nécessite pas le paramètre de mode de moteur `parallelquery`. Ainsi, vous n'avez pas besoin de créer un nouveau cluster ou de procéder à une restauration à partir d'un instantané existant pour utiliser une requête parallèle avec ces versions. Vous pouvez utiliser les procédures de mise à niveau décrites dans Mise à niveau de la version mineure ou du niveau de correctif d'un cluster de bases de données Aurora MySQL pour mettre à niveau votre cluster vers une de ces versions. Vous pouvez mettre à niveau un cluster plus ancien, qu'il s'agisse d'un cluster de requête parallèle ou d'un cluster alloué. Pour réduire le nombre de choix dans le menu Engine version (Version du moteur), vous pouvez choisir Show versions that support the parallel query feature (Afficher les versions prenant en charge la fonction de requête parallèle) pour filtrer les entrées de ce menu. Choisissez ensuite Aurora MySQL 2.09 ou version ultérieure.

Après avoir mis à niveau un cluster de requête parallèle antérieur vers Aurora MySQL 2.09 ou version ultérieure, vous activez la requête parallèle dans le cluster mis à niveau. La requête parallèle est désactivée par défaut dans ces versions. La procédure pour l'activer est différente. L'optimisation de jointure par hachage est également désactivée par défaut et doit être activée séparément. Ainsi, veillez à activer à nouveau ces paramètres après la mise à niveau. Pour obtenir des instructions à ce

sujet, consultez Activation et désactivation des requêtes parallèles dans Aurora MySQL et Activation de la jointure par hachage pour les clusters de requête parallèle.

En particulier, vous activez la requête parallèle en utilisant les paramètres de configuration `aurora_parallel_query=ON` et `aurora_disable_hash_join=OFF` au lieu de `aurora_pq_supported` et `aurora_pq`. Les paramètres `aurora_pq_supported` et `aurora_pq` sont obsolètes dans les versions d'Aurora MySQL les plus récentes.

Dans le cluster mis à niveau, l'attribut `EngineMode` a la valeur `provisioned` au lieu de `parallelquery`. Pour vérifier si la requête parallèle est disponible pour une version de moteur spécifiée, vous devez vérifier la valeur du champ `SupportsParallelQuery` dans la sortie de la commande `describe-db-engine-versions` AWS CLI. Dans les versions antérieures d'Aurora MySQL, vous avez vérifié la présence de `parallelquery` dans la liste `SupportedEngineModes`.

Après la mise à niveau vers Aurora MySQL 2.09 ou version ultérieure, vous pouvez profiter des fonctionnalités suivantes. Ces fonctions ne sont pas disponibles pour les clusters de requête parallèle exécutant des versions plus anciennes d'Aurora MySQL.

- Analyse des performances. Pour plus d'informations, consultez Surveillance de la charge de la base de données avec Performance Insights sur .
- Retour sur trace. Pour plus d'informations, consultez Retour en arrière d'un cluster de bases de données Aurora.
- Arrêt et démarrage du cluster. Pour plus d'informations, consultez Arrêt et démarrage d'un cluster de bases de données Amazon Aurora.

Réglage de performances pour les requêtes parallèles

Pour gérer les performances d'une charge de travail avec la fonction de requête parallèle, assurez-vous que cette dernière est utilisée pour les requêtes pour lesquelles cette optimisation est la plus utile.

Pour ce faire, vous pouvez effectuer les opérations suivantes :

- Assurez-vous que vos tables les plus volumineuses sont compatibles avec la requête parallèle. Vous pouvez modifier les propriétés des tables ou recréer certaines tables afin que les requêtes pour ces tables puissent tirer parti de l'optimisation de requête parallèle. Pour savoir comment procéder, consultez Création d'objets de schéma pour tirer parti des requêtes parallèles.

- Surveillez les requêtes qui utilisent la fonction de requête parallèle. Pour savoir comment procéder, consultez Surveillance des requêtes parallèles pour Aurora MySQL.
- Vérifiez que la requête parallèle est utilisée pour les requêtes à grand volume de données et de longue durée, et avec le niveau de concurrence approprié pour votre charge de travail. Pour savoir comment procéder, consultez Vérification des instructions utilisant les requêtes parallèles pour Aurora MySQL.
- Affinez votre code SQL pour activer la requête parallèle et l'appliquer aux requêtes de votre choix. Pour savoir comment procéder, consultez Constructions SQL pour les requêtes parallèles dans Aurora MySQL.

Création d'objets de schéma pour tirer parti des requêtes parallèles

Avant de créer ou de modifier des tables que vous prévoyez d'utiliser pour une requête parallèle, veillez à vous familiariser avec les exigences décrites dans Prérequis et Limitations.

La fonction de requête parallèle nécessitant que les tables utilisent le paramètre `ROW_FORMAT=Compact` ou `ROW_FORMAT=Dynamic`, vérifiez si des modifications ont été apportées à l'option de configuration `INNODB_FILE_FORMAT` dans les paramètres de configuration Aurora. Exécutez l'instruction `SHOW TABLE STATUS` pour confirmer le format de ligne de toutes les tables d'une base de données.

Avant de modifier votre schéma pour permettre à la requête parallèle de fonctionner avec d'autres tables, assurez-vous de procéder à des tests. Vos tests doivent confirmer si une requête parallèle entraîne une augmentation nette des performances pour ces tables. Assurez-vous également que les exigences en matière de schéma pour les requêtes parallèles coïncident avec vos objectifs.

Par exemple, avant de passer de `ROW_FORMAT=Compressed` à `ROW_FORMAT=Compact` ou `ROW_FORMAT=Dynamic`, testez les performances des charges de travail pour les tables d'origine et les nouvelles tables. Tenez également compte des autres effets potentiels tels que l'augmentation du volume de données.

Vérification des instructions utilisant les requêtes parallèles pour Aurora MySQL

En règle générale, aucune action spécifique n'est requise de votre part pour tirer parti des requêtes parallèles. Lorsqu'une requête est compatible avec la fonction de requête parallèle, l'optimiseur détermine automatiquement dans quel cas utiliser cette fonction pour chaque requête.

Si vous effectuez des expérimentations dans un environnement de développement ou de test, vous constaterez peut-être que la fonction de requête parallèle n'est pas utilisée, car vos tables ne contiennent pas assez de lignes ou pas assez de données. Les données associées à la table, notamment celles que vous avez créées récemment pour réaliser ces expérimentations, peuvent également se trouver entièrement dans un pool de mémoires tampons.

À mesure que vous surveillez ou ajustez les performances de vos clusters, veillez à déterminer si les requêtes parallèles sont déclenchées dans les contextes appropriés. Pour tirer parti de cette fonction, vous pouvez ajuster le schéma de base de données, les paramètres, les requêtes SQL, voire la topologie du cluster et les paramètres de connexion de l'application.

Pour vérifier si une requête utilise la fonction de requête parallèle, consultez le plan de requête (ou « plan d'explication ») en exécutant l'instruction EXPLAIN. Pour consulter des exemples de l'impact des expressions, des clauses et des instructions SQL sur la sortie `EXPLAIN`, reportez-vous à Constructions SQL pour les requêtes parallèles dans Aurora MySQL.

L'exemple suivant illustre la différence entre un plan de requête traditionnel et un plan de requête parallèle. Ce plan d'explication provient de la requête 3 du benchmark TPC-H. Un grand nombre d'exemples de requête présentés dans cette section utilise les tables de l'ensemble de données TPC-H. Vous pouvez obtenir les définitions de table, les requêtes et le programme `dbgen` qui génère des exemples de données à partir du site Web TPC-H.

```
EXPLAIN SELECT l_orderkey,
  sum(l_extendedprice * (1 - l_discount)) AS revenue,
  o_orderdate,
  o_shippriority
FROM customer,
  orders,
  lineitem
WHERE c_mktsegment = 'AUTOMOBILE'
AND c_custkey = o_custkey
AND l_orderkey = o_orderkey
AND o_orderdate < date '1995-03-13'
AND l_shipdate > date '1995-03-13'
GROUP BY l_orderkey,
  o_orderdate,
  o_shippriority
ORDER BY revenue DESC,
  o_orderdate LIMIT 10;
```

Par défaut, la requête peut avoir un plan semblable au suivant. Si vous ne voyez pas la jointure par hachage utilisée dans le plan de requête, assurez-vous que l'optimisation est activée en premier.

```
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len |
 ref  | rows     | filtered | Extra                                              |
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
|  1 | SIMPLE      | customer | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL |  1480234 |    10.00 | Using where; Using temporary; Using filesort       |
|  1 | SIMPLE      | orders   | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL | 14875240 |     3.33 | Using where; Using join buffer (Block Nested Loop) |
|  1 | SIMPLE      | lineitem | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL | 59270573 |     3.33 | Using where; Using join buffer (Block Nested Loop) |
+----+-------------+----------+------------+------+---------------+------+---------
+------+----------+----------+----------------------------------------------------+
```

Pour Aurora MySQL version 3, vous activez la jointure par hachage au niveau de la session en exécutant l'instruction suivante.

```
SET optimizer_switch='block_nested_loop=on';
```

Pour Aurora MySQL version 2.09 et versions ultérieures, vous définissez le paramètre de base de données `aurora_disable_hash_join` ou le paramètre de cluster de bases de données sur `0` (off). La désactivation de `aurora_disable_hash_join` définit `optimizer_switch` sur la valeur `hash_join=on`.

Après avoir activé la jointure par hachage, réessayez d'exécuter l'instruction `EXPLAIN`. Pour plus d'informations sur la façon d'utiliser les jointures par hachage efficacement, consultez Optimisation des requêtes de jointure MySQL Aurora volumineuses avec des jointures de hachage.

Lorsque la jointure par hachage est activée et que la fonction de requête parallèle est désactivée, la requête peut avoir un plan d'exécution semblable au suivant, qui utilise la jointure par hachage, mais pas la requête parallèle.

```
+----+-------------+----------+...+-----------
+-------------------------------------------------------------+
| id | select_type | table    |...| rows      | Extra
                          |
```

```
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
|  1 | SIMPLE      | customer |...|   5798330 | Using where; Using index; Using
 temporary; Using filesort       |
|  1 | SIMPLE      | orders   |...| 154545408 | Using where; Using join buffer (Hash
 Join Outer table orders)   |
|  1 | SIMPLE      | lineitem |...| 606119300 | Using where; Using join buffer (Hash
 Join Outer table lineitem) |
+----+-------------+----------+...+-----------
+-------------------------------------------------------------------+
```

Lorsque la fonction de requête parallèle est activée, deux étapes de ce plan de requête peuvent utiliser l'optimisation de requête parallèle, comme l'indique la colonne `Extra` de la sortie `EXPLAIN`. Le traitement des tâches impliquant un usage intensif de l'UC et un grand nombre d'I/O pour ces étapes est délégué à la couche de stockage.

```
+----+...
+-------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                                    |
+----+...
+-------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using index; Using temporary; Using filesort
                                                    |
|  1 |...| Using where; Using join buffer (Hash Join Outer table orders); Using
 parallel query (4 columns, 1 filters, 1 exprs; 0 extra)   |
|  1 |...| Using where; Using join buffer (Hash Join Outer table lineitem); Using
 parallel query (4 columns, 1 filters, 1 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------
+
```

Pour plus d'informations sur l'interprétation de la sortie `EXPLAIN` pour une requête parallèle et sur les parties des instructions SQL auxquelles s'appliquent les requêtes parallèles, consultez Constructions SQL pour les requêtes parallèles dans Aurora MySQL.

L'exemple de sortie suivant présente les résultats de l'exécution de la requête précédente au niveau d'une instance db.r4.2xlarge avec un pool de mémoires tampons à froid. L'exécution de la requête est beaucoup plus rapide avec la fonction de requête parallèle.

> **Note**
>
> Comme la durée dépend de nombreux facteurs environnementaux, vos résultats peuvent différer. Menez toujours vos propres tests de performances pour confirmer ces résultats avec votre propre environnement, votre charge de travail, etc.

```
-- Without parallel query
+------------+-------------+-------------+----------------+
| l_orderkey | revenue     | o_orderdate | o_shippriority |
+------------+-------------+-------------+----------------+
|   92511430 | 514726.4896 | 1995-03-06  |              0 |
.
.
|   28840519 | 454748.2485 | 1995-03-08  |              0 |
+------------+-------------+-------------+----------------+
10 rows in set (24 min 49.99 sec)
```

```
-- With parallel query
+------------+-------------+-------------+----------------+
| l_orderkey | revenue     | o_orderdate | o_shippriority |
+------------+-------------+-------------+----------------+
|   92511430 | 514726.4896 | 1995-03-06  |              0 |
.
.
|   28840519 | 454748.2485 | 1995-03-08  |              0 |
+------------+-------------+-------------+----------------+
10 rows in set (1 min 49.91 sec)
```

Un grand nombre d'exemples de requêtes présentés dans cette section utilise les tables de l'ensemble de données TPC-C, notamment la table PART qui contient 20 millions de lignes et la définition suivante.

```
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| p_partkey     | int(11)       | NO   | PRI | NULL    |       |
| p_name        | varchar(55)   | NO   |     | NULL    |       |
| p_mfgr        | char(25)      | NO   |     | NULL    |       |
| p_brand       | char(10)      | NO   |     | NULL    |       |
| p_type        | varchar(25)   | NO   |     | NULL    |       |
```

```
| p_size        | int(11)       | NO   |     | NULL    |       |
| p_container   | char(10)      | NO   |     | NULL    |       |
| p_retailprice | decimal(15,2) | NO   |     | NULL    |       |
| p_comment     | varchar(23)   | NO   |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

Faites les expérimentations nécessaires avec votre charge de travail afin de déterminer si les instructions SQL individuelles peuvent tirer parti de la fonction de requête parallèle. Utilisez ensuite les techniques de surveillance suivantes pour identifier la fréquence d'utilisation des requêtes parallèles dans les charges de travail réelles au fil du temps. Pour les charges de travail réelles, des facteurs supplémentaires tels que les limites de simultanéité s'appliquent.

Surveillance des requêtes parallèles pour Aurora MySQL

Si votre cluster Aurora MySQL utilise une requête parallèle, vous pouvez voir une augmentation des valeurs d'`VolumeReadIOPS`. Les requêtes parallèles n'utilisent pas le pool de mémoires tampons. Ainsi, bien que les requêtes soient rapides, ce traitement optimisé peut entraîner une augmentation des opérations de lecture et des frais associés.

Outre les métriques Amazon CloudWatch décrites dans Affichage des métriques dans la console Amazon RDS, Aurora fournit d'autres variables d'état globales. Vous pouvez utiliser ces variables d'état global pour surveiller l'exécution des requêtes parallèles. Elles peuvent vous donner des informations sur les raisons pour lesquelles l'optimiseur peut utiliser ou non une requête parallèle dans une situation donnée. Pour accéder à ces variables, vous pouvez utiliser la commande `SHOW GLOBAL STATUS`. Ces variables sont également répertoriées ci-dessous.

Une session de requête parallèle n'est pas nécessairement un mappage un-à-un avec les requêtes effectuées par la base de données. Par exemple, supposons que votre plan de requête contienne deux étapes utilisant la fonction de requête parallèle. Dans ce cas, la requête implique deux sessions parallèles, et les compteurs de tentatives de requêtes et de requêtes réussies sont incrémentés de 2.

Lorsque vous testez la fonction de requête parallèle en émettant des instructions `EXPLAIN`, attendez-vous à constater une augmentation des compteurs désignés comme « non choisis », même si les requêtes ne sont pas en cours d'exécution. Lorsque vous utilisez la fonction de requête parallèle en production, vous pouvez vérifier si le compteur « non choisis » augmentent plus rapidement que prévu. À ce stade, vous pouvez procéder à un ajustement de sorte que la requête parallèle s'exécute pour les requêtes que vous attendez. Pour ce faire, vous pouvez modifier vos paramètres de cluster, la combinaison des requêtes, les instances de base de données où la requête parallèle est activée, etc.

Ces compteurs sont suivis au niveau de l’instance de base de données. Lorsque vous vous connectez à un point de terminaison différent, les métriques peuvent varier, car chaque instance de base de données exécute son propre ensemble de requêtes parallèles. Vous pouvez également voir des métriques différentes lorsque le point de terminaison du lecteur se connecte à une instance de base de données distincte pour chaque session.

Nom	Description
`Aurora_pq_bytes_returned`	Nombre d’octets des structures de données à tuple transmises au nœud principal lors des requêtes parallèles. Divisez cette valeur par 16 384 pour la comparer à `Aurora_pq_pages_pushed_down` .
`Aurora_pq_max_concurrent_requests`	Nombre maximal de sessions de requêtes parallèles pouvant être exécutées simultanément sur cette instance de base de données Aurora. Il s’agit d’un nombre fixe qui dépend de la classe d’instance de base de données AWS.
`Aurora_pq_pages_pushed_down`	Nombre de pages de données (chacune avec une taille fixe de 16 Kio) pour lesquelles une requête parallèle a évité une transmission réseau au nœud principal.
`Aurora_pq_request_attempted`	Nombre de sessions de requêtes parallèles demandées. Cette valeur peut représenter plus d’une session par requête, en fonction des constructions SQL telles que les sous-requêtes et les jointures.
`Aurora_pq_request_executed`	Nombre de sessions de requêtes parallèles ayant réussi.
`Aurora_pq_request_failed`	Nombre de sessions de requêtes parallèles ayant renvoyé une erreur au client. Dans certains cas, les demandes de requêtes parallèles peuvent échouer (en cas de

Nom	Description
	problème au niveau de la couche de stockage, par exemple). Dans ce cas, la partie de la requête ayant échoué fait l'objet d'une autre tentative avec un mécanisme de requête non parallèle. Si cette nouvelle tentative échoue également, une erreur est renvoyée au client, et ce compteur est incrémenté.
`Aurora_pq_request_in_progress`	Nombre de sessions de requêtes parallèles en cours. Ce nombre s'applique à l'instance de base de données Aurora spécifique à laquelle vous êtes connecté, et non à l'ensemble du cluster de bases de données Aurora. Pour déterminer si une instance de base de données se rapproche de sa limite de simultané ité, comparez cette valeur à `Aurora_pq _max_concurrent_requests` .
`Aurora_pq_request_not_chosen`	Nombre de fois qu'une requête parallèle n'a pas été choisie pour accomplir une requête. Cette valeur correspond à la somme de plusieurs autres compteurs plus précis. Une instruction `EXPLAIN` peut incrémenter ce compteur même si la requête n'est pas réellement effectuée.
`Aurora_pq_request_not_chose n_below_min_rows`	Nombre de fois qu'une requête parallèle n'a pas été choisie en raison du nombre de lignes dans la table. Une instruction `EXPLAIN` peut incrémenter ce compteur même si la requête n'est pas réellement effectuée.

Nom	Description
`Aurora_pq_request_not_chosen_column_bit`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle en raison d'un type de données non pris en charge dans la liste des colonnes projetées.
`Aurora_pq_request_not_chosen_column_geometry`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table comporte des colonnes avec le type de données `GEOMETRY`. Pour plus d'informations sur les versions Aurora MySQL qui suppriment cette limitation, consultez Mise à niveau des clusters de requêtes parallèles vers Aurora MySQL version 3.
`Aurora_pq_request_not_chosen_column_lob`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table comporte des colonnes avec un type de données `LOB` ou des colonnes `VARCHAR` stockées en externe en raison de la longueur déclarée. Pour plus d'informations sur les versions Aurora MySQL qui suppriment cette limitation, consultez Mise à niveau des clusters de requêtes parallèles vers Aurora MySQL version 3.
`Aurora_pq_request_not_chosen_column_virtual`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table contient une colonne virtuelle.
`Aurora_pq_request_not_chosen_custom_charset`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table comporte des colonnes avec un jeu de caractères personnalisé.

Nom	Description
`Aurora_pq_request_not_chosen_fast_ddl`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle car la table est actuellement modifiée par une instruction DDL rapide `ALTER`.
`Aurora_pq_request_not_chosen_few_pages_outside_buffer_pool`	Nombre de fois qu'une requête parallèle n'a pas été choisie, même si moins de 95 % des données de la table se trouvait dans le pool de mémoires tampons, car il n'y avait pas suffisamment de données hors mémoire tampon pour que l'application de la fonction de requête parallèle soit justifiée.
`Aurora_pq_request_not_chosen_full_text_index`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table comporte des index de texte intégral.
`Aurora_pq_request_not_chosen_high_buffer_pool_pct`	Nombre de fois qu'une requête parallèle n'a pas été choisie, car un pourcentage élevé de données de la table (pourcentage actuellement supérieur à 95 %) se trouvait déjà dans un pool de mémoires tampons. Dans ce cas, l'optimiseur détermine que la lecture des données à partir du pool de mémoires tampons est plus efficace. Une instruction `EXPLAIN` peut incrémenter ce compteur même si la requête n'est pas réellement effectuée.
`Aurora_pq_request_not_chosen_index_hint`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la requête inclut un indicateur d'index.

Nom	Description
`Aurora_pq_request_not_chosen_innodb_table_format`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table utilise un format de ligne InnoDB non pris en charge. Une requête parallèle Aurora s'applique uniquement aux formats de ligne `COMPACT`, `REDUNDANT` et `DYNAMIC`.
`Aurora_pq_request_not_chosen_long_trx`	Nombre de demandes de requêtes parallèles ayant utilisé le chemin de traitement de requête non parallèle en raison du lancement de la requête dans une transaction de longue durée. Une instruction `EXPLAIN` peut incrémenter ce compteur même si la requête n'est pas réellement effectuée.
`Aurora_pq_request_not_chosen_no_where_clause`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la requête n'inclut aucune clause `WHERE`.
`Aurora_pq_request_not_chosen_range_scan`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la requête utilise une analyse de plage sur un index.
`Aurora_pq_request_not_chosen_row_length_too_long`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la longueur totale combinée de toutes les colonnes est trop élevée.

Nom	Description
`Aurora_pq_request_not_chosen_small_table`	Nombre de fois qu'une requête parallèle n'a pas été choisie en raison de la taille globale de la table, telle que déterminée par le nombre de lignes dans la table et leur longueur moyenne. Une instruction `EXPLAIN` peut incrémenter ce compteur même si la requête n'est pas réellement effectuée.
`Aurora_pq_request_not_chosen_temporary_table`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la requête fait référence à des tables temporaires qui utilisent les types de table `MyISAM` ou `memory` non pris en charge.
`Aurora_pq_request_not_chosen_tx_isolation`	Nombre de demandes de requête parallèle qui utilisent le chemin de traitement de requête non parallèle, car la requête utilise un niveau d'isolement de transaction non pris en charge. Sur les instances de base de données de lecteur, la requête parallèle s'applique uniquement aux niveaux d'isolement `REPEATABLE READ` et `READ COMMITTED` .
`Aurora_pq_request_not_chosen_update_delete_stmts`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la requête fait partie d'une instruction `UPDATE` ou `DELETE`.

Nom	Description
`Aurora_pq_request_not_chosen_unsupported_access`	Nombre de demandes de requêtes parallèles qui utilisent le chemin de traitement de requête non parallèle, car la clause `WHERE` ne remplit pas les critères des requêtes parallèles. Ce résultat peut se produire si la requête ne nécessite aucune analyse à usage intensif de données ou si la requête est une instruction `DELETE` ou `UPDATE`.
`Aurora_pq_request_not_chosen_unsupported_storage_type`	Nombre de demandes de requêtes parallèles qui utilisent le chemin de traitement des requêtes non parallèles parce que le cluster de bases de données Aurora MySQL n'utilise pas de configuration de stockage de cluster Aurora prise en charge. Ce paramètre est disponible dans Aurora MySQL version 3.04 et versions ultérieures. Pour plus d'informations, consultez Limitations.
`Aurora_pq_request_throttled`	Nombre de fois qu'une requête parallèle n'a pas été choisie en raison du nombre maximal de requêtes parallèles simultanées déjà exécutées sur une instance de base de données Aurora spécifique.

Constructions SQL pour les requêtes parallèles dans Aurora MySQL

Dans la section suivante, vous trouverez plus de détails sur les raisons pour lesquelles certaines instructions SQL utilisent ou n'utilisent pas la requête parallèle. Cette section détaille également la façon dont les fonctions Aurora MySQL interagissent avec la requête parallèle. Ces informations peuvent vous aider à diagnostiquer les problèmes de performances d'un cluster qui utilise les requêtes parallèles, ou à comprendre comment cette fonction s'applique pour une charge de travail spécifique.

La décision d'utiliser la fonction de requête parallèle dépend d'un grand nombre de facteurs qui interviennent au moment de l'exécution de l'instruction. Dès lors, les requêtes parallèles peuvent être déclenchées pour certaines requêtes à chaque fois, jamais ou uniquement dans certaines conditions.

> **Tip**
>
> Lorsque vous affichez ces exemples au format HTML, vous pouvez utiliser le widget Copy (Copier) dans le coin supérieur droit de chaque liste de codes pour copier le code SQL de manière à l'essayer par vous-même. L'utilisation du widget Copy (Copier) permet d'éviter de copier les caractères supplémentaires autour de l'invite `mysql>` et des lignes `->` suivantes.

Rubriques

- Tables temporaires MyISAM

L'instruction EXPLAIN

Comme illustré dans divers exemples de cette section, l'instruction `EXPLAIN` indique si chaque stade d'une requête est éligible à la fonction de requête parallèle. Elle indique également quels aspects d'une requête peuvent être délégués à la couche de stockage. Voici les éléments les plus importants du plan de requête :

- Une valeur autre que `NULL` pour la colonne `key` suggère que la requête peut être effectuée efficacement via les recherches d'index et qu'une requête parallèle est peu probable.
- Une faible valeur pour la colonne `rows` (valeur inférieure à 1 million) suggère que la requête n'accède pas à suffisamment de données pour que la fonction de requête parallèle soit justifiée. Cela signifie que la requête parallèle est peu probable.
- La colonne `Extra` vous indique si l'utilisation de la fonction de requête parallèle est prévue. La sortie ressemble à l'exemple suivant.

  ```
  Using parallel query (A columns, B filters, C exprs; D extra)
  ```

  Le nombre `columns` représente le nombre de colonnes auquel le bloc de requêtes fait référence.

  Le nombre `filters` indique de nombre de prédicats `WHERE` représentant une simple comparaison d'une valeur de colonne par rapport à une constante. La comparaison peut rechercher une égalité, une inégalité ou une plage. Aurora permet de mettre en parallèle ces types de prédicats plus efficacement.

  Le nombre `exprs` représente le nombre d'expressions, comme les appels de fonction, les opérateurs ou autres, qui peuvent également être mis en parallèle, bien que cela soit moins efficace qu'avec une condition de filtrage.

  Le nombre `extra` représente le nombre d'expressions qui ne peuvent pas être déléguées et qui sont effectuées par le nœud principal.

Par exemple, considérons la sortie `EXPLAIN` suivante.

```
mysql> explain select p_name, p_mfgr from part
    -> where p_brand is not null
    -> and upper(p_type) is not null
```

```
    -> and round(p_retailprice) is not null;
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type | table |...| rows     | Extra
                                |
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
|  1 | SIMPLE      | part  |...| 20427936 | Using where; Using parallel query (5
 columns, 1 filters, 2 exprs; 0 extra) |
+----+-------------+-------+...+----------
+-------------------------------------------------------------------------------+
```

Les informations de la colonne `Extra` montre que cinq colonnes sont extraites de chaque ligne afin d'évaluer les conditions de la requête et de construire le jeu de résultats. Un prédicat `WHERE` implique un filtre, à savoir une colonne directement testée dans la clause `WHERE`. Deux clauses `WHERE` nécessitent l'évaluation d'expressions plus complexes et qui impliquent ici des appels de fonction. Le champ `0 extra` confirme que toutes les opérations de la clause `WHERE` sont déléguées à la couche de stockage dans le cadre du traitement de requête parallèle.

Dans les cas où une requête parallèle n'est pas choisie, vous pouvez généralement déduire la raison à partir des autres colonnes de la sortie `EXPLAIN`. Par exemple, la valeur `rows` peut être trop faible, ou la colonne `possible_keys` peut indiquer que la requête est en mesure d'utiliser une recherche d'index au lieu d'une analyse à usage intensif de données. L'exemple suivant montre une requête dans laquelle l'optimiseur peut estimer que la requête n'analysera qu'un petit nombre de lignes. Cette estimation est basée sur les caractéristiques de la clé principale. Dans ce cas, aucune requête parallèle n'est requise.

```
mysql> explain select count(*) from part where p_partkey between 1 and 100;
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
| id | select_type | table | type  | possible_keys | key     | key_len | ref  | rows |
 Extra                    |
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
|  1 | SIMPLE      | part  | range | PRIMARY       | PRIMARY | 4       | NULL |   99 |
 Using where; Using index |
+----+-------------+-------+-------+---------------+---------+---------+------+------
+--------------------------+
```

La sortie indiquant si une requête parallèle sera utilisée tient compte de tous les facteurs qui entrent en jeu au moment de l'exécution de l'instruction `EXPLAIN`. L'optimiseur peut choisir une autre option

lorsque la requête est exécutée, si la situation a changé entre-temps. Par exemple, `EXPLAIN` peut indiquer qu'une instruction utilisera la fonction de requête parallèle. Toutefois, au moment d'exécuter la requête, il peut choisir de ne pas utiliser cette fonction selon les conditions qui s'appliquent à ce moment-là. Ces conditions peuvent inclure plusieurs autres requêtes parallèles s'exécutant simultanément. Elles peuvent également inclure des lignes supprimées de la table, un nouvel index en cours de création, trop de temps passé dans une transaction ouverte, etc.

Clause WHERE

Pour qu'une requête puisse tirer parti de l'optimisation via les requêtes parallèles, elle doit inclure une clause `WHERE`.

L'optimisation via les requêtes parallèles accélère de nombreux types d'expressions utilisés dans la clause `WHERE` :

- Simples comparaisons d'une valeur de colonne par rapport à une constante, aussi connues en tant que filtres. Ces comparaisons sont optimales lorsqu'elles sont déléguées à la couche de stockage. Le nombre d'expressions de filtrage d'une requête est indiqué dans la sortie `EXPLAIN`.
- Les autres types d'expressions de la clause `WHERE` sont également déléguées à la couche de stockage, le cas échéant. Le nombre d'expressions de ce type dans une requête est indiqué dans la sortie `EXPLAIN`. Il peut s'agir d'appels de fonction, d'opérateurs `LIKE`, d'expressions `CASE`, etc.
- Certaines fonctions ne sont pas déléguées par les requêtes parallèles. Le nombre d'expressions de ce type dans une requête est indiqué sous la forme du compteur `extra` dans la sortie `EXPLAIN`. Le reste de la requête peut utiliser la fonction de requête parallèle.
- Bien que les expressions de la liste de sélection ne soient pas déléguées, les requêtes contenant ces fonctions peuvent bénéficier d'une réduction du trafic réseau pour les résultats intermédiaires des requêtes parallèles. Par exemple, les requêtes qui appellent des fonctions d'agrégation dans la liste de sélection peuvent bénéficier des requêtes parallèles même si les fonctions d'agrégation ne sont pas déléguées.

Par exemple, la requête suivante effectue une analyse de la table complète et traite toutes les valeurs pour la colonne `P_BRAND`. Toutefois, elle n'a pas recours à une requête, car elle n'inclut pas de clause `WHERE`.

```
mysql> explain select count(*), p_brand from part group by p_brand;
+----+-------------+-------+------+--------------+------+---------+------+----------
+---------------------------------+
```

```
| id | select_type | table | type | possible_keys | key  | key_len | ref  | rows     |
 Extra                          |
+----+-------------+-------+------+---------------+------+---------+------+----------
+--------------------------------+
|  1 | SIMPLE      | part  | ALL  | NULL          | NULL | NULL    | NULL | 20427936 |
 Using temporary; Using filesort |
+----+-------------+-------+------+---------------+------+---------+------+----------
+--------------------------------+
```

En revanche, la requête suivante comprend les prédicats `WHERE` qui filtrent les résultats, de sorte qu'une requête parallèle peut être appliquée :

```
mysql> explain select count(*), p_brand from part where p_name is not null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
    -> group by p_brand;
+----+...+----------
+-----------------------------------------------------------------------------------------------
+
| id |...| rows     | Extra
                                          |
+----+...+----------
+-----------------------------------------------------------------------------------------------
+
|  1 |...| 20427936 | Using where; Using temporary; Using filesort; Using parallel
 query (5 columns, 1 filters, 2 exprs; 0 extra) |
+----+...+----------
+-----------------------------------------------------------------------------------------------
+
```

Si l'optimiseur estime que le nombre de lignes renvoyées pour un bloc de requêtes est faible, la fonction de requête parallèle n'est pas utilisée pour ce bloc. L'exemple suivant présente un scénario où un opérateur « supérieur à » dans la colonne de clé primaire s'applique à des millions de lignes, ce qui entraîne l'utilisation d'une requête parallèle. Il est estimé que l'opérateur contraire « inférieur à » s'applique uniquement à quelques lignes et qu'il n'utilise donc pas la fonction de requête parallèle.

```
mysql> explain select count(*) from part where p_partkey > 10;
+----+...+----------
+-------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
```

```
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------------+

mysql> explain select count(*) from part where p_partkey < 10;
+----+...+------+--------------------------+
| id |...| rows | Extra                    |
+----+...+------+--------------------------+
|  1 |...|    9 | Using where; Using index |
+----+...+------+--------------------------+
```

Langage de définition de données (DDL)

Dans Aurora MySQL version 2, les requêtes parallèles sont disponibles uniquement pour les tables pour lesquelles aucune opération rapide en langage de définition de données (DDL) n'est en attente. Dans Aurora MySQL version 3, vous pouvez utiliser une requête parallèle sur une table en même temps qu'une opération Instant DDL.

Le langage DDL instantané dans Aurora MySQL version 3 remplace la fonctionnalité de langage DDL rapide d'Aurora MySQL version 2. Pour plus d'informations sur Instant DDL, consultez Instant DDL (Aurora MySQL version 3).

Types de données de colonne

Dans Aurora MySQL version 3, les requêtes parallèles sont compatibles avec des tables contenant des colonnes avec des types de données `TEXT`, `BLOB`, `JSON` et `GEOMETRY`. Elles peuvent également être utilisées avec les colonnes `VARCHAR` et `CHAR` dont la longueur maximale déclarée est supérieure à 768 octets. Si votre requête fait référence à des colonnes contenant de tels types d'objets volumineux, le travail supplémentaire de récupération ajoute une surcharge au traitement des requêtes. Le cas échéant, vérifiez si la requête peut omettre les références à ces colonnes. Dans le cas contraire, exécutez des points de référence pour vérifier si ces requêtes sont plus rapides lorsque la requête parallèle est activée ou désactivée.

Dans Aurora MySQL version 2, les requêtes parallèles présentent les limites suivantes pour les types d'objets volumineux :

- Les types de données `TEXT`, `BLOB`, `JSON` et `GEOMETRY` ne sont pas pris en charge avec les requêtes parallèles. Les requêtes faisant référence à des colonnes de ce type ne peuvent pas utiliser les requêtes parallèles.
- Les colonnes de longueur variable (types de données `VARCHAR` et `CHAR`) sont compatibles avec les requêtes parallèles jusqu'à une longueur déclarée maximale de 768 octets. Les requêtes faisant référence à des colonnes de ce type ayant une longueur maximale supérieure ne peuvent pas utiliser les requêtes parallèles. Pour les colonnes utilisant des jeux de caractères multi-octets, la limite du nombre d'octets tient compte du nombre maximal d'octets de ces jeux de caractères. Par exemple, pour le jeu de caractères `utf8mb4` (dont la longueur de caractères maximale est de 4 octets), une colonne `VARCHAR(192)` est compatible avec les requêtes parallèles, mais pas une colonne `VARCHAR(193)`.

Tables partitionnées

Vous pouvez utiliser des tables partitionnées avec des requêtes parallèles dans Aurora MySQL version 3. Les tables partitionnées étant représentées en interne sous la forme de plusieurs tables plus petites, une requête qui utilise une requête parallèle sur une table non partitionnée peut ne pas utiliser de requête parallèle sur une table partitionnée identique. Aurora MySQL examine si chaque partition est suffisamment volumineuse pour être admissible à l'optimisation des requêtes parallèles, au lieu d'évaluer la taille de la table entière. Vérifiez si la variable d'état `Aurora_pq_request_not_chosen_small_table` est incrémentée si une requête sur une table partitionnée n'utilise pas de requête parallèle lorsque vous l'attendez.

Par exemple, considérez une table partitionnée avec `PARTITION BY HASH (`*`column`*`) PARTITIONS 2` et une autre table partitionnée avec `PARTITION BY HASH (`*`column`*`) PARTITIONS 10`. Dans le tableau comportant deux partitions, les partitions sont cinq fois plus volumineuses que la table avec dix partitions. Par conséquent, la requête parallèle est plus susceptible d'être utilisée pour les requêtes sur la table comportant moins de partitions. Dans l'exemple suivant, la table `PART_BIG_PARTITIONS` compte deux partitions et `PART_SMALL_PARTITIONS` possède dix partitions. Avec des données identiques, la requête parallèle est plus susceptible d'être utilisée pour la table comportant moins de partitions volumineuses.

```
mysql> explain select count(*), p_brand from part_big_partitions where p_name is not
 null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
 group by p_brand;
```

```
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------------
+
| id | select_type | table               | partitions | Extra
                                                                                          |
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------------
+
|  1 | SIMPLE      | part_big_partitions | p0,p1      | Using where; Using temporary;
 Using parallel query (4 columns, 1 filters, 1 exprs; 0 extra; 1 group-bys, 1 aggrs) |
+----+-------------+---------------------+------------
+-------------------------------------------------------------------------------------------
+

mysql> explain select count(*), p_brand from part_small_partitions where p_name is not
 null
    ->   and p_mfgr in ('Manufacturer#1', 'Manufacturer#3') and p_retailprice > 1000
 group by p_brand;
+----+-------------+----------------------+------------------------------
+------------------------------+
| id | select_type | table                | partitions                   | Extra
                  |
+----+-------------+----------------------+------------------------------
+------------------------------+
|  1 | SIMPLE      | part_small_partitions | p0,p1,p2,p3,p4,p5,p6,p7,p8,p9 | Using
 where; Using temporary |
+----+-------------+----------------------+------------------------------
+------------------------------+
```

Fonctions d’agrégation et clauses GROUP BY et HAVING

Les requêtes impliquant des fonctions d’agrégation sont souvent adaptées aux requêtes parallèles, car elles impliquent l’analyse de grand nombres de lignes dans des tables de grande taille.

Dans Aurora MySQL 3, une requête parallèle permet d’optimiser les appels de fonction agrégés dans la liste de sélection et la clause `HAVING`.

Avant Aurora MySQL 3, les appels de fonction d’agrégation qui se trouvent dans la liste de sélection ou la clause `HAVING` ne sont pas délégués à la couche de stockage. Cependant, la fonction de requête parallèle peut tout de même améliorer les performances de ces requêtes avec les fonctions d’agrégation. Pour ce faire, elle commence par extraire en parallèle les valeurs de colonne à partir des pages de données brutes au niveau de la couche de stockage. Puis, elle retransmet ces valeurs

au nœud principal sous forme de tuple compact au lieu de pages de données complètes. Comme toujours, la requête nécessite au moins un prédicat `WHERE` pour que la fonction de requête parallèle soit activée.

Les exemples simples suivants illustrent les types de requêtes agrégées pouvant bénéficier d'une requête parallèle. Les résultats intermédiaires sont renvoyés sous forme compacte au nœud principal et/ou les lignes qui ne correspondent pas sont filtrées des résultats intermédiaires.

```
mysql> explain select sql_no_cache count(distinct p_brand) from part where p_mfgr =
 'Manufacturer#5';
+----+...+-------------------------------------------------------------------------+
| id |...| Extra                                                                   |
+----+...+-------------------------------------------------------------------------+
|  1 |...| Using where; Using parallel query (2 columns, 1 filters, 0 exprs; 0 extra) |
+----+...+-------------------------------------------------------------------------+

mysql> explain select sql_no_cache p_mfgr from part where p_retailprice > 1000 group by
 p_mfgr having count(*) > 100;
+----+...
+-------------------------------------------------------------------------------------------------
+
| id |...| Extra
                                  |
+----+...
+-------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using filesort; Using parallel query (3
 columns, 0 filters, 1 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------
+
```

Appels de fonction dans une clause WHERE

Aurora permet d'appliquer l'optimisation des appels via les requêtes parallèles dans la plupart des fonctions intégrées de la clause `WHERE`. La mise en parallèle de ces appels de fonction permet de réduire la sollicitation de l'UC depuis le nœud principal. L'évaluation des fonctions de prédicat en parallèle lors des premiers stades d'une requête permet à Aurora de minimiser la quantité de données transmises et traitées lors des stades ultérieurs.

À ce stade, la parallélisation ne s'applique pas aux appels de fonction qui se trouvent dans la liste de sélection. Ces fonctions sont évaluées par le nœud principal même si des appels de

fonction identiques apparaissent dans la clause `WHERE`. Les valeurs d'origine issues des colonnes appropriées sont incluses dans les tuples retransmis depuis les nœuds de stockage vers le nœud principal. Le nœud principal effectue toutes les transformations telles que `UPPER`, `CONCATENATE`, etc., afin de générer les valeurs finales du jeu de résultats.

Dans l'exemple suivant, la requête parallèle met en parallèle l'appel à `LOWER`, car il apparaît dans la clause `WHERE`. La fonction de requête parallèle ne s'applique pas aux appels à `SUBSTR` et `UPPER`, car ils se trouvent dans la liste de sélection.

```
mysql> explain select sql_no_cache distinct substr(upper(p_name),1,5) from part
    -> where lower(p_name) like '%cornflower%' or lower(p_name) like '%goldenrod%';
+----+...
+-------------------------------------------------------------------------------------------
+
| id |...| Extra
                |
+----+...
+-------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using parallel query (2 columns, 0 filters, 1
 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------
+
```

Les mêmes considérations sont valables pour les autres expressions, telles que les expressions `CASE` ou les opérateurs `LIKE`. L'exemple suivant montre une requête parallèle évaluant l'expression `CASE` et les opérateurs `LIKE` dans la clause `WHERE`.

```
mysql> explain select p_mfgr, p_retailprice from part
    -> where p_retailprice > case p_mfgr
    ->   when 'Manufacturer#1' then 1000
    ->   when 'Manufacturer#2' then 1200
    ->   else 950
    -> end
    -> and p_name like '%vanilla%'
    -> group by p_retailprice;
+----+...
+-------------------------------------------------------------------------------------------
+
| id |...| Extra
                                |
```

```
+----+...
+-------------------------------------------------------------------------------------------------
+
|  1 |...| Using where; Using temporary; Using filesort; Using parallel query (4
 columns, 0 filters, 2 exprs; 0 extra) |
+----+...
+-------------------------------------------------------------------------------------------------
+
```

Clause LIMIT

À ce stade, les requêtes parallèles ne sont pas utilisées pour les blocs de requêtes incluant une clause `LIMIT`. Les requêtes parallèles peuvent être utilisées pour les phases de requêtes précédentes avec les clauses `GROUP BY`, `ORDER BY` ou `JOIN`.

Opérateurs de comparaison

L'optimiseur estime le nombre de lignes à analyser pour évaluer les opérateurs de comparaison et se base sur cette estimation pour déterminer si une requête parallèle est justifiée ou non.

Le premier exemple ci-dessous montre qu'une comparaison d'égalité par rapport à la colonne de clé primaire peut être effectuée efficacement sans requête parallèle. Le deuxième exemple ci-dessous montre qu'une comparaison similaire par rapport à une colonne non indexée nécessite l'analyse de millions de lignes et peut donc tirer parti de la fonction de requête parallèle.

```
mysql> explain select * from part where p_partkey = 10;
+----+...+------+-------+
| id |...| rows | Extra |
+----+...+------+-------+
|  1 |...|    1 | NULL  |
+----+...+------+-------+

mysql> explain select * from part where p_type = 'LARGE BRUSHED BRASS';
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (9 columns, 1 filters, 0 exprs;
 0 extra) |
```

```
+----+...+----------
+-------------------------------------------------------------------------+
```

Les mêmes considérations s'appliquent pour les comparaisons d'inégalité ou de plages telles que celles réalisées avec les opérateurs « inférieur à », « supérieur à » ou « égal à », ou encore `BETWEEN`. L'optimiseur estime le nombre de lignes à analyser et se base sur le volume total d'I/O pour déterminer si une requête parallèle est justifiée.

Jointures

Les requêtes de jointure comprenant des tables de grande taille impliquent généralement des opérations à usage intensif de données qui tirent parti de l'optimisation via les requêtes parallèles. À ce stade, les comparaisons de valeurs de colonne entre plusieurs tables (autrement dit, les prédicats de jointure eux-mêmes) ne sont pas mises en parallèle. Cependant, la fonction de requête parallèle peut déléguer une partie du traitement interne pour d'autres phases de jointure, telles que la construction du filtre Bloom lors d'une jointure de hachage. La fonction de requête parallèle peut d'appliquer aux requêtes de jointure même sans clause `WHERE`. Par conséquent, les requêtes de jointure sont l'exception à la règle selon laquelle une clause `WHERE` est requise pour pouvoir utiliser une requête parallèle.

Chaque phase de traitement d'une jointure est évaluée afin de déterminer si elle est éligible à la fonction de requête parallèle. Si plusieurs phases sont éligibles, elles sont effectuées l'une après l'autre. De cette manière, chaque requête de jointure est comptabilisée comme une seule session de requête parallèle en termes de limites de simultanéité.

Par exemple, lorsqu'une requête de jointure inclut des prédicats `WHERE` pour filtrer les lignes de l'une des tables jointes, cette option de filtrage peut utiliser la fonction de requête parallèle. Un autre exemple est celui d'une requête de jointure qui utilise le mécanisme de jointure de hachage pour joindre une table de grande taille à une petite table. Dans ce cas, la fonction de requête parallèle peut s'appliquer à l'analyse des tables permettant de générer la structure de données du filtre Bloom.

> **Note**
>
> Les requêtes parallèles sont généralement utilisées pour les types de requêtes gourmandes en ressources qui tirent parti de l'optimisation de la jointure par hachage. La méthode d'activation de l'optimisation de la jointure par hachage dépend de la version d'Aurora MySQL. Pour plus d'informations sur chaque version, consultez Activation de la jointure par hachage pour les clusters de requête parallèle. Pour plus d'informations sur la façon d'utiliser

les jointures par hachage efficacement, consultez Optimisation des requêtes de jointure MySQL Aurora volumineuses avec des jointures de hachage.

```
mysql> explain select count(*) from orders join customer where o_custkey = c_custkey;
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
| id |...| table    | type  | possible_keys | key         |...| rows      | Extra

              |
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
|  1 |...| customer | index | PRIMARY       | c_nationkey |...|  15051972 | Using index

              |
|  1 |...| orders   | ALL   | o_custkey     | NULL        |...| 154545408 | Using join
 buffer (Hash Join Outer table orders); Using parallel query (1 columns, 0 filters, 1
 exprs; 0 extra) |
+----+...+----------+-------+--------------+-------------+...+-----------
+-------------------------------------------------------------------------------------------
+
```

Pour une requête de jointure qui utilise le mécanisme de boucle imbriquée, le bloc de boucles imbriquées qui se trouve le plus à l'extérieur peut utiliser la fonction de requête parallèle. L'utilisation des requêtes parallèles dépend des facteurs habituels, tels que la présence d'autres conditions de filtrage dans la clause `WHERE`.

```
mysql> -- Nested loop join with extra filter conditions can use parallel query.
mysql> explain select count(*) from part, partsupp where p_partkey != ps_partkey and
 p_name is not null and ps_availqty > 0;
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------+
| id | select_type | table    |...| rows     | Extra
                                  |
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------+
|  1 | SIMPLE      | part     |...| 20427936 | Using where; Using parallel query (2
 columns, 1 filters, 0 exprs; 0 extra) |
|  1 | SIMPLE      | partsupp |...| 78164450 | Using where; Using join buffer (Block
 Nested Loop)                       |
```

```
+----+-------------+----------+...+----------
+-------------------------------------------------------------------------------+
```

Sous-requêtes

Le bloc de requête externe et le bloc de sous-requête interne peuvent chacun utiliser ou non une requête parallèle. Cela dépend des caractéristiques habituelles de la table, de la clause `WHERE`, et ainsi de suite, pour chaque bloc. Par exemple, la requête suivante utilise la fonction de requête parallèle pour le bloc de sous-requêtes, mais pas pour le bloc externe.

```
mysql> explain select count(*) from part where
   --> p_partkey < (select max(p_partkey) from part where p_name like '%vanilla%');
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type |...| rows     | Extra
                       |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
|  1 | PRIMARY     |...|     NULL | Impossible WHERE noticed after reading const tables
                       |
|  2 | SUBQUERY    |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
+----+-------------+...+----------
+-------------------------------------------------------------------------------+
```

Actuellement, les sous-requêtes corrélées ne peuvent pas utiliser la fonction d'optimisation via les requêtes parallèles.

UNION

Chaque bloc de requêtes d'une instruction `UNION` peut utiliser la fonction de requête parallèle ou pas selon les caractéristiques habituelles de la table, de la clause `WHERE`, etc. pour chaque partie de l'instruction `UNION`.

```
mysql> explain select p_partkey from part where p_name like '%choco_ate%'
    -> union select p_partkey from part where p_name like '%vanil_a%';
+----+---------------+...+----------
+-------------------------------------------------------------------------------+
| id | select_type    |...| rows     | Extra
                          |
+----+---------------+...+----------
+-------------------------------------------------------------------------------+
```

```
|  1 | PRIMARY        |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
|  2 | UNION          |...| 20427936 | Using where; Using parallel query (2 columns, 0
 filters, 1 exprs; 0 extra) |
| NULL | UNION RESULT | <union1,2> |...|     NULL | Using temporary
                       |
+----+--------------+...+----------
+-------------------------------------------------------------------------------+
```

> **Note**
>
> Chaque clause `UNION` de la requête est exécutée dans l'ordre. Même si la requête inclut plusieurs stades qui utilisent tous la fonction de requête parallèle, elle exécute une seule et même requête parallèle. Par conséquent, même une requête complexe à plusieurs phases est comptabilisée comme une seule et même requête dans la limite de requêtes parallèles simultanées.

Vues

L'optimiseur réécrit les requêtes avec une vue comme requête de longue taille utilisant les tables sous-jacentes. Dès lors, la requête parallèle fonctionne de la même manière, et ce que les références de tables soient des vues ou des tables réelles. Toutes les considérations concernant l'utilisation de la fonction de requête parallèle pour une requête, ainsi que les parties qui sont déléguées, s'appliquent à la requête réécrite finale.

Par exemple, le plan de requête suivant présente une définition de vue qui n'utilise généralement pas les requêtes parallèles. Lorsque cette vue est interrogée avec d'autres clauses `WHERE`, Aurora MySQL fait appel à une requête parallèle.

```
mysql> create view part_view as select * from part;
mysql> explain select count(*) from part_view where p_partkey is not null;
+----+...+----------
+-------------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (1 columns, 0 filters, 0 exprs;
 1 extra) |
```

```
+----+...+----------
+-------------------------------------------------------------------------+
```

Instructions en langage de manipulation de données (DML)

L'instruction `INSERT` peut utiliser la fonction de requête parallèle pour la phase de traitement `SELECT` si la partie `SELECT` remplit les autres conditions relatives à cette fonction.

```
mysql> create table part_subset like part;
mysql> explain insert into part_subset select * from part where p_mfgr =
 'Manufacturer#1';
+----+...+----------
+-------------------------------------------------------------------------+
| id |...| rows     | Extra
          |
+----+...+----------
+-------------------------------------------------------------------------+
|  1 |...| 20427936 | Using where; Using parallel query (9 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+----------
+-------------------------------------------------------------------------+
```

> **Note**
>
> Habituellement, après une instruction `INSERT`, les données des lignes qui viennent d'être insérées se trouvent dans le pool de mémoires tampons. Dès lors, une table n'est pas toujours éligible à la fonction de requête parallèle juste après l'insertion d'un grand nombre de lignes. Ultérieurement, une fois que les données seront éliminées du pool de mémoires tampons pendant le fonctionnement normal, les requêtes liées à la table pourront commencer à réutiliser la fonction de requête parallèle.

L'instruction `CREATE TABLE AS SELECT` n'utilise pas la fonction de requête parallèle même si la portion `SELECT` de l'instruction y est éligible. L'aspect DDL de cette instruction la rend incompatible avec le traitement de requête parallèle. En revanche, dans l'instruction `INSERT ... SELECT`, la portion `SELECT` peut utiliser la fonction de requête parallèle.

Les requêtes parallèles ne sont jamais utilisées pour les instructions `DELETE` ou `UPDATE`, indépendamment de la taille de la table et des prédicats de la clause `WHERE`.

```
mysql> explain delete from part where p_name is not null;
+----+-------------+...+----------+-------------+
| id | select_type |...| rows     | Extra       |
+----+-------------+...+----------+-------------+
|  1 | SIMPLE      |...| 20427936 | Using where |
+----+-------------+...+----------+-------------+
```

Transactions et verrouillage

Vous pouvez utiliser tous les niveaux d'isolement de l'instance principale Aurora.

Sur les instances de base de données de lecteur Aurora, la requête parallèle s'applique aux instructions exécutées sous le niveau d'isolement `REPEATABLE READ`. Aurora MySQL version 2.09 ou ultérieure peut également utiliser le niveau d'isolement `READ COMMITTED` sur des instances de base de données de lecteur. `REPEATABLE READ` est le niveau d'isolement par défaut pour les instances de base de données de lecteur Aurora. Pour utiliser le niveau d'isolement `READ COMMITTED` sur les instances de base de données de lecteur, l'option de configuration `aurora_read_replica_read_committed` doit être définie au niveau de la session. Le niveau d'isolement `READ COMMITTED` pour les instances de lecteur est conforme au comportement SQL standard. Toutefois, l'isolation est moins stricte sur les instances de lecteur que lorsque les requêtes utilisent le niveau d'isolement `READ COMMITTED` sur l'instance d'enregistreur.

Pour plus d'informations sur les niveaux d'isolement d'Aurora, en particulier sur les différences dans `READ COMMITTED` entre les instances d'enregistreur et de lecteur, consultez Niveaux d'isolement Aurora MySQL.

Une fois qu'une transaction importante est terminée, les statistiques de la table peuvent être obsolètes. Ces statistiques obsolètes peuvent nécessiter une instruction `ANALYZE TABLE` avant qu'Aurora puisse estimer précisément le nombre de lignes. Une instruction DML à grand échelle peut également entraîner le stockage d'une portion significative des données de la table dans le pool de mémoires tampons. Le stockage de ces données dans le pool de mémoires tampons peut entraîner une utilisation moins fréquente de la fonction de requête parallèle pour cette table tant que les données ne seront pas éliminées du pool.

Lorsque votre session fait partie d'une transaction de longue durée (par défaut, 10 minutes), les autres requêtes de cette session n'utilisent pas la fonction de requête parallèle. L'expiration du délai d'attente peut également avoir lieu lors d'une requête unique de longue durée. Cela peut se produit si la requête dure plus longtemps que l'intervalle maximal autorisé (actuellement fixé à 10 minutes) avant le début du traitement de requête parallèle

Pour limiter le risque de lancement accidentel de transactions de longue durée, définissez `autocommit=1` dans les sessions `mysql` dans lesquelles vous effectuez des requêtes ad hoc (exceptionnelles). Même une instruction `SELECT` par rapport à une table commence une transaction en créant une vue de lecture. Une vue de lecture est un ensemble de données constant pour les requêtes ultérieures. Elle est conservée jusqu'à ce que la transaction soit validée. Gardez cette restriction à l'esprit, notamment lorsque vous utilisez des applications JDBC ou ODBC avec Aurora, car celles-ci peuvent être exécutées sans que le paramètre `autocommit` soit activé.

L'exemple suivant montre comment l'exécution d'une requête liée à une table crée une vue de lecture lançant implicitement une transaction lorsque le paramètre `autocommit` est désactivé. Les requêtes exécutées peu de temps après peuvent utiliser la fonction de requête parallèle. Toutefois, après une pause de plusieurs minutes, ces requêtes ne sont plus éligibles à la fonction de requête parallèle. Mettre fin à la transaction avec `COMMIT` ou `ROLLBACK` restaure l'éligibilité à cette fonction.

```
mysql> set autocommit=0;

mysql> explain select sql_no_cache count(*) from part where p_retailprice > 10.0;
+----+...+---------
+-------------------------------------------------------------------------------+
| id |...| rows    | Extra
         |
+----+...+---------
+-------------------------------------------------------------------------------+
|  1 |...| 2976129 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+---------
+-------------------------------------------------------------------------------+

mysql> select sleep(720); explain select sql_no_cache count(*) from part where
 p_retailprice > 10.0;
+------------+
| sleep(720) |
+------------+
|          0 |
+------------+
1 row in set (12 min 0.00 sec)

+----+...+---------+-------------+
| id |...| rows    | Extra       |
+----+...+---------+-------------+
|  1 |...| 2976129 | Using where |
+----+...+---------+-------------+
```

```
mysql> commit;

mysql> explain select sql_no_cache count(*) from part where p_retailprice > 10.0;
+----+...+---------
+-------------------------------------------------------------------------------+
| id |...| rows    | Extra
         |
+----+...+---------
+-------------------------------------------------------------------------------+
|  1 |...| 2976129 | Using where; Using parallel query (1 columns, 1 filters, 0 exprs;
 0 extra) |
+----+...+---------
+-------------------------------------------------------------------------------+
```

Pour déterminer le nombre de fois que des requêtes n'ont pas été éligibles à une requête parallèle parce qu'elles faisaient partie de transactions de longue durée, vérifiez la variable de statut `Aurora_pq_request_not_chosen_long_trx`.

```
mysql> show global status like '%pq%trx%';
+--------------------------------------+-------+
| Variable_name                        | Value |
+--------------------------------------+-------+
| Aurora_pq_request_not_chosen_long_trx | 4     |
+---------------------------------+-------+
```

Toute instruction `SELECT` qui acquiert des verrous, telle la syntaxe `SELECT FOR UPDATE` ou `SELECT LOCK IN SHARE MODE`, ne peut pas utiliser la fonction de requête parallèle.

Les requêtes parallèles peuvent s'appliquer aux tables verrouillées par une instruction `LOCK TABLES`.

```
mysql> explain select o_orderpriority, o_shippriority from orders where o_clerk =
 'Clerk#000095055';
+----+...+-----------
+-------------------------------------------------------------------------------+
| id |...| rows      | Extra
           |
+----+...+-----------
+-------------------------------------------------------------------------------+
|  1 |...| 154545408 | Using where; Using parallel query (3 columns, 1 filters, 0
 exprs; 0 extra) |
```

```
+----+...+-----------
+-------------------------------------------------------------------------------+

mysql> explain select o_orderpriority, o_shippriority from orders where o_clerk =
 'Clerk#000095055' for update;
+----+...+-----------+-------------+
| id |...| rows      | Extra       |
+----+...+-----------+-------------+
|  1 |...| 154545408 | Using where |
+----+...+-----------+-------------+
```

Index B-Tree

Les statistiques recueillies par l'instruction `ANALYZE TABLE` permettent à l'optimiseur de déterminer quand utiliser la fonction de requête parallèle ou les recherches d'index en fonction des caractéristiques des données pour chaque colonne. Pour que les statistiques restent à jour, exécutez `ANALYZE TABLE` après toute opération DML apportant des modifications significatives aux données d'une table.

Si les recherches d'index peuvent effectuer une requête efficacement sans analyse à usage intensif de données, Aurora peut privilégier cette option. Cette approche permet d'éviter les frais supplémentaires liés au traitement de requête parallèle. Des limites de simultanéité s'appliquent également au nombre de requêtes parallèle qui peuvent être exécutées simultanément dans un cluster de bases de données Aurora. Veillez à respecter les bonnes pratiques pour l'indexation des tables, de sorte que les requêtes les plus fréquentes et utilisant le plus la simultanéité aient recours aux recherches d'index.

Index de recherche en texte intégral

Actuellement, les requêtes parallèles ne sont pas utilisées pour les tables qui contiennent un index de recherche en texte intégral, peu importe que la requête fasse référence à ces colonnes indexées ou qu'elle utilise l'opérateur `MATCH`.

Colonnes virtuelles

Actuellement, la requête parallèle n'est pas utilisée pour les tables qui contiennent une colonne virtuelle, que la requête se réfère ou non à des colonnes virtuelles.

Mécanismes intégrés de mise en cache

Aurora inclut des mécanismes intégrés de mise en cache, notamment le pool de mémoires tampons et le cache de requête. L'optimiseur Aurora choisit entre ces mécanismes de mise en cache et la fonction de requête parallèle selon leur efficacité pour une requête spécifique.

Lorsqu'une requête parallèle filtre les lignes et transforme et extrait les valeurs de colonne, les données sont retransmises au nœud principal sous forme de tuples au lieu de pages de données. Dès lors, l'exécution d'une requête parallèle n'ajoute aucune page au pool de mémoires tampons et n'élimine aucune page qui se trouve déjà dans ce pool.

Aurora vérifie le nombre de pages de données de table présentes dans le pool de mémoires tampons, ainsi que la proportion des données de table que ce nombre représente. Aurora utilise ces informations pour déterminer s'il est plus efficace d'utiliser une requête parallèle (et de contourner ainsi les données du pool de mémoires tampons). Aurora peut également recourir au chemin de traitement de requête non parallèle, qui utilise les données mises en cache dans le pool de mémoires tampons. Les pages mises en caches ainsi que l'impact des requêtes à usage intensif de données sur la mise en cache et l'éviction dépendent des paramètres de configuration liés au pool de mémoires tampons. Dès lors, il peut être difficile de déterminer si une requête spécifique utilisera la fonction de requête parallèle, car le choix dépend des données qui se trouvent dans le pool de mémoires tampons, lesquelles changent constamment.

Aurora impose également des limites de simultanéité pour les requêtes parallèles. Comme les requêtes n'utilisent pas toute la fonction de requête parallèle, une partie significative des données des tables interrogées par plusieurs requêtes simultanément se trouve dans le pool de mémoires tampons. Dès lors, Aurora ne choisit pas souvent ces tables pour les requêtes parallèles.

Lorsque vous exécutez une séquence de requêtes non parallèles pour la même table, la première requête peut prendre du temps, car les données ne sont pas dans le pool de mémoires tampons. Les requêtes suivantes sont beaucoup plus rapides, car le pool de mémoires tampons contient déjà des données. Les requêtes parallèles se traduisent généralement par des performances constantes entre les différentes requêtes d'une même table. Lorsque vous effectuez des tests de performance, comparez les requêtes non parallèles à la fois avec un pool de mémoires tampons à froid et à chaud. Dans certains cas, les résultats pour le pool de mémoires tampons à chaud sont comparables à ceux des requêtes parallèles. Dans ces cas de figure, tenez compte de facteurs tels que la fréquence des requêtes sur cette table. Déterminez également s'il est intéressant de conserver les données de cette table dans le pool de mémoires tampons.

Le cache de requête évite de devoir réexécuter une requête lorsqu'une requête identique est soumise et que les données de la table sous-jacente n'ont pas changé. Les requêtes optimisées par la fonction de requête parallèle peuvent être acheminées dans le cache de requête afin de fournir des résultats instantanés la prochaine fois qu'elles seront réexécutées.

> **Note**
>
> Lorsque l'on compare les performances, le cache de requête peut fournir des chiffres artificiellement bas en matière de durée. Dès lors, lorsque vous souhaitez effectuer une comparaison, vous pouvez utiliser l'indicateur `sql_no_cache`. Celui-ci empêche le résultat d'être généré par le cache de requête, même si la même requête a été exécutée précédemment. Cet indicateur vient juste après l'instruction `SELECT` dans une requête. De nombreux exemples de requêtes parallèles fournis dans cette rubrique utilisent cet indicateur afin de pouvoir comparer les délais entre les versions d'une requête pour laquelle la fonction de requête parallèle est activée ou désactivée.
> Veillez à supprimer cet indicateur du code source lorsque vous utiliserez la fonction de requête parallèle en environnement de production.

Indicateurs de l'optimiseur

Une autre façon de contrôler l'optimiseur consiste à utiliser des indices d'optimiseur, qui peuvent être spécifiés dans des instructions individuelles. Par exemple, vous pouvez activer une optimisation pour une table dans une instruction, puis désactiver l'optimisation pour une autre table. Pour plus d'informations sur ces indicateurs, consultez Optimizer Hints (Indicateurs d'optimiseur) dans le Manuel de référence MySQL.

Vous pouvez utiliser des indicateurs SQL avec des requêtes Aurora MySQL pour ajuster les performances. Vous pouvez également utiliser des indicateurs pour empêcher que les plans d'exécution des requêtes importantes ne changent en fonction de conditions imprévisibles.

Nous avons étendu la fonction d'indicateurs SQL pour vous aider à contrôler les choix d'optimiseurs pour vos plans de requêtes. Ces indicateurs s'appliquent aux requêtes qui utilisent l'optimisation via les requêtes parallèles. Pour plus d'informations, consultez Indicateurs Aurora MySQL.

Tables temporaires MyISAM

L'optimisation via les requêtes parallèles s'appliquent uniquement aux tables InnoDB. Comme Aurora MySQL utilise MyISAM en arrière-plan pour les tables temporaires, les phases de requêtes internes

impliquant des tables temporaires n’utilisent jamais la fonction de requête parallèle. Ces phases de requête sont indiquées par `Using temporary` dans la sortie `EXPLAIN`.

Utilisation de l'Audit avancé avec un cluster de bases de données Amazon Aurora MySQL

Vous pouvez utiliser la fonctionnalité d'Audit avancé hautement performante d'Amazon Aurora MySQL pour contrôler l'activité de la base de données. Pour ce faire, vous activez l'ensemble des journaux d'audit en définissant plusieurs paramètres du cluster de bases de données. Lorsque l'Audit avancé est activé, vous pouvez l'utiliser pour consigner n'importe quelle combinaison d'événements pris en charge.

Vous pouvez afficher ou télécharger les journaux d'audit pour consulter les informations d'audit d'une instance de base de données à la fois. Pour ce faire, vous pouvez utiliser les procédures présentées dans Surveillance des fichiers journaux Amazon Aurora.

> **Tip**
>
> Pour un cluster de bases de données Aurora contenant plusieurs instances de base de données, il peut être plus pratique d'examiner les journaux d'audit de toutes les instances du cluster. Pour ce faire, vous pouvez utiliser CloudWatch Logs. Vous pouvez activer un paramètre au niveau du cluster pour publier les données des journaux d'audit Aurora MySQL dans un groupe de journaux dans CloudWatch. Vous pouvez ensuite afficher, filtrer et examiner les journaux d'audit via l'interface CloudWatch. Pour plus d'informations, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs.

Activation de l'Audit avancé

Utilisez les paramètres décrits dans cette section pour activer et configurer l'Audit avancé pour votre cluster de bases de données.

Utilisez le paramètre `server_audit_logging` pour activer ou désactiver l'Audit avancé.

Utilisez le paramètre `server_audit_events` pour spécifier les événements à journaliser.

Utilisez les paramètres `server_audit_incl_users` et `server_audit_excl_users` pour spécifier les utilisateurs à auditer. Par défaut, tous les utilisateurs sont audités. Pour plus d'informations sur le fonctionnement de ces paramètres lorsque l'un ou les deux sont vides, ou que les mêmes noms d'utilisateur sont spécifiés dans les deux, consultez les sections server_audit_incl_users et server_audit_excl_users.

Configurez l'Audit avancé en définissant ces paramètres dans le groupe de paramètres utilisé par votre cluster de bases de données. Vous pouvez utiliser la procédure présentée dans Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora pour modifier les paramètres de cluster de bases de données à l'aide d'AWS Management Console. Vous pouvez utiliser la commande de l'AWS CLI modify-db-cluster-parameter-group ou l'opération d'API Amazon RDS ModifyDBClusterParameterGroup pour modifier des paramètres de cluster de bases de données par programmation.

La modification de ces paramètres n'exige pas de redémarrage du cluster de bases de données lorsque le groupe de paramètres est déjà associé à votre cluster. Lorsque vous associez le groupe de paramètres au cluster pour la première fois, un redémarrage du cluster est nécessaire.

Rubriques

- server_audit_logging
- server_audit_events
- server_audit_incl_users
- server_audit_excl_users

server_audit_logging

Active ou désactive l'Audit avancé. Par défaut, ce paramètre est défini sur OFF. Définissez-le sur ON pour activer l'Audit avancé.

Aucune donnée d'audit n'apparaît dans les journaux, à moins que vous ne définissiez également un ou plusieurs types d'événements à auditer à l'aide du paramètre `server_audit_events`.

Pour confirmer que les données d'audit sont journalisées pour une instance de base de données, vérifiez que certains fichiers journaux de cette instance soient nommés sous la forme `audit/audit.log.`*`other_identifying_information`*. Pour voir les noms des fichiers journaux, suivez la procédure présentée dans Liste et affichage des fichiers journaux de base de données.

server_audit_events

Contient la liste séparée par des virgules des événements à consigner. Les événements doivent être indiqués en lettres majuscules, et il ne doit y avoir aucun espace entre les éléments de liste, par exemple : `CONNECT,QUERY_DDL`. La valeur par défaut de ce paramètre est une chaîne vide.

Vous pouvez consigner n'importe quelle combinaison des événements suivants :

- CONNECT – Consigne les connexions réussies et celles ayant échoué, ainsi que les déconnexions. Cet événement inclut des informations utilisateur.
- QUERY – Consigne toutes les requêtes en texte brut, notamment les requêtes ayant échoué suite à des erreurs de syntaxe ou d'autorisation.

> **Tip**
>
> Lorsque ce type d'événement est activé, les données d'audit incluent des informations sur la surveillance continue et les informations de surveillance de l'état effectuées automatiquement par Aurora. Si vous ne vous intéressez qu'à des types particuliers d'opérations, vous pouvez utiliser les types d'événements les plus spécifiques. Vous pouvez également utiliser l'interface CloudWatch pour rechercher dans les journaux des événements liés à des bases de données, des tables ou des utilisateurs spécifiques.

- QUERY_DCL – Semblable à l'événement QUERY, mais renvoie uniquement les requêtes en langage de contrôle de données (DCL) (GRANT, REVOKE, etc.).
- QUERY_DDL – Semblable à l'événement QUERY, mais renvoie uniquement les requêtes en langage de définition de données (DDL) (CREATE, ALTER, etc.).
- QUERY_DML – Semblable à l'événement QUERY, mais renvoie uniquement les requêtes en langage de manipulation de données (DML) (INSERT, UPDATE, etc. ainsi que SELECT).
- TABLE – Consigne les tables affectées par l'exécution d'une requête.

> **Note**
>
> Aurora ne comporte aucun filtre qui exclut certaines requêtes des journaux d'audit. Pour exclure des requêtes `SELECT`, vous devez exclure toutes les instructions DML.
> Si un utilisateur signale ces requêtes `SELECT` internes dans les journaux d'audit, vous pouvez l'exclure en définissant le paramètre de cluster de bases de données server_audit_excl_users. Toutefois, si cet utilisateur est également utilisé dans d'autres activités et ne peut pas être omis, il n'existe aucune autre option permettant d'exclure les requêtes `SELECT`.

server_audit_incl_users

Contient la liste séparée par des virgules des noms d'utilisateur des utilisateurs dont l'activité est consignée. Il ne doit y avoir aucun espace blanc entre les éléments de la liste, par exemple : `user_3,user_4`. La valeur par défaut de ce paramètre est une chaîne vide. La longueur maximale est de 1 024 caractères. Les noms d'utilisateur spécifiés doivent correspondre aux valeurs figurant dans la colonne `User` de la table `mysql.user`. Pour plus d'informations sur les noms d'utilisateur, consultez Noms d'utilisateur et mots de passe de compte dans la documentation sur MySQL.

Si `server_audit_incl_users` et `server_audit_excl_users` sont vides (valeurs par défaut), tous les utilisateurs sont audités.

Si vous ajoutez des utilisateurs à `server_audit_incl_users` et laissez `server_audit_excl_users` vide, alors seuls ces utilisateurs sont audités.

Si vous ajoutez des utilisateurs à `server_audit_excl_users` et laissez `server_audit_incl_users` vide, alors tous les utilisateurs sont audités, à l'exception de ceux répertoriés dans `server_audit_excl_users`.

Si vous ajoutez les mêmes utilisateurs à `server_audit_excl_users` et `server_audit_incl_users`, alors ces utilisateurs sont audités. Lorsque le même utilisateur est répertorié dans les deux paramètres, `server_audit_incl_users` se voit accorder la priorité.

Les événements de connexion et de déconnexion ne sont pas affectés par cette variable ; ils sont toujours consignés, le cas échéant. Un utilisateur est journalisé même s'il est également spécifié dans le paramètre `server_audit_excl_users`, car la priorité de `server_audit_incl_users` est plus élevée.

server_audit_excl_users

Contient la liste séparée par des virgules des noms d'utilisateur des utilisateurs dont l'activité n'est pas consignée. Il ne doit y avoir aucun espace blanc entre les éléments de la liste, par exemple : `rdsadmin,user_1,user_2`. La valeur par défaut de ce paramètre est une chaîne vide. La longueur maximale est de 1 024 caractères. Les noms d'utilisateur spécifiés doivent correspondre aux valeurs figurant dans la colonne `User` de la table `mysql.user`. Pour plus d'informations sur les noms d'utilisateur, consultez Noms d'utilisateur et mots de passe de compte dans la documentation sur MySQL.

Si `server_audit_incl_users` et `server_audit_excl_users` sont vides (valeurs par défaut), tous les utilisateurs sont audités.

Si vous ajoutez des utilisateurs à `server_audit_excl_users` et laissez `server_audit_incl_users` vide, alors seuls ces utilisateurs répertoriés dans `server_audit_excl_users` ne sont pas audités, tous les autres le sont.

Si vous ajoutez les mêmes utilisateurs à `server_audit_excl_users` et `server_audit_incl_users`, alors ces utilisateurs sont audités. Lorsque le même utilisateur est répertorié dans les deux paramètres, `server_audit_incl_users` se voit accorder la priorité.

Les événements de connexion et de déconnexion ne sont pas affectés par cette variable ; ils sont toujours consignés, le cas échéant. Un utilisateur est consigné si ce dernier est également spécifié dans le paramètre `server_audit_incl_users` car celui-ci possède une priorité supérieure à `server_audit_excl_users`.

Consultation des journaux d'audit

Vous pouvez consulter et télécharger les journaux d'audit à l'aide de la console. Dans la page Bases de données, choisissez l'instance de base de données pour en afficher les détails, puis faites défiler la page jusqu'à la section Journaux. Les journaux d'audit produits par la fonction d'audit avancé sont nommés sous la forme `audit/audit.log.`*`other_identifying_information`*.

Pour télécharger un fichier journal, recherchez le fichier dans la section Journaux puis choisissez Télécharger.

Vous pouvez également obtenir la liste des fichiers journaux à l'aide de la commande describe-db-log-files de l'AWS CLI. Vous pouvez télécharger le contenu d'un fichier journal à l'aide de la commande download-db-log-file-portion de l'AWS CLI. Pour plus d'informations, consultez Liste et affichage des fichiers journaux de base de données et Téléchargement d'un fichier journal de base de données.

Détails du journal d'audit

Les fichiers journaux sont représentés sous forme de fichiers CSV (variables séparées par des virgules) au format UTF-8. Les requêtes sont également placées entre guillemets simples (').

Le journal d'audit est stocké séparément sur le stockage local de chaque instance de base de données Aurora MySQL. Chaque instance distribue les écritures entre quatre fichiers journaux à la fois. La taille maximale d'un fichier journal est de 100 Mo. Lorsque cette limite non configurable est atteinte, Aurora effectue une rotation de fichier et en génère un nouveau.

> **Tip**
>
> Les entrées de fichier journal ne sont pas classées par ordre séquentiel. Pour ordonner les entrées, utilisez la valeur d'horodatage. Pour consulter les derniers événements, vous devrez peut-être passer en revue tous les fichiers journaux. Pour plus de flexibilité dans le tri et la recherche des données de journaux, activez le paramètre pour charger les journaux d'audit sur CloudWatch et les afficher à l'aide de l'interface CloudWatch.
> Pour afficher des données d'audit avec plus de types de champs et avec une sortie au format JSON, vous pouvez également utiliser la fonction Flux d'activité de base de données. Pour plus d'informations, consultez Surveillance d'Amazon Aurora à l'aide des flux d'activité de base de données.

Les fichiers journaux d'audit incluent les informations séparées par des virgules suivantes en lignes, dans l'ordre indiqué :

Champ	Description
timestamp	L'horodatage Unix pour l'événement consigné avec une précision à la microseconde.
serverhost	Le nom de l'instance pour laquelle*** l'événement est consigné.
username	Le nom d'utilisateur connecté de l'utilisateur.
hôte	L'hôte à partir duquel** l'utilisateur s'est connecté.
connectionid	Le numéro d'identification de la connexion pour l'opération consignée.
queryid	Le numéro d'identification de la requête qui peut être utilisé pour trouver les événements de la table relationnelle et les requêtes liées. Pour les événements `TABLE`, plusieurs lignes sont ajoutées.
fonctionnement	Le type d'action enregistrée. Les valeurs possibles sont : `CONNECT`, `QUERY`, `READ`, `WRITE`, `CREATE`, `ALTER`, `RENAME` et `DROP`.
database	La base de données active, telle que définie par la commande `USE`.

Champ	Description
objet	Pour les événements `QUERY`, cette valeur indique la demande effectuée par la base de données. Pour les événements `TABLE`, cette valeur indique le nom de la table.
retcode	Le code de retour de l’opération consignée.

Réplication avec Amazon Aurora MySQL

Les fonctions de réplication d'Aurora MySQL sont essentielles pour garantir la haute disponibilité et les performances de votre cluster. Aurora permet de créer ou de redimensionner facilement des clusters avec jusqu'à 15 réplicas Aurora.

Tous les réplicas fonctionnent à partir des mêmes données sous-jacentes. Si certaines instances de base de données passent hors connexion, les autres restent disponibles pour continuer à traiter les requêtes ou pour prendre la relève en tant qu'enregistreur si nécessaire. Aurora répartit automatiquement vos connexions en lecture seule entre plusieurs instances de base de données, ce qui permet à un cluster de prendre en charge des charges de travail exigeantes en requêtes.

Dans les rubriques suivantes, vous trouverez des informations sur la manière dont la réplication Aurora MySQL fonctionne, ainsi que sur le réglage des paramètres de réplication pour garantir une disponibilité et des performances optimales.

Rubriques

Utilisation de réplicas Aurora

Les réplicas Aurora sont les points de terminaison indépendants d'un cluster de bases de données Aurora, utilisés de préférence pour le dimensionnement des opérations de lecture et l'augmentation de la disponibilité. Le nombre de réplicas Aurora pouvant être distribués entre les zones de disponibilité que couvre un cluster de bases de données au sein d'une Région AWS est limité à 15. Même si le volume du cluster de bases de données est composé de plusieurs copies des données

pour le cluster de bases de données, les données du volume de cluster sont représentées comme un seul volume logique à l'instance principale et aux réplicas Aurora du cluster de bases de données. Pour plus d'informations sur les réplicas Aurora, consultez Réplicas Aurora.

Les réplicas Aurora fonctionnement parfaitement pour le dimensionnement en lecture, car ils sont intégralement dédiés aux opérations de lecture de votre volume de cluster. Les opérations d'écriture sont gérées par l'instance principale. Comme le volume de cluster est partagé entre toutes les instances de votre cluster de bases de données Aurora MySQL, aucun travail supplémentaire n'est nécessaire pour répliquer une copie des données pour chaque réplica Aurora. En revanche, les réplicas en lecture MySQL doivent relire, sur un seul thread, toutes les opérations d'écriture depuis l'instance de base de données source vers leur magasin de données local. Cette limitation peut affecter la capacité des réplicas en lecture MySQL à prendre en charge d'importants volumes de trafic en lecture.

Avec Aurora MySQL, quand un réplica Aurora est supprimé, le point de terminaison de son instance est supprimé immédiatement et le réplica Aurora est supprimé du point de terminaison du lecteur. S'il y a des instructions qui s'exécutent sur le réplica Aurora en cours de suppression, une période de grâce de trois minutes est accordée. Les instructions existantes peuvent se terminer élégamment pendant la période de grâce. Lorsque la période de grâce se termine, le réplica Aurora est arrêté et supprimé.

Important

Les réplicas Aurora pour Aurora MySQL utilisent toujours le niveau d'isolation de transaction par défaut `REPEATABLE READ` pour les opérations sur les tables InnoDB. Vous pouvez utiliser la commande `SET TRANSACTION ISOLATION LEVEL` pour modifier uniquement le niveau de transaction de l'instance principale d'un cluster de bases de données Aurora MySQL. Cette restriction évite les verrous de niveau utilisateur sur les réplicas Aurora et permet aux réplicas Aurora d'évoluer pour prendre en charge des milliers de connexions utilisateur actives tout en maintenant le retard du réplica à une valeur minimale.

Note

Les instructions DDL exécutées sur l'instance principale peuvent interrompre les connexions de base de données sur les réplicas Aurora associés. Si une connexion de réplica Aurora utilise de manière active un objet de base de données, par exemple une table, et que cet

objet est modifié sur l'instance principale à l'aide d'une instruction DDL, la connexion du réplica Aurora est interrompue.

> **Note**
>
> La région Chine (Ningxia) ne prend pas en charge les réplicas en lecture entre régions.

Options de réplication pour Amazon Aurora MySQL

Vous pouvez configurer la réplication entre n'importe lesquelles des options suivantes :

- Deux clusters de bases de données Aurora MySQL dans des Régions AWS différentes, en créant un réplica en lecture entre régions d'un cluster de bases de données Aurora MySQL.

  Pour de plus amples informations, consultez Réplication de clusters de bases de données Amazon Aurora MySQL dans différentes Régions AWS.

- Deux clusters de bases de données Aurora MySQL dans la même Région AWS, en utilisant la réplication du journal binaire (binlog) MySQL.

  Pour de plus amples informations, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

- Une instance source de bases de données RDS for MySQL et un cluster de bases de données Aurora MySQL, en créant un réplica en lecture Aurora d'une instance de bases de données RDS for MySQL.

  Vous pouvez utiliser cette approche pour intégrer les modifications de données existantes et en cours à Aurora MySQL lors de la migration vers Aurora. Pour de plus amples informations, consultez Migration des données d'une instance de base de données RDS for MySQL vers un cluster de bases de données Amazon Aurora MySQL à l'aide d'un réplica en lecture Aurora.

  Vous pouvez également utiliser cette approche pour augmenter la scalabilité des requêtes en lecture de vos données. Pour ce faire, interrogez les données à l'aide d'une ou de plusieurs instances de base de données dans un cluster Aurora MySQL en lecture seule. Pour de plus amples informations, consultez Mise à l'échelle des lectures pour votre base de données MySQL avec Amazon Aurora.

- Un cluster de bases de données Aurora MySQL dans une Région AWS et jusqu'à cinq clusters de bases de données Aurora MySQL en lecture seule Aurora dans différentes régions, en créant une base de données Aurora globale.

  Vous pouvez utiliser une base de données Aurora globale pour prendre en charge des applications ayant une présence mondiale. Le cluster de bases de données Aurora MySQL principal possède une instance de scripteur et jusqu'à 15 réplicas Aurora. Les clusters de bases de données Aurora MySQL secondaires en lecture seule peuvent être composés de 16 réplicas Aurora maximum. Pour plus d'informations, consultez Utilisation d'Amazon Aurora Global Database.

> **Note**
>
> Le redémarrage de l'instance principale d'un cluster de bases de données Amazon Aurora entraîne automatiquement le redémarrage des réplicas Aurora pour ce cluster de bases de données afin de ré-établir un point d'entrée garantissant la cohérence en lecture/écriture dans le cluster de bases de données.

Considérations sur les performances pour la réplication Amazon Aurora MySQL

Les fonctions suivantes vous permettent d'affiner les performances d'une réplication Aurora MySQL.

La fonction de compression des journaux de réplica réduit automatiquement la bande passante réseau pour les messages de réplication. Étant donné que chaque message est transmis à tous les réplicas Aurora, les avantages sont supérieurs pour les clusters plus volumineux. Cette fonction implique une certaine surcharge de l'UC sur le nœud d'enregistreur pour effectuer la compression. Elle est toujours activée dans Aurora MySQL version 2 et 3.

La fonction de filtrage des journaux binaires réduit automatiquement la bande passante réseau pour les messages de réplication. Étant donné que les réplicas Aurora n'utilisent pas les informations de journal binaire qui sont incluses dans les messages de réplication, ces données sont omises dans les messages envoyés à ces nœuds.

Dans Aurora MySQL version 2, vous pouvez contrôler cette fonctionnalité en modifiant le paramètre `aurora_enable_repl_bin_log_filtering`. Ce paramètre est activé par défaut. Étant donné que cette optimisation est conçue pour être transparente, vous pouvez désactiver ce paramètre uniquement pendant le diagnostic ou le dépannage des problèmes liés à la réplication, par exemple

pour correspondre au comportement d'un ancien cluster Aurora MySQL où cette fonction n'était pas disponible.

Le filtrage des journaux binaires est toujours activé dans Aurora MySQL version 3.

Configuration des filtres de réplication avec Aurora MySQL

Vous pouvez utiliser des filtres de réplication pour spécifier quelles bases de données et tables sont répliquées avec un réplica en lecture. Les filtres de réplication peuvent inclure des bases de données et des tables dans la réplication ou les exclure de la réplication.

Voici quelques cas d'utilisation pour les filtres de réplication :

- Pour réduire la taille d'un réplica en lecture. Avec le filtrage de réplication, vous pouvez exclure les bases de données et les tables qui ne sont pas nécessaires sur le réplica en lecture.
- Pour exclure des bases de données et des tables des réplicas en lecture, pour des raisons de sécurité.
- Pour répliquer différentes bases de données et tables pour des cas d'utilisation spécifiques au niveau de différents réplicas en lecture. Par exemple, vous pouvez utiliser des réplicas en lecture spécifiques pour l'analyse ou le partage.
- Pour un cluster de bases de données qui a des réplicas en lecture dans différentes Régions AWS, pour répliquer différentes bases de données ou tables dans des Régions AWS différentes.
- Pour spécifier quelles bases de données et tables sont répliquées avec un cluster de bases de données Aurora MySQL configuré en tant que réplica dans une topologie de réplication entrante. Pour en savoir plus sur cette configuration, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

Rubriques

Définition des paramètres de filtrage de la réplication pour Aurora MySQL

Pour configurer des filtres de réplication, définissez les paramètres suivants :

- `binlog-do-db` : répliquer les modifications apportées aux journaux binaires spécifiés. Lorsque vous définissez ce paramètre pour un cluster source de journaux binaires, seuls les journaux binaires spécifiés dans le paramètre sont répliqués.
- `binlog-ignore-db` : ne pas répliquer les modifications apportées aux journaux binaires spécifiés. Lorsque le paramètre `binlog-do-db` est défini pour un cluster source de journaux binaires, ce paramètre n'est pas évalué.
- `replicate-do-db` – Répliquer les modifications apportées aux bases de données spécifiées. Lorsque vous définissez ce paramètre pour un cluster de réplica de journaux binaires, seules les bases de données spécifiées dans le paramètre sont répliquées.
- `replicate-ignore-db` – Ne pas répliquer les modifications apportées aux bases de données spécifiées. Lorsque le paramètre `replicate-do-db` est défini pour un cluster de réplica de journaux binaires, ce paramètre n'est pas évalué.
- `replicate-do-table` – Répliquer les modifications apportées aux tables spécifiées. Lorsque vous définissez ce paramètre pour un réplica en lecture, seules les tables spécifiées dans le paramètre sont répliquées. En outre, lorsque le paramètre `replicate-do-db` ou `replicate-ignore-db` est défini, assurez-vous d'inclure la base de données qui comprend les tables spécifiées dans la réplication avec le cluster de réplica des journaux binaires.
- `replicate-ignore-table` – Ne pas répliquer les modifications apportées aux tables spécifiées. Lorsque le paramètre `replicate-do-table` est défini pour un cluster de réplica de journaux binaires, ce paramètre n'est pas évalué.
- `replicate-wild-do-table` – Répliquer les tables en fonction des modèles de nom de base de données et nom de table spécifiés. Les caractères génériques `%` et `_` sont pris en charge. Lorsque le paramètre `replicate-do-db` ou `replicate-ignore-db` est défini, assurez-vous d'inclure la base de données qui comprend les tables spécifiées dans la réplication avec le cluster de réplica des journaux binaires.
- `replicate-wild-ignore-table` – Ne pas répliquer les tables en fonction des modèles de nom de base de données et de nom de table spécifiés. Les caractères génériques `%` et `_` sont pris en charge. Lorsque le paramètre `replicate-do-table` ou `replicate-wild-do-table` est défini pour un cluster de réplica de journaux binaires, ce paramètre n'est pas évalué.

Les paramètres sont évalués dans l'ordre dans lequel ils sont répertoriés. Pour plus d'informations sur le fonctionnement de ces paramètres, consultez la documentation MySQL :

- Pour plus d'informations générales, voir Options et variables du serveur de réplication.

- Pour plus d'informations sur la façon dont les paramètres de filtrage de réplication de base de données sont évalués, voir Évaluation des options de réplication au niveau de la base de données et des options de la journalisation binaire.
- Pour plus d'informations sur l'évaluation des paramètres de filtrage de réplication de table, reportez-vous à la section Évaluation des options de réplication au niveau de la table.

Par défaut, chacun de ces paramètres a une valeur vide. Sur chaque cluster de journaux binaires, vous pouvez utiliser ces paramètres pour définir, modifier et supprimer des filtres de réplication. Lorsque vous définissez l'un de ces paramètres, séparez chaque filtre des autres par une virgule.

Vous pouvez utiliser les caractères génériques `%` et _ dans les paramètres `replicate-wild-do-table` et `replicate-wild-ignore-table`. Le caractère générique `%` correspond à un nombre quelconque de caractères, et le caractère générique _ ne correspond qu'à un seul caractère.

Le format de journalisation binaire de l'instance de base de données source est important pour la réplication, car il détermine l'enregistrement des modifications de données. Le réglage du paramètre `binlog_format` détermine si la réplication est basée sur les lignes ou les instructions. Pour plus d'informations, consultez Configuration d'Aurora MySQL pour la journalisation des bases de données mono-AZ.

> **Note**
>
> Toutes les instructions DDL (Data Definition Language) sont répliquées en tant qu'instructions, quel que soit le paramètre `binlog_format` de l'instance de base de données source.

Limites du filtrage de réplication pour Aurora MySQL

Les limites suivantes s'appliquent au filtrage de réplication pour Aurora MySQL :

- Les filtres de réplication sont uniquement pris charge pour Aurora MySQL version 3.
- Chaque paramètre de filtrage de réplication a une limite de 2 000 caractères.
- Les virgules ne sont pas prises en charge dans les filtres de réplication.
- Le filtrage de réplication ne prend pas en charge les transactions XA.

  Pour plus d'informations, consultez Restrictions on XA Transactions (Restrictions sur les transactions XA) dans la documentation MySQL.

Exemples de filtrage de réplication pour Aurora MySQL

Pour configurer le filtrage de réplication pour un réplica en lecture, modifiez les paramètres de filtrage de réplication dans le groupe de paramètres de cluster de bases de données associé au réplica en lecture.

> **Note**
>
> Vous ne pouvez pas modifier un groupe de paramètres de cluster de bases de données par défaut. Si le réplica en lecture utilise un groupe de paramètres par défaut, créez un nouveau groupe de paramètres et associez-le au réplica en lecture. Pour plus d'informations sur les groupes de paramètres de cluster de bases de données, consultez Groupes de paramètres pour Amazon Aurora.

Vous pouvez définir des paramètres dans un groupe de paramètres de cluster de bases de données à l'aide de la AWS Management Console, de l'interface AWS CLI ou de l'API RDS. Pour plus d'informations sur la définition des paramètres, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora. Lorsque vous définissez des paramètres dans un groupe de paramètres de cluster de bases de données, tous les clusters de bases de données associés au groupe de paramètres utilisent les réglages des paramètres. Si vous définissez les paramètres de filtrage de réplication dans un groupe de paramètres de cluster de bases de données, assurez-vous que le groupe de paramètres est associé uniquement aux clusters de réplicas en lecture. Laissez les paramètres de filtrage de réplication vides pour les instances de base de données source.

Les exemples suivants définissent les paramètres à l'aide de la AWS CLI. Ces exemples définissent `ApplyMethod` sur `immediate` de sorte que les modifications de paramètre se produisent immédiatement après la fin de la commande de la CLI. Si vous souhaitez qu'une modification en attente soit appliquée après le redémarrage du réplica en lecture, définissez `ApplyMethod` sur `pending-reboot`.

Les exemples suivants définissent des filtres de réplication :

- Including databases in replication
- Including tables in replication
- Including tables in replication with wildcard characters
- Excluding databases from replication

- Excluding tables from replication
- Excluding tables from replication using wildcard characters

Example Inclusion de bases de données dans la réplication

L'exemple suivant inclut les bases de données `mydb1` et `mydb2` dans la réplication.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
db,ParameterValue='mydb1,mydb2',ApplyMethod=immediate"
```

Example Inclusion de tables dans la réplication

L'exemple suivant inclut les tables `table1` et `table2` dans la base de données `mydb1` dans la réplication.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-do-
table,ParameterValue='mydb1.table1,mydb1.table2',ApplyMethod=immediate"
```

Example Inclusion de tables dans la réplication à l'aide de caractères génériques

L'exemple suivant inclut des tables dont les noms commencent par `order` et `return` dans la base de données `mydb` dans la réplication.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-do-table,ParameterValue='mydb.order
%,mydb.return%',ApplyMethod=immediate"
```

Example Exclusion de bases de données de la réplication

L'exemple suivant exclut les bases de données `mydb5` et `mydb6` de la réplication.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6',ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
db,ParameterValue='mydb5,mydb6,ApplyMethod=immediate"
```

Example Exclusion de tables de la réplication

L'exemple suivant exclut les tables `table1` dans la base de données `mydb5` et `table2` dans la base de données `mydb6` de la réplication.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-ignore-
table,ParameterValue='mydb5.table1,mydb6.table2',ApplyMethod=immediate"
```

Example Exclusion de tables de la réplication à l'aide des caractères génériques

L'exemple suivant exclut de la réplication les tables dont les noms commencent par `order` et `return` dans la base de données `mydb7`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
  --db-cluster-parameter-group-name myparametergroup \
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
  --db-cluster-parameter-group-name myparametergroup ^
  --parameters "ParameterName=replicate-wild-ignore-table,ParameterValue='mydb7.order
%,mydb7.return%',ApplyMethod=immediate"
```

Affichage des filtres de réplication pour un réplica en lecture

Vous pouvez afficher les filtres de réplication pour un réplica en lecture de la manière suivante :

- Vérifiez les réglages des paramètres de filtrage de réplication dans le groupe de paramètres associé au réplica en lecture.

Pour obtenir des instructions, consultez Affichage des valeurs de paramètres pour un groupe de paramètres de base de données dans Amazon Aurora.

- Dans un client MySQL, connectez-vous au réplica en lecture et exécutez l'instruction `SHOW REPLICA STATUS`.

  Dans la sortie, les champs suivants affichent les filtres de réplication pour le réplica en lecture :

  - `Binlog_Do_DB`
  - `Binlog_Ignore_DB`
  - `Replicate_Do_DB`
  - `Replicate_Ignore_DB`
  - `Replicate_Do_Table`
  - `Replicate_Ignore_Table`
  - `Replicate_Wild_Do_Table`
  - `Replicate_Wild_Ignore_Table`

  Pour plus d'informations sur ces champs, consultez Vérification du statut de la réplication dans la documentation MySQL.

Surveillance de la réplication Amazon Aurora MySQL

Le dimensionnement en lecture et la haute disponibilité dépendent d'un temps de retard minimal. Vous pouvez surveiller jusqu'à quel point un réplica Aurora est en retard par rapport à l'instance principale de votre cluster de bases de données Aurora MySQL en surveillant la métrique Amazon CloudWatch `AuroraReplicaLag`. La métrique `AuroraReplicaLag` est enregistrée dans chaque réplica Aurora.

L'instance de base de données principale enregistre également les métriques Amazon CloudWatch `AuroraReplicaLagMaximum` et `AuroraReplicaLagMinimum`. La métrique `AuroraReplicaLagMaximum` enregistre le décalage maximal entre l'instance de base de données principale et chaque réplica Aurora dans le cluster de bases de données. La métrique `AuroraReplicaLagMinimum` enregistre le décalage minimal entre l'instance de base de données principale et chaque réplica Aurora dans le cluster de bases de données.

Si vous avez besoin de la valeur la plus actuelle du retard de réplica Aurora, vous pouvez consulter la métrique `AuroraReplicaLag` dans Amazon CloudWatch. Le retard de réplica Aurora est également

enregistré sur chaque réplica Aurora de votre cluster de bases de données Aurora MySQL dans la table `information_schema.replica_host_status`. Pour plus d'informations sur cette table, consultez information_schema.replica_host_status.

Pour plus d'informations sur la surveillance des instances RDS et des métriques CloudWatch, consultez Surveillance des métriques d’un cluster de bases de données Amazon Aurora.

Réplication de clusters de bases de données Amazon Aurora MySQL dans différentes Régions AWS

Vous pouvez créer un cluster de bases de données Amazon Aurora MySQL en tant que réplica en lecture dans une Région AWS autre que celle du cluster de bases de données source. Cette approche vous permet d'améliorer vos capacités de reprise après sinistre, de faire évoluer les opérations de lecture dans une Région AWS qui est plus proche de vos utilisateurs et de faciliter la migration d'une Région AWS à une autre.

Vous pouvez créer des réplicas en lecture pour les clusters de bases de données chiffrés et non chiffrés. Le réplica en lecture doit être chiffré si le cluster de bases de données source est chiffré.

Pour chaque cluster de bases de données source, vous pouvez avoir jusqu'à cinq clusters de bases de données entre régions qui sont des réplicas en lecture.

> **Note**
>
> Comme alternative aux réplicas en lecture entre régions, vous pouvez mettre à l'échelle les opérations de lecture avec un temps de latence minimal à l'aide d'une base de données Aurora globale. Une base de données Aurora globale a un cluster de bases de données Aurora principal dans une Région AWS, et jusqu'à 10 clusters de bases de données secondaires en lecture seule dans d'autres régions. Chaque cluster de bases de données secondaire peut inclure jusqu'à 16 réplicas Aurora (plutôt que 15). La réplication du cluster de bases de données primaire vers tous les clusters secondaires est gérée par la couche de stockage Aurora plutôt que par le moteur de base de données. Ainsi, le temps de latence de réplication des modifications est minime (généralement inférieur à 1 seconde). Exclure le moteur de base de données du processus de réplication permet de le dédier au traitement des charges de travail. En outre, vous n'avez pas besoin de configurer ou de gérer la réplication binlog (journalisation binaire) d'Aurora MySQL. Pour en savoir plus, consultez Utilisation d'Amazon Aurora Global Database.

Lorsque vous créez un réplica en lecture de cluster de bases de données Aurora MySQL dans une autre Région AWS, vous devez être conscient des points suivants :

- Votre cluster de bases de données source et votre cluster de bases de données de réplica en lecture entre régions peuvent avoir jusqu'à 15 réplicas Aurora, avec l'instance principale du

cluster de bases de données. En utilisant cette fonctionnalité, vous pouvez mettre à l'échelle les opérations de lecture pour votre Région AWS source et votre Région AWS cible de réplication.

- Dans un scénario entre régions, le retard entre le cluster de bases de données source et le réplica en lecture est plus important du fait que les canaux de réseau entre les Régions AWS sont plus longs.
- Les données transférées pour la réplication entre régions génèrent des frais de transfert de données Amazon RDS. Les actions de réplication entre régions suivantes génèrent des frais pour les données transférées hors de la Région AWS source :
  - Lorsque vous créez le réplica en lecture, Amazon RDS prend un instantané du cluster source et transfère cet instantané vers la Région AWS où se trouve le réplica en lecture.
  - Pour chaque modification des données effectuée dans les bases de données source, Amazon RDS transfère les données de la région source vers la Région AWS où se trouve le réplica en lecture.

  Pour plus d'informations sur la tarification du transfert de données Amazon RDS, consultez Tarification Amazon Aurora.
- Vous pouvez exécuter plusieurs actions de création ou de suppression simultanées pour les réplicas en lecture qui référencent le même cluster de bases de données source. Vous devez toutefois rester dans la limite de cinq réplicas en lecture pour chaque cluster de bases de données source.
- Pour que la réplication fonctionne de façon efficace, chaque réplica en lecture doit avoir la même quantité de ressources de calcul et de stockage que le cluster de bases de données source. Si vous mettez à l'échelle le cluster de bases de données source, vous devez également le faire pour les réplicas en lecture.

Rubriques

- Avant de commencer
- Création d'un cluster de bases de données de réplica en lecture entre régions pour Aurora MySQL
- Promotion d'un réplica en lecture en cluster de bases de données pour Aurora MySQL
- Dépannage des réplicas entre régions pour Amazon Aurora MySQL

Avant de commencer

Pour pouvoir créer un cluster de bases de données Aurora MySQL en tant que réplica en lecture entre régions, vous devez commencer par activer la journalisation binaire sur votre cluster de bases de données Aurora MySQL source. La réplication entre régions pour Aurora MySQL utilise la réplication binaire MySQL pour relire les modifications apportées au cluster de bases de données du réplica en lecture entre régions.

Pour activer la journalisation binaire sur un cluster de bases de données Aurora MySQL, mettez à jour le paramètre `binlog_format` de votre cluster de bases de données source. Le paramètre `binlog_format` est un paramètre de niveau cluster qui figure dans le groupe de paramètres de cluster par défaut. Si votre cluster de bases de données utilise le groupe de paramètres de cluster de bases de données par défaut, créez un nouveau groupe de paramètres de cluster de bases de données pour modifier les paramètres `binlog_format`. Nous vous recommandons de définir le `binlog_format` sur `MIXED`. Cependant, vous pouvez également définir `binlog_format` sur `ROW` ou `STATEMENT` si vous avez besoin d'un format binlog spécifique. Redémarrez votre cluster de bases de données Aurora pour que les modifications prennent effet.

Pour plus d'informations sur l'utilisation de la journalisation binaire Aurora, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires). Pour plus d'informations sur la modification des paramètres de configuration de Aurora MySQL, consultez Paramètres de cluster de bases de données et d'instance de base de données Amazon Aurora et Groupes de paramètres pour Amazon Aurora.

Création d'un cluster de bases de données de réplica en lecture entre régions pour Aurora MySQL

Vous pouvez créer un cluster de bases de données Aurora en tant que réplica en lecture entre régions à l'aide de l'AWS Management Console, de l'AWS Command Line Interface (AWS CLI) ou de l'API Amazon RDS. Vous pouvez créer des réplicas en lecture entre régions à partir des clusters de bases de données chiffrés et non chiffrés.

Lorsque vous créez un réplica en lecture entre régions pour Aurora MySQL à l'aide de l'AWS Management Console, Amazon RDS crée un cluster de bases de données dans la Région AWS cible, puis crée automatiquement une instance de base de données qui est l'instance principale de ce cluster de bases de données.

Lorsque vous créez un réplica en lecture entre régions à l'aide de la AWS CLI ou de l'API RDS, vous devez commencer par créer le cluster de bases de données dans la Région AWS cible, puis attendre

qu'il devienne actif. Une fois qu'il est actif, vous pouvez alors créer une instance de base de données qui est l'instance principale de ce cluster de bases de données.

La réplication commence lorsque l'instance principale du cluster de bases de données de réplica en lecture devient disponible.

Procédez comme suit pour créer un réplica en lecture entre régions à partir d'un cluster de bases de données Aurora MySQL. Ces procédures permettent de créer des réplicas en lecture à partir de clusters de bases de données chiffrés ou non chiffrés.

Console

Pour créer un cluster de bases de données Aurora MySQL en tant que réplica en lecture entre régions avec l'AWS Management Console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le coin supérieur droit d'AWS Management Console, sélectionnez la Région AWS qui héberge votre cluster de bases de données source.
3. Dans la panneau de navigation, choisissez Bases de données.
4. Sélectionnez le cluster de bases de données pour lequel vous souhaitez créer un réplica en lecture entre régions.
5. Pour Actions, choisissez Create cross-Region read replica (Créer un réplica en lecture entre régions).
6. Sur la page Créer un réplica en lecture entre régions, choisissez les paramètres d'option de votre cluster de bases de données de réplica en lecture entre régions, comme décrit dans le tableau suivant.

Option	Description
Région de destination	Choisissez la Région AWS qui hébergera le nouveau cluster de bases de données de réplica en lecture entre régions.
Groupe de sous-réseaux de base de données de destination	Sélectionnez le groupe de sous-réseaux de base de données à utiliser pour le cluster de bases de données de réplica en lecture entre régions.

Option	Description
Accessible publiquement	Choisissez Oui pour attribuer une adresse IP publique au cluster de bases de données de réplica en lecture entre régions ; sinon, sélectionnez Non.
Chiffrement	Sélectionnez Enable Encryption (Activer le chiffrement) pour activer le chiffrement au repos pour ce cluster de bases de données. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.
AWS KMS key	Disponible uniquement si l'option Chiffrement est définie sur Activer le chiffrement. Sélectionnez la AWS KMS key à utiliser pour le chiffrement de ce cluster de bases de données. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.
Classe d'instance de base de données	Choisissez une classe d'instance de base de données qui définit les exigences de mémoire et de traitement pour l'instance principale du cluster de bases de données. Pour plus d'informations sur les options de classe d'instance de base de données, consultez Classes d'instance de base de données Amazon Aurora.
déploiement multi-AZ	Sélectionnez Yes (Oui) pour créer un réplica en lecture du nouveau cluster de bases de données dans une autre zone de disponibilité de la Région AWS cible pour la prise en charge du basculement. Pour plus d'informations sur les zones de disponibilité multiples, consultez Régions et zones de disponibilité.
Source du réplica en lecture	Choisissez le cluster de bases de données source pour lequel créer un réplica en lecture entre régions.

Option	Description
Identifiant d’instance de base de données	Attribuez un nom à l’instance principale de votre cluster de bases de données de réplica en lecture entre régions. Cet identifiant est utilisé dans l’adresse de point de terminaison de l’instance principale du nouveau cluster de bases de données. L’identifiant d’instance de base de données obéit aux contraintes suivantes : • Il doit contenir entre 1 et 63 caractères alphanumériques ou traits d’union. • Son premier caractère doit être une lettre. • Il ne peut pas se terminer par un trait d’union ou contenir deux traits d’union consécutifs. • Il doit être unique pour toutes les instances de bases de données de chaque Compte AWS, pour chaque Région AWS. Étant donné que le cluster de bases de données de réplica en lecture entre régions est créé à partir d’un instantané du cluster de bases de données source, le nom utilisateur et le mot de passe principaux du réplica en lecture sont les mêmes que ceux du cluster de bases de données source.

Option	Description
Identifiant du cluster de bases de données	Attribuez un nom à votre cluster de bases de données de réplica en lecture entre régions qui est unique pour votre compte dans la Région AWS cible de votre réplica. Cet identifiant est utilisé dans l'adresse de point de terminaison de votre cluster de bases de données. Pour plus d'informations sur le point de terminaison de cluster, consultez Connexions de point de terminaison Amazon Aurora. L'identifiant de cluster de bases de données obéit aux contraintes suivantes : • Il doit contenir entre 1 et 63 caractères alphanumériques ou traits d'union. • Son premier caractère doit être une lettre. • Il ne peut pas se terminer par un trait d'union ou contenir deux traits d'union consécutifs. • Il doit être unique pour tous les clusters de bases de données de chaque Compte AWS, pour chaque Région AWS.
Priorité	Sélectionnez une priorité de basculement pour l'instance principale du nouveau cluster de bases de données. Cette priorité détermine l'ordre dans lequel les réplicas Aurora sont promus lors de la reprise après une défaillance de l'instance principale. Si vous ne sélectionnez pas de valeur, la valeur par défaut est tier-1. Pour plus d'informations, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.

Option	Description
Port de la base de données	Spécifiez le port utilisé par les applications et les utilitaires pour accéder à la base de données. Les clusters de bases de données Aurora utilisent par défaut le port MySQL 3306. Les pare-feux de certaines entreprises bloquent les connexions vers ce port. Si le pare-feu de votre entreprise bloque le port par défaut, choisissez un autre port pour le nouveau cluster DB.
Surveillance améliorée	Choisissez Enable enhanced monitoring (Activer la surveillance améliorée) pour activer la collecte de métriques en temps réel pour le système d'exploit ation sur lequel votre cluster de bases de données s'exécute. Pour plus d'informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.
Monitoring Role (Rôle de surveilla nce)	Disponible uniquement si l'option Surveillance améliorée est définie sur Activer la surveillance améliorée. Choisissez le rôle IAM que vous avez créé pour autoriser Amazon RDS à communiquer avec Amazon CloudWatch Logs en votre nom ou choisisse z Par défaut pour que RDS crée automatiquement un rôle nommé `rds-monitoring-role` . Pour plus d'informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.
Granularité	Disponible uniquement si l'option Surveillance améliorée est définie sur Activer la surveillance améliorée. Définissez l'intervalle, en secondes, entre les recueils des métriques pour votre cluster DB.

Option	Description
Mise à niveau automatique de versions mineures	Ce paramètre ne s'applique pas aux clusters de bases de données Aurora MySQL. Pour plus d'informations sur les mises à jour de moteur pour Aurora MySQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL.

7. Choisissez Créer pour créer votre réplica en lecture entre régions pour Aurora.

AWS CLI

Pour créer un cluster de bases de données Aurora MySQL en tant que réplica en lecture entre régions avec la CLI

1. Appelez la commande AWS CLI create-db-cluster dans la Région AWS où vous voulez créer le cluster de bases de données de réplica en lecture. Incluez l'option `--replication-source-identifier` et indiquez l'Amazon Resource Name (ARN) du cluster de bases de données source pour lequel vous créez un réplica en lecture.

   Pour une réplication entre régions où le cluster de bases de données identifié par `--replication-source-identifier` est chiffré, spécifiez les options `--kms-key-id` et `--storage-encrypted`.

   > **Note**
   >
   > Vous pouvez configurer la réplication entre régions à partir d'un cluster de bases de données non chiffré vers un réplica en lecture chiffré en spécifiant `--storage-encrypted` et en fournissant une valeur pour l'option `--kms-key-id`.

   Vous ne pouvez pas spécifier les paramètres `--master-username` et `--master-user-password`. Ces valeurs sont extraites du cluster de bases de données source.

   L'exemple de code suivant crée un réplica en lecture dans la région us-east-1 à partir d'un instantané du cluster de bases de données non chiffré dans la région us-west-2. La commande

est appelée dans la région us-east-1. Cet exemple spécifie l'option `--manage-master-user-password` permettant de générer le mot de passe de l'utilisateur principal et de le gérer dans Secrets Manager. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager. Vous pouvez également utiliser l'option `--master-password` pour spécifier et gérer vous-même le mot de passe.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.08.0 \
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster
```

Pour Windows :

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora-mysql ^
  --engine-version 8.0.mysql_aurora.3.08.0 ^
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster
```

L'exemple de code suivant crée un réplica en lecture dans la région us-east-1 à partir d'un instantané du cluster de bases de données chiffré dans la région us-west-2. La commande est appelée dans la région us-east-1.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
  --db-cluster-identifier sample-replica-cluster \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.08.0 \
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster \
  --kms-key-id my-us-east-1-key \
  --storage-encrypted
```

Pour Windows :

```
aws rds create-db-cluster ^
  --db-cluster-identifier sample-replica-cluster ^
  --engine aurora-mysql ^
  --engine-version 8.0.mysql_aurora.3.08.0 ^
  --replication-source-identifier arn:aws:rds:us-
west-2:123456789012:cluster:sample-master-cluster ^
  --kms-key-id my-us-east-1-key ^
  --storage-encrypted
```

L'option `--source-region` est obligatoire pour la réplication entre les régions AWS GovCloud (US-East) et AWS GovCloud (US-West), où le cluster de bases de données identifié par `--replication-source-identifier` est chiffré. Pour `--source-region`, spécifiez la Région AWS du cluster de bases de données source.

Si `--source-region` n'est pas spécifié, spécifiez une valeur `--pre-signed-url`. Une URL présignée est une URL qui contient une demande signée via Signature Version 4 pour la commande `create-db-cluster` qui est appelée dans la Région AWS source. Pour en savoir plus sur l'option `pre-signed-url`, consultez create-db-cluster dans le manuel AWS CLI Command Reference.

2. Vérifiez que le cluster de bases de données est disponible pour être utilisé à l'aide de la commande AWS CLI describe-db-clusters, comme indiqué dans l'exemple suivant.

```
aws rds describe-db-clusters --db-cluster-identifier sample-replica-cluster
```

Lorsque les résultats **`describe-db-clusters`** affichent le statut `available`, créez l'instance principale pour le cluster de bases de données afin que la réplication commence. Pour ce faire, utilisez la commande AWS CLI create-db-instance, comme illustré dans l'exemple suivant.

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance \
  --db-cluster-identifier sample-replica-cluster \
  --db-instance-class db.r5.large \
  --db-instance-identifier sample-replica-instance \
  --engine aurora-mysql
```

Pour Windows :

```
aws rds create-db-instance ^
  --db-cluster-identifier sample-replica-cluster ^
  --db-instance-class db.r5.large ^
  --db-instance-identifier sample-replica-instance ^
  --engine aurora-mysql
```

Lorsque l'instance de base de données est créée et disponible, la réplication commence. Vous pouvez déterminer si l'instance de base de données est disponible en appelant la commande AWS CLI describe-db-instances.

API RDS

Pour créer un cluster de bases de données Aurora MySQL en tant que réplica en lecture entre régions avec l'API

1. Appelez l'opération CreateDBCluster de l'API RDS dans la Région AWS où vous voulez créer le cluster de bases de données de réplica en lecture. Incluez le paramètre `ReplicationSourceIdentifier` et indiquez l'Amazon Resource Name (ARN) du cluster de bases de données source pour lequel vous créez un réplica en lecture.

   Pour une réplication entre régions où le cluster de bases de données identifié par `ReplicationSourceIdentifier` est chiffré, spécifiez le paramètre `KmsKeyId` et définissez le paramètre `StorageEncrypted` sur `true`.

   > **Note**
   >
   > Vous pouvez configurer la réplication entre régions à partir d'un cluster de bases de données non chiffré vers un réplica en lecture chiffré en définissant `StorageEncrypted` sur **`true`** et en fournissant une valeur pour l'option `KmsKeyId`. Dans ce cas, il n'est pas nécessaire de spécifier `PreSignedUrl`.

   Vous n'avez pas besoin d'inclure les paramètres `MasterUsername` et `MasterUserPassword`, parce que ces valeurs sont extraites du cluster de bases de données source.

L’exemple de code suivant crée un réplica en lecture dans la région us-east-1 à partir d’un instantané du cluster de bases de données non chiffré dans la région us-west-2. L’action est appelée dans la région us-east-1.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

L’exemple de code suivant crée un réplica en lecture dans la région us-east-1 à partir d’un instantané du cluster de bases de données chiffré dans la région us-west-2. L’action est appelée dans la région us-east-1.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBCluster
  &KmsKeyId=my-us-east-1-key
  &StorageEncrypted=true
  &PreSignedUrl=https%253A%252F%252Frds.us-west-2.amazonaws.com%252F
         %253FAction%253DCreateDBCluster
         %2526DestinationRegion%253Dus-east-1
         %2526KmsKeyId%253Dmy-us-east-1-key
         %2526ReplicationSourceIdentifier%253Darn%25253Aaws%25253Ards%25253Aus-
west-2%25253A123456789012%25253Acluster%25253Asample-master-cluster
         %2526SignatureMethod%253DHmacSHA256
         %2526SignatureVersion%253D4
         %2526Version%253D2014-10-31
         %2526X-Amz-Algorithm%253DAWS4-HMAC-SHA256
         %2526X-Amz-Credential%253DAKIADQKE4SARGYLE%252F20161117%252Fus-
west-2%252Frds%252Faws4_request
         %2526X-Amz-Date%253D20161117T215409Z
         %2526X-Amz-Expires%253D3600
```

```
        %2526X-Amz-SignedHeaders%253Dcontent-type%253Bhost%253Buser-agent%253Bx-
amz-content-sha256%253Bx-amz-date
        %2526X-Amz-Signature
%253D255a0f17b4e717d3b67fad163c3ec26573b882c03a65523522cf890a67fca613
  &ReplicationSourceIdentifier=arn:aws:rds:us-west-2:123456789012:cluster:sample-
master-cluster
  &DBClusterIdentifier=sample-replica-cluster
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T001547Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=a04c831a0b54b5e4cd236a90dcb9f5fab7185eb3b72b5ebe9a70a4e95790c8b7
```

Pour la réplication entre les régions AWS GovCloud (US-East) et AWS GovCloud (US-West), où le cluster de bases de données identifié par `ReplicationSourceIdentifier` est chiffré, spécifiez également le paramètre `PreSignedUrl`. L'URL présignée doit être une demande valide pour l'opération d'API `CreateDBCluster`, laquelle peut être effectuée dans la Région AWS source qui contient le cluster de bases de données chiffré à répliquer. L'identifiant de clé KMS qui permet de chiffrer le réplica en lecture, qui doit être une clé KMS valide pour la Région AWS de destination. Pour générer automatiquement plutôt que manuellement une URL présignée, utilisez plutôt la commande AWS CLI create-db-cluster avec l'option `--source-region`.

2. Vérifiez que le cluster de bases de données est disponible pour être utilisé avec l'opération DescribeDBClusters de l'API RDS, comme indiqué dans l'exemple suivant.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=DescribeDBClusters
  &DBClusterIdentifier=sample-replica-cluster
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T002223Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=84c2e4f8fba7c577ac5d820711e34c6e45ffcd35be8a6b7c50f329a74f35f426
```

Lorsque les résultats `DescribeDBClusters` affichent le statut `available`, créez l'instance principale pour le cluster de bases de données afin que la réplication commence. Pour ce faire, utilisez l'action CreateDBInstance de l'API RDS, comme illustré dans l'exemple suivant.

```
https://rds.us-east-1.amazonaws.com/
  ?Action=CreateDBInstance
  &DBClusterIdentifier=sample-replica-cluster
  &DBInstanceClass=db.r5.large
  &DBInstanceIdentifier=sample-replica-instance
  &Engine=aurora-mysql
  &SignatureMethod=HmacSHA256
  &SignatureVersion=4
  &Version=2014-10-31
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20161117/us-east-1/rds/aws4_request
  &X-Amz-Date=20160201T003808Z
  &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
  &X-Amz-Signature=125fe575959f5bbcebd53f2365f907179757a08b5d7a16a378dfa59387f58cdb
```

Lorsque l'instance de base de données est créée et disponible, la réplication commence. Vous pouvez déterminer si l'instance de base de données est disponible en appelant la commande AWS CLI DescribeDBInstances.

Affichage des réplicas entre régions Amazon Aurora MySQL

Vous pouvez afficher les relations de réplication entre régions de vos clusters de bases de données Amazon Aurora MySQL en appelant la commande AWS CLI describe-db-clusters ou l'opération DescribeDBClusters de l'API RDS. Dans la réponse, reportez-vous au champ `ReadReplicaIdentifiers` pour obtenir les identifiants de cluster de bases de données de tous les clusters de bases de données de réplica en lecture entre régions. Reportez-vous à l'élément `ReplicationSourceIdentifier` pour obtenir l'ARN du cluster de bases de données source qui est la source de réplication.

Promotion d'un réplica en lecture en cluster de bases de données pour Aurora MySQL

Vous pouvez promouvoir un réplica en lecture Aurora MySQL en cluster de bases de données autonome. Lorsque vous promouvez un réplica en lecture Aurora MySQL, les instances de base de données sont redémarrées avant de devenir disponibles.

En règle générale, vous effectuez la promotion d'un réplica en lecture Aurora MySQL en cluster de bases de données autonome comme plan de récupération des données en cas de défaillance du cluster de bases de données source.

Pour cela, commencez par créer un réplica en lecture, puis surveillez le cluster de bases de données source pour détecter les défaillances. En cas de panne, procédez comme suit :

1. Promouvez le réplica en lecture.
2. Dirigez le trafic de base de données vers le cluster de bases de données promu.
3. Créez un réplica en lecture de remplacement en utilisant le cluster de bases de données promu comme source.

Lorsque vous effectuez la promotion d'un réplica en lecture, celui-ci devient un cluster de bases de données Aurora autonome. Le processus de promotion peut prendre plusieurs minutes ou plus longtemps, selon la taille du réplica en lecture. Une fois le réplica en lecture promu en nouveau cluster de bases de données, il est similaire à tout autre cluster de bases de données. Par exemple, vous pouvez créer des réplicas en lecture à partir de celui-ci et effectuer des opérations de restauration à un moment donné. Vous pouvez également créer des réplicas Aurora pour le cluster de bases de données.

Étant donné que le cluster de bases de données promu n'est plus un réplica en lecture, vous ne pouvez pas l'utiliser comme cible de réplication.

Les étapes suivantes décrivent le processus général de promotion d'un réplica en lecture en cluster de bases de données :

1. Arrêtez l'écriture de toute transaction sur le cluster de bases de données source du réplica en lecture, puis attendez que toutes les mises à jour soient terminées sur le réplica en lecture. Les mises à jour de la base de données ont lieu sur les réplicas en lecture après avoir eu lieu sur le cluster de bases de données source, et cette latence de réplication peut varier de façon significative. Utilisez la métrique `ReplicaLag` pour déterminer à quel moment toutes les mises à jour ont été effectuées sur le réplica en lecture. La métrique `ReplicaLag` enregistre la durée pendant laquelle une instance de base de données de réplica en lecture retarde l'instance de base de données source. Lorsque la métrique `ReplicaLag` atteint `0`, le réplica en lecture a rattrapé l'instance de base de données source.
2. Promouvez le réplica en lecture à l'aide de l'option Promote (Promouvoir) sur la console Amazon RDS, de la commande AWS CLI promote-read-replica-db-cluster ou de l'opération PromoteReadReplicaDBCluster de l'API Amazon RDS.

Vous choisissez une instance de base de données Aurora MySQL pour promouvoir le réplica en lecture. Une fois le réplica en lecture promu, le cluster de bases de données Aurora MySQL est promu en cluster de bases de données autonome. L'instance de base de données ayant la priorité de basculement la plus élevée est promue en instance de base de données pour le cluster de bases de données. Les autres instances de base de données deviennent des réplicas Aurora.

> **Note**
>
> Le processus de promotion dure quelques minutes. Lorsque vous promouvez un réplica en lecture, la réplication est arrêtée et les instances de base de données sont redémarrées. Une fois le redémarrage terminé, le réplica en lecture est disponible en tant que nouveau cluster de bases de données.

Console

Pour promouvoir un réplica en lecture Aurora MySQL en cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la console, choisissez Instances.

   Le panneau Instance s'affiche.
3. Dans le panneau Instances, choisissez le réplica en lecture que vous souhaitez promouvoir.

   Les réplicas en lecture apparaissent comme instances de base de données Aurora MySQL.
4. Sous Actions, choisissez Promouvoir le réplica en lecture.
5. Dans la page de confirmation, sélectionnez Promote read replica (Promouvoir le réplica en lecture).

AWS CLI

Pour promouvoir un réplica en lecture en cluster de bases de données, utilisez la commande AWS CLI promote-read-replica-db-cluster.

Example

Pour Linux, macOS ou Unix :

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier mydbcluster
```

Pour Windows :

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier mydbcluster
```

API RDS

Pour promouvoir un réplica en lecture en cluster de bases de données, appelez PromoteReadReplicaDBCluster.

Dépannage des réplicas entre régions pour Amazon Aurora MySQL

La liste ci-dessous répertorie les messages d'erreur courants que vous pouvez rencontrer lors de la création d'un réplica en lecture entre régions Amazon Aurora et indique comment les résoudre.

Source cluster [DB cluster ARN] doesn't have binlogs enabled

Pour résoudre ce problème, activez la journalisation binaire sur le cluster de bases de données source. Pour plus d'informations, consultez Avant de commencer.

Source cluster [DB cluster ARN] doesn't have cluster parameter group in sync on writer

Vous recevez cette erreur si vous avez mis à jour le paramètre du cluster de bases de données `binlog_format` sans redémarrer l'instance principale du cluster de bases de données. Redémarrez l'instance principale (c'est-à-dire, l'auteur) du cluster de bases de données, puis réessayez.

Source cluster [DB cluster ARN] already has a read replica in this region

Vous pouvez avoir jusqu'à cinq clusters de bases de données inter-régions constituant des réplicas en lecture pour chaque cluster de bases de données source dans n'importe quelle Région AWS. Si vous disposez déjà du nombre maximal de réplicas en lecture pour un cluster de bases de données dans une Région AWS particulière, vous devez supprimer un cluster existant avant de pouvoir créer un cluster de bases de données inter-régions dans cette région.

DB cluster [DB cluster ARN] requires a database engine upgrade for cross-region replication support (Le cluster de bases de données [ARN du cluster de bases de données] nécessite une mise à jour du moteur de base de données pour la prise en charge de la réplication entre régions)

Pour résoudre ce problème, mettez à niveau la version du moteur de base de données pour toutes les instances du cluster de bases de données source vers la version de moteur de base de données la plus récente, puis réessayez de créer une base de données de réplica en lecture entre régions.

Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires)

Comme Amazon Aurora MySQL est compatible avec MySQL, vous pouvez configurer la réplication entre une base de données MySQL et un cluster de bases de données Amazon Aurora MySQL. Ce type de réplication utilise la réplication du journal binaire MySQL et est également appelé réplication de journaux binaires. Si vous utilisez la réplication de journaux binaires avec Aurora, nous vous recommandons que votre base de données MySQL exécute MySQL version 5.5 ou ultérieure. Vous pouvez configurer la réplication où votre cluster de bases de données Aurora MySQL est la source de réplication ou le réplica. Vous pouvez répliquer avec une instance de base de données Amazon RDS MySQL, une base de données MySQL externe à Amazon RDS ou un autre cluster de bases de données Aurora MySQL.

> **Note**
>
> Vous ne pouvez pas utiliser la réplication des journaux binaires vers ou depuis certains types de clusters de bases de données Aurora. En particulier, la réplication des journaux binaires n'est pas disponible pour les clusters Aurora Serverless v1. Si l'instruction `SHOW MASTER STATUS` et `SHOW SLAVE STATUS` (Aurora MySQL version 2) ou `SHOW REPLICA STATUS` (Aurora MySQL version 3) ne renvoie aucune sortie, vérifiez que le cluster que vous utilisez prend en charge la réplication des journaux binaires.

Vous pouvez aussi effectuer la réplication avec une instance de base de données RDS for MySQL ou un cluster de bases de données Aurora MySQL d'une autre Région AWS. Lorsque vous exécutez la réplication entre Régions AWS, assurez-vous que vos clusters et instances de bases de données sont publiquement accessibles. Si les clusters de bases de données MySQL Aurora se trouvent dans des sous-réseaux privés de votre VPC, utilisez l'appairage de VPC entre les Régions AWS. Pour plus d'informations, consultez Un cluster de bases de données d'un VPC accessible par une instance EC2 d'un autre VPC.

Si vous voulez configurer la réplication entre un cluster de bases de données Aurora MySQL et un cluster de bases de données Aurora MySQL dans une autre Région AWS, vous pouvez créer un cluster de bases de données Aurora MySQL en tant que réplica en lecture dans une Région AWS différente de celle du cluster de bases de données source. Pour plus d'informations, consultez Réplication de clusters de bases de données Amazon Aurora MySQL dans différentes Régions AWS.

Avec Aurora MySQL versions 2 et 3, vous pouvez procéder à la réplication entre Aurora MySQL et une source ou une cible externe qui utilise des identifiants de transaction globaux (GTID) pour la réplication. Vérifiez que les paramètres liés aux GTID dans le cluster de bases de données Aurora MySQL comportent des paramètres compatibles avec le statut GTID de la base de données externe. Pour savoir comment procéder, consultez Utilisation de la réplication basée sur des identifiants de transaction globaux (GTID). Dans Aurora MySQL 3.01 et versions ultérieures, vous pouvez choisir comment attribuer des GTID à des transactions répliquées à partir d'une source n'utilisant pas de GTID. Pour en savoir plus sur la procédure stockée qui contrôle ce paramètre, consultez mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL version 3).

> **Warning**
>
> Lorsque vous effectuez une réplication entre Aurora MySQL et MySQL, assurez-vous que vous n'utilisez que les tables InnoDB. Si vous souhaitez répliquer des tables MyISAM, vous pouvez les convertir en InnoDB avant de configurer la réplication avec la commande suivante.
>
> ```
> alter table <schema>.<table_name> engine=innodb, algorithm=copy;
> ```

Dans les sections suivantes, configurez la réplication, arrêtez la réplication, mettez à l'échelle les lectures pour votre base de données, optimisez la réplication des journaux binaires et configurez le journal binaire amélioré.

Rubriques

- Configuration de la réplication des journaux binaires pour Aurora MySQL
- Arrêt de la réplication des journaux binaires pour Aurora MySQL
- Mise à l'échelle des lectures pour votre base de données MySQL avec Amazon Aurora
- Optimisation de la réplication des journaux binaires pour Aurora MySQL
- Configuration du binlog amélioré pour Aurora MySQL

Configuration de la réplication des journaux binaires pour Aurora MySQL

La configuration de la réplication MySQL avec Aurora MySQL implique les étapes suivantes, qui sont présentées en détail :

Table des matières

1. Activer la journalisation binaire sur la source de réplication

Vous trouverez des instructions sur la façon d'activer la journalisation binaire sur la source de réplication pour votre moteur de base de données ci-après.

Moteur de base de données	Instructions
Aurora MySQL	Pour activer la journalisation binaire sur un cluster de bases de données Aurora MySQL Définissez le paramètre de cluster de bases de données `binlog_format` sur `ROW`, `STATEMENT` ou `MIXED`. La valeur `MIXED` est recommandée sauf si vous avez besoin d'un format de journal binaire spécifique. (La valeur par défaut est `OFF`.) Pour modifier le paramètre `binlog_format` , créez un groupe de paramètres de cluster de bases de données personnalisé et associez ce groupe de paramètres

Moteur de base de données	Instructions
	personnalisé à votre cluster de bases de données. Vous ne pouvez pas modifier les paramètres dans le groupe de paramètres de cluster de bases de données par défaut. Si vous modifiez le paramètre `binlog_format` en remplaçant `OFF` par une autre valeur, redémarrez votre cluster de bases de données Aurora pour que la modificat ion prenne effet. Pour plus d'informations, consultez Paramètres de cluster de bases de données et d'instance de base de données Amazon Aurora et Groupes de paramètres pour Amazon Aurora.
RDS for MySQL	Pour activer la journalisation binaire sur une instance de base de données Amazon RDS Vous ne pouvez pas activer la journalisation binaire directement pour une instance de base de données Amazon RDS, mais vous pouvez l'activer en exécutant l'une des actions suivantes : • Activez les sauvegardes automatiques de l'instance de base de données. Vous pouvez activer les sauvegardes automatiques lorsque vous créez une instance de base de données ou vous pouvez activer les sauvegardes en modifiant une instance de base de données existante. Pour plus d'informations, consultez Création d'une instance de base de données dans le Guide de l'utilisateur Amazon RDS. • Créez un réplica en lecture pour l'instance de base de données. Pour plus d'informations, consultez Utilisation des réplicas en lecture dans le Guide de l'utilisa teur Amazon RDS.

Moteur de base de données	Instructions
MySQL (externe)	Pour configurer la réplication chiffrée Pour répliquer des données en toute sécurité avec Aurora MySQL version 2, vous pouvez utiliser la réplication chiffrée. **Note** Si vous n'avez pas besoin d'utiliser la réplication chiffrée, vous pouvez ignorer ces étapes. Pour pouvoir utiliser la réplication chiffrée, vous devez impérativement disposer des éléments suivants : • Le protocole SSL doit être activé sur la base de données source MySQL externe. • Une clé client et un certificat client doivent être préparés pour le cluster de bases de données Aurora MySQL. Pendant la réplication chiffrée, le cluster de bases de données Aurora MySQL agit comme client du serveur de base de données MySQL. Les certificats et les clés privées du client Aurora MySQL sont au format .pem dans les fichiers. 1. Vérifiez bien que vous êtes prêt à procéder à la réplication chiffrée : • Si le protocole SSL n'est pas activé sur la base de données source MySQL externe et que vous ne disposez pas d'une clé client et d'un certificat client prêts, activez le protocole SSL sur le serveur de base de données MySQL et générez la clé client et le certificat client requis. • Si le protocole SSL est activé sur la base de données source externe, fournissez une clé et un certificat client pour le cluster de bases de données Aurora MySQL. En leur absence, générez une nouvelle clé et un nouveau certificat pour le cluster de bases de données Aurora MySQL. Pour signer le

Moteur de base de données	Instructions

certificat client, vous devez disposer de la clé d'autorité de certification utilisée pour configurer le protocole SSL sur la base de données source MySQL externe.

Pour plus d'informations, consultez Création de certificats et clés SSL à l'aide d'openssl dans la documentation MySQL.

Vous avez besoin du certificat de l'autorité de certification, de la clé client et du certificat client.

2. Connectez-vous au cluster de bases de données Aurora MySQL en tant qu'utilis ateur principal à l'aide du protocole SSL.

   Pour plus d'informations sur la connexion à un cluster de bases de données Aurora MySQL avec le protocole SSL, consultez Connexions TLS aux clusters de bases de données Aurora MySQL.

3. Exécutez la procédure stockée `mysql.rds_import_binlog_ssl _material` pour importer les informations SSL dans le cluster de bases de données Aurora MySQL.

   Pour le paramètre `ssl_material_value` , insérez les informations des fichiers au format .pem pour le cluster de bases de données Aurora MySQL dans les données utiles JSON correctes.

   L'exemple suivant importe des informations SSL dans un cluster de bases de données Aurora MySQL. Dans les fichiers au format .pem, le code du corps est généralement plus long que le code du corps affiché dans l'exemple.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcj
qP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96
xbiFveSFJuOp/d6RJhJOI0iBXr
```

Moteur de base de données	Instructions

```
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/
i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz22
1CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICA
TE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcj
qP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96
xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/
i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz22
1CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE
 KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pc
jqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSF
JuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJ
tjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMu
cxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

Pour plus d'informations, consultez mysql.rds_import_binlog_ssl_material et Connexions TLS aux clusters de bases de données Aurora MySQL.

Moteur de base de données	Instructions

> **Note**
>
> Après l'exécution de la procédure, les secrets sont stockés dans les fichiers. Pour supprimer les fichiers ultérieurement, vous pouvez exécuter la procédure stockée mysql.rds_remove_binlog_ssl_material.

Pour activer la journalisation binaire sur une base de données MySQL externe

1. Depuis un shell de commande, arrêtez le service mysql.

```
sudo service mysqld stop
```

2. Modifiez le fichier `my.cnf` (qui se trouve généralement sous `/etc`).

```
sudo vi /etc/my.cnf
```

Ajoutez les options `log_bin` et `server_id` à la section `[mysqld]`. L'option `log_bin` fournit un identifiant de nom de fichier pour les fichiers journaux binaires. L'option `server_id` fournit un identifiant unique pour le serveur dans les relations source/réplica.

Si la réplication chiffrée n'est pas obligatoire, assurez-vous que la base de données MySQL externe est démarrée avec les journaux binaires activés et le protocole SSL désactivé.

Les entrées correspondantes du fichier `/etc/my.cnf` pour les données non chiffrées sont les suivantes.

```
log-bin=mysql-bin
server-id=2133421
innodb_flush_log_at_trx_commit=1
sync_binlog=1
```

Moteur de base de données	Instructions

Si la réplication chiffrée est obligatoire, assurez-vous que la base de données MySQL externe est démarrée avec le protocole SSL et les journaux binaires activés.

Les entrées du fichier `/etc/my.cnf` incluent les emplacements de fichier .pem pour le serveur de base de données MySQL.

```
log-bin=mysql-bin
server-id=2133421
innodb_flush_log_at_trx_commit=1
sync_binlog=1

# Setup SSL.
ssl-ca=/home/sslcerts/ca.pem
ssl-cert=/home/sslcerts/server-cert.pem
ssl-key=/home/sslcerts/server-key.pem
```

En outre, l'option `sql_mode` de votre instance de base de données MySQL doit être définie avec la valeur 0 ou ne pas être incluse dans votre fichier my.cnf.

Une fois connecté à la base de données MySQL externe, enregistrez la position du journal binaire de la base de données MySQL externe.

```
mysql> SHOW MASTER STATUS;
```

Votre sortie doit ressembler à ce qui suit :

```
+------------------+----------+--------------+------------------
+-------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
 Executed_Gtid_Set |
+------------------+----------+--------------+------------------
+-------------------+
```

Moteur de base de données	Instructions

```
| mysql-bin.000031 |      107 |              |                  |
              |
+------------------+----------+--------------+------------------
+-------------------+
1 row in set (0.00 sec)
```

Pour plus d'informations, consultez Setting the replication source configuration dans la documentation MySQL.

3. Démarrez le service mysql.

```
sudo service mysqld start
```

2. Conserver les journaux binaires sur la source de réplication jusqu'à ce qu'ils ne soient plus nécessaires

Lorsque vous utilisez la réplication des journaux binaires MySQL, Amazon RDS ne gère pas le processus de réplication. En conséquence, vous devez vous assurer que les fichiers des journaux binaires sur votre source de réplication sont conservés jusqu'après que les modifications ont été appliquées au réplica. Cette maintenance vous permet de restaurer votre base de données source en cas de panne.

Suivez les instructions suivantes pour conserver les journaux binaires de votre moteur de base de données.

Moteur de base de données	Instructions
Aurora MySQL	Pour conserver les journaux binaires sur un cluster de bases de données Aurora MySQL

Moteur de base de données	Instructions
	Vous n'avez pas accès aux fichiers journaux binaires pour un cluster de bases de données Aurora MySQL. En conséquence, vous devez choisir une période assez longue de conservation des fichiers journaux binaires sur votre source de réplication pour être assuré que les modifications ont été appliquées à votre réplica avant que le fichier journal binaire ne soit supprimé par Amazon RDS. Vous pouvez conserver les fichiers journaux binaires sur un cluster de bases de données Aurora MySQL pendant 90 jours maximum. Si vous configurez la réplication avec une base de données MySQL ou une instance de base de données RDS for MySQL comme réplica, et que la base de données pour laquelle vous créez un réplica est très volumineuse, choisissez une durée conséquente pour conserver les fichiers journaux binaires jusqu'à ce que la copie initiale de la base de données sur le réplica soit complète et que le retard du réplica ait atteint la valeur 0. Pour définir la période de rétention des journaux binaires, utilisez la procédure mysql.rds_set_configuration et spécifiez un paramètre de configuration `'binlog retention hours'`, ainsi que le nombre d'heures pendant lequel conserver les fichiers journaux binaires sur le cluster de bases de données. La valeur maximum pour Aurora MySQL versions 2.11.0 et ultérieures et version 3 est 2 160 (90 jours). L'exemple suivant définit la période de rétention des fichiers journaux binaires sur 6 jours : `CALL mysql.rds_set_configuration('binlog retention hours', 144);` Une fois la réplication démarrée, vous pouvez vérifier que les modifications ont été appliquées à votre réplica en exécutant la commande `SHOW SLAVE STATUS` (Aurora MySQL version 2) ou `SHOW REPLICA STATUS` (Aurora MySQL version 3) sur votre réplica et en vérifiant le champ `Seconds behind master`. Si la valeur du champ `Seconds behind master` est 0, il n'y a pas de décalage de réplica. Quand il n'y a pas de retard du réplica, réduisez la période pendant laquelle les

Moteur de base de données	Instructions
	fichiers journaux binaires sont conservés en définissant le paramètre de configuration `binlog retention hours` sur une durée plus petite. Si ce paramètre n'est pas spécifié, la valeur par défaut pour Aurora MySQL est 24 (1 jour). Si vous spécifiez une valeur supérieure à la valeur maximale pour `'binlog retention hours'`, Aurora MySQL utilise la valeur maximale.
RDS for MySQL	Pour conserver les journaux binaires sur une instance de base de données Amazon RDS Vous pouvez conserver les fichiers journaux binaires sur une instance de base de données Amazon RDS en définissant les heures de conservation des journaux binaires comme vous le feriez pour un cluster de bases de données Aurora MySQL (procédure décrite à la ligne précédente). Vous pouvez également conserver les fichiers journaux binaires sur une instance de base de données Amazon RDS en créant un réplica en lecture pour l'instance de base de données. Ce réplica en lecture a pour seul but temporaire et exclusif de conserver les fichiers journaux binaires. Une fois le réplica en lecture créé, appelez la procédure mysql.rds_stop_replication sur le réplica en lecture. Lorsque la réplicati on est arrêtée, Amazon RDS ne supprime aucun des fichiers journaux binaires sur la source de réplication. Après avoir configuré la réplication avec votre réplica permanent, vous pouvez supprimer le réplica en lecture lorsque le retard du réplica (champ `Seconds behind master`) entre votre source de réplication et votre réplica permanent atteint 0.

Moteur de base de données	Instructions
MySQL (externe)	Pour conserver les journaux binaires sur une base de données MySQL externe Comme les fichiers journaux binaires sur une base de données MySQL externe ne sont pas gérés par Amazon RDS, ils sont conservés jusqu'à ce que vous les supprimiez. Une fois la réplication démarrée, vous pouvez vérifier que les modifications ont été appliquées à votre réplica en exécutant la commande `SHOW SLAVE STATUS` (Aurora MySQL version 2) ou `SHOW REPLICA STATUS` (Aurora MySQL version 3) sur votre réplica et en vérifiant le champ `Seconds behind master`. Si la valeur du champ `Seconds behind master` est 0, il n'y a pas de décalage de réplica. Quand il n'y a pas de retard du réplica, vous pouvez supprimer les anciens fichiers journaux binaires.

3. Créer une copie ou un vidage de votre source de réplication

Vous utilisez un instantané, un clone ou un vidage de votre source de réplication pour charger une copie de référence de vos données sur votre réplica. Ensuite, vous commencez la réplication à partir de ce point.

Utilisez les instructions suivantes pour créer une copie ou un vidage de votre source de réplication pour votre moteur de base de données.

Moteur de base de données	Instructions
Aurora MySQL	Pour créer une copie d'un cluster de bases de données Aurora MySQL Utilisez l'une des méthodes suivantes : • Restaurez un instantané de cluster de bases de données :

Moteur de base de données	Instructions
	1. Créez un instantané de cluster de bases de données de votre cluster de bases de données Amazon Aurora. Pour plus d'informations, consultez Création d'un instantané de cluster de bases de données. 2. Créez un cluster de bases de données Aurora en procédant à une restauration à partir de l'instantané de cluster de bases de données que vous venez de créer. Veillez bien à conserver le même groupe de paramètres de base de données pour votre cluster de bases de données restauré que pour votre cluster de bases de données original. Ceci vous garantit que la journalisation binaire est activée pour la copie de votre cluster de bases de données. Pour plus d'informations, consultez Restauration à partir d'un instantané de cluster de bases de données. • Clonez votre cluster de bases de données. Pour plus d'informations, consultez Clonage d'un volume pour un cluster de bases de données Amazon Aurora. Pour déterminer le nom et la position du fichier binlog Utilisez l'une des méthodes suivantes : • Dans AWS Management Console : 1. Choisissez Bases de données, puis choisissez l'instance principale (enregist reur) de votre nouveau cluster de bases de données Aurora afin d'en afficher les détails. 2. Faites défiler l'écran jusqu'à Événements récents. Un message d'événeme nt s'affiche pour indiquer le nom et la position du fichier journal binaire. Ce message d'événement est au format suivant. `Binlog position from crash recovery is` *`binlog-file-name  binlog-po sition`* 3. Enregistrez le nom et la position du fichier journal binaire lorsque vous commencez la réplication.

Moteur de base de données	Instructions
	• Appelez la commande describe-events de l'AWS CLI, comme dans l'exemple suivant. `aws rds describe-events` `{` `  "Events": [` `  {` `  "EventCategories": [],` `  "SourceType": "db-instance",` `  "SourceArn": "arn:aws:rds:us-west-2:123456789012:db:sample-restored-instance",` `  "Date": "2016-10-28T19:43:46.862Z",` `  "Message": "Binlog position from crash recovery is mysql-bin-changelog.000003 4278",` `  "SourceIdentifier": "sample-restored-instance"` `  }` `  ]` `}` • Vérifiez dans le journal des erreurs MySQL la dernière position du fichier binlog MySQL.

Pour créer un vidage d'un cluster de bases de données Aurora MySQL

Si votre cible de réplica est une base de données MySQL externe ou une instance de base de données RDS for MySQL, vous devez créer un fichier de vidage à partir de votre cluster de bases de données Aurora.

Veillez bien à exécuter la commande `mysqldump` sur la copie du cluster de bases de données Aurora que vous avez créée. Cela permet d'éviter le verrouillage des tables sources lors du vidage. Si le vidage était effectué directement sur le cluster de bases de données source, il serait nécessaire de verrouiller ces tables pour empêcher les écritures simultanées pendant le vidage.

Moteur de base de données	Instructions
	1. Connectez-vous à votre cluster de bases de données à l'aide d'un client MySQL. 2. Émettez la commande `mysqldump` . Exemples : `PROMPT> mysqldump --databases database_name --single-transaction --order-by-primary -r backup.sql -u local_user s -p` 3. Une fois le fichier de vidage créé, vous pouvez supprimer la copie du cluster de bases de données.
RDS for MySQL	Pour créer un instantané d'une instance de base de données Amazon RDS Créez un réplica en lecture de votre instance de base de données Amazon RDS. Pour plus d'informations, consultez Création d'un réplica en lecture dans le Guide de l'utilisateur Amazon Relational Database Service. 1. Connectez-vous à votre réplica en lecture et arrêtez la réplication en exécutant la procédure mysql.rds_stop_replication. 2. Une fois le réplica en lecture arrêté, connectez-vous à ce réplica et exécutez la commande `SHOW SLAVE STATUS` (Aurora MySQL version 2) ou `SHOW REPLICA STATUS` (Aurora MySQL version 3). Extrayez le nom du fichier journal binaire actif du champ `Relay_Master_Log_File` et la position du fichier journal du champ `Exec_Master_Log_Pos` . Enregistrez ces valeurs lorsque vous démarrez la réplication. 3. Pendant que le réplica en lecture reste à l'état Arrêté, créez un instantané de base de données du réplica en lecture. Pour plus d'informations, consultez Création d'un instantané de base de données dans le Guide de l'utilisateur Amazon Relational Database Service. 4. Supprimez le réplica en lecture.

Moteur de base de données	Instructions
MySQL (externe)	Pour créer un vidage d'une base de données MySQL externe 1. Avant de créer un vidage, vous devez vous assurer que l'emplacement du journal binaire du vidage est à jour avec les données de votre instance source. Pour ce faire, vous devez d'abord arrêter toutes les opérations d'écriture sur l'instance avec la commande suivante : `mysql> FLUSH TABLES WITH READ LOCK;` 2. Créez un vidage de votre base de données MySQL à l'aide de la commande `mysqldump` comme illustré ci-après : `PROMPT> sudo mysqldump --databases database_name --master-data=2 --single-transaction \ --order-by-primary -r backup.sql -u local_user -p` 3. Après avoir créé le vidage, déverrouillez les tables de votre base de données MySQL avec la commande suivante : `mysql> UNLOCK TABLES;`

4. Charger le vidage dans votre cible de réplica (si nécessaire)

Si vous prévoyez de charger les données à partir d'un vidage d'une base de données MySQL externe à Amazon RDS, vous souhaiterez peut-être créer une instance EC2 sur laquelle copier les fichiers de vidage. Vous pourrez ensuite charger les données dans votre cluster de bases de données ou votre instance de base de données à partir de cette instance EC2. À l'aide de cette approche, vous pouvez compresser les fichiers de vidage avant de les copier sur l'instance EC2 afin de réduire les coûts réseau associés à la copie des données sur Amazon RDS. Vous pouvez aussi chiffrer les fichiers de vidage pour sécuriser les données tandis qu'elles sont transférées sur le réseau.

> **Note**
>
> Si vous créez un cluster de bases de données Aurora MySQL comme cible de réplica, vous n'avez pas besoin de charger un fichier de vidage :
>
> - Vous pourrez ultérieurement restaurer un cluster de bases de données pour créer le cluster. Pour plus d'informations, consultez Restauration à partir d'un instantané de cluster de bases de données.
> - Vous pouvez cloner votre cluster de bases de données source pour créer un cluster de bases de données. Pour plus d'informations, consultez Clonage d'un volume pour un cluster de bases de données Amazon Aurora.
> - Vous pouvez migrer les données d'un instantané d'instance de base de données vers un nouveau cluster de bases de données. Pour plus d'informations, consultez Migration de données vers un cluster de bases de données Amazon Aurora MySQL.

Utilisez les instructions suivantes pour charger le vidage de votre source de réplication dans votre cible de réplica pour votre moteur de base de données.

Moteur de base de données	Instructions
Aurora MySQL	Pour charger un vidage dans un cluster de bases de données Aurora MySQL 1. Copiez la sortie de la commande `mysqldump` de votre source de réplication vers un emplacement qui peut aussi se connecter à votre cluster de bases de données Aurora MySQL. 2. Connectez-vous à votre cluster de bases de données Aurora MySQL à l'aide de la commande `mysql`. Voici un exemple de. `PROMPT> mysql -h host_name -port=3306 -u db_master_user -p` 3. À l'invite de commande `mysql`, exécutez la commande `source` et transmettez-lui le nom du fichier de vidage de votre base de données pour charger les données dans le cluster de bases de données Aurora MySQL, par exemple :

Moteur de base de données	Instructions
	`` mysql> source backup.sql; ```
RDS for MySQL	Pour charger un vidage dans une instance de base de données Amazon RDS 1. Copiez la sortie de la commande `mysqldump` depuis votre source de réplication vers un emplacement qui peut aussi se connecter à votre instance de base de données MySQL. 2. Connectez-vous à votre instance de base de données MySQL à l'aide de la commande `mysql`. Voici un exemple de. ``` PROMPT> mysql -h host_name -port=3306 -u db_master_user -p ``` 3. A l'invite de commande `mysql`, exécutez la commande `source` et transmettez-lui le nom du fichier de vidage de votre base de données pour charger les données dans l'instance de base de données MySQL, par exemple : ``` mysql> source backup.sql; ```

Moteur de base de données	Instructions
MySQL (externe)	Pour charger un vidage dans une base de données MySQL externe Vous ne pouvez pas charger un instantané de base de données ou un instantané de cluster de bases de données dans une base de données MySQL externe. A la place, vous devez utiliser la sortie de la commande `mysqldump` . 1. Copiez la sortie de la commande `mysqldump` depuis votre source de réplicati on vers un emplacement qui peut aussi se connecter à votre base de données MySQL. 2. Connectez-vous à votre base de données MySQL à l'aide de la commande `mysql`. Voici un exemple de. `PROMPT> mysql -h host_name -port=3306 -u db_master_user -p` 3. À l'invite de commande `mysql`, exécutez la commande `source` et transmettez-lui le nom du fichier de vidage de votre base de données pour charger les données dans votre base de données MySQL. Voici un exemple de. `mysql> source backup.sql;`

5. Créer un utilisateur de réplication sur votre source de réplication

Créez un ID utilisateur sur la source qui est utilisé uniquement pour la réplication. L'exemple suivant concerne RDS for MySQL ou les bases de données sources MySQL externes.

```
mysql> CREATE USER 'repl_user'@'domain_name' IDENTIFIED BY 'password';
```

Pour les bases de données source Aurora MySQL, le paramètre de cluster de bases de données `skip_name_resolve` est défini sur 1 (`ON`) et ne peut pas être modifié. Vous devez donc utiliser une adresse IP pour l'hôte au lieu d'un nom de domaine. Pour plus d'informations, consultez skip_name_resolve dans la documentation MySQL.

```
mysql> CREATE USER 'repl_user'@'IP_address' IDENTIFIED BY 'password';
```

L'utilisateur nécessite les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE`. Accordez ces privilèges à l'utilisateur.

Si vous avez besoin d'utiliser la réplication chiffrée, demandez des connexions SSL à l'utilisateur de la réplication. Par exemple, vous pouvez utiliser l'une des instructions suivantes pour demander les connexions SSL sur le compte d'utilisateur `repl_user`.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'IP_address';
```

```
GRANT USAGE ON *.* TO 'repl_user'@'IP_address' REQUIRE SSL;
```

> **Note**
>
> Si `REQUIRE SSL` n'est pas inclus, la connexion de réplication peut revenir de façon silencieuse à une connexion non chiffrée.

6. Activer la réplication sur votre cible de réplica

Avant d'activer la réplication, nous vous recommandons de prendre un instantané manuel du cluster de bases de données Aurora MySQL ou de la cible de réplica de l'instance de base de données RDS for MySQL. Si un problème survient et que vous avez besoin de rétablir la réplication avec la cible de réplica du cluster de bases de données ou de l'instance de base de données, vous pouvez restaurer le cluster de bases de données ou l'instance de base de données à partir de cet instantané au lieu de devoir importer à nouveau les données dans votre cible de réplica.

Suivez les instructions suivantes pour activer la réplication pour votre moteur de base de données.

Moteur de base de données	Instructions
Aurora MySQL	Pour activer la réplication à partir d'un cluster de bases de données Aurora MySQL

Moteur de base de données	Instructions
	1. Trouvez le point de départ de la réplication. Vous avez besoin du nom du fichier journal binaire et de la position du journal binaire. Si la cible de réplication de votre cluster de bases de données a été créée à partir de ce qui suit : • Instantané du cluster de bases de données ou clone : récupérez le nom et la position du fichier binlog à partir des événements récents de votre nouveau cluster de bases de données, comme indiqué dans 3. Créer une copie ou un vidage de votre source de réplication. • Instantané de la base de données : vous avez extrait le nom et la position du fichier de journal binaire à partir de la commande `SHOW SLAVE STATUS` (Aurora MySQL version 2) ou `SHOW REPLICA STATUS` (Aurora MySQL version 3) lors de la création de l'instantané de votre source de réplication. 2. Connectez-vous au cluster de bases de données et exécutez les procédures suivantes pour démarrer la réplication avec votre source de réplication à l'aide du nom du fichier journal binaire et de l'emplacement de l'étape précédente : • mysql.rds_set_external_source (Aurora MySQL version 3) • mysql.rds_set_external_master (Aurora MySQL version 2) • mysql.rds_start_replication (toutes les versions) L'exemple suivant concerne Aurora MySQL version 3. `CALL mysql.rds_set_external_source ('mydbinstance.123456789012 .us-east-1.rds.amazonaws.com', 3306, 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0); CALL mysql.rds_start_replication;` Pour utiliser le chiffrement SSL, définissez la valeur finale sur 1 au lieu de `0`.

Moteur de base de données	Instructions
RDS for MySQL	Pour activer la réplication à partir d'une instance de base de données Amazon RDS 1. Si votre cible de réplica d'instance de base de données a été créé à partir d'un instantané de base de données, vous avez besoin du fichier journal binaire et de la position du journal binaire qui sont le point de départ de la réplication. Vous avez extrait ces valeurs à partir de la commande `SHOW SLAVE STATUS` (Aurora MySQL version 2) ou `SHOW REPLICA STATUS` (Aurora MySQL version 3) lors de la création de l'instantané de votre source de réplication. 2. Connectez-vous à l'instance de base de données et appelez les procédures mysql.rds_set_external_master (Aurora MySQL version 2) ou mysql.rds_set_external_source (Aurora MySQL version 3) et mysql.rds_start_replication pour démarrer la réplication avec votre source de réplication. Utilisez le nom du fichier journal binaire et l'emplacement à partir de l'étape précédente. Voici un exemple. `CALL mysql.rds_set_external_master ('mydbcluster.cluster-12345 6789012.us-east-1.rds.amazonaws.com', 3306, 'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0); CALL mysql.rds_start_replication;` Pour utiliser le chiffrement SSL, définissez la valeur finale sur 1 au lieu de `0`.

Moteur de base de données	Instructions
MySQL (externe)	Pour activer la réplication à partir d'une base de données MySQL externe 1. Extrayez le fichier journal binaire et la position du journal binaire qui sont le point de départ de la réplication. Vous avez extrait ces valeurs à partir de la commande `SHOW SLAVE STATUS` (Aurora MySQL version 2) ou `SHOW REPLICA STATUS` (Aurora MySQL version 3) lors de la création de l'instantané de votre source de réplication. Si votre cible de réplica MySQL externe a été renseignée à partir de la sortie de la commande `mysqldump` avec l'option `--master-data=2` , le fichier journal binaire et la position du journal binaire sont inclus dans la sortie. Voici un exemple. `-- -- Position to start replication or point-in-time recovery from -- -- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin-changelog.000031', MASTER_LOG_POS=107;` 2. Connectez-vous à la cible de réplica MySQL externe et exécutez `CHANGE MASTER TO` et `START SLAVE` (Aurora MySQL version 2) ou `START REPLICA` (Aurora MySQL version 3) pour démarrer la réplication avec votre source de réplication à l'aide du nom du fichier journal binaire et de l'emplacement de l'étape précédente, par exemple : `CHANGE MASTER TO MASTER_HOST = 'mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com', MASTER_PORT = 3306, MASTER_USER = 'repl_user', MASTER_PASSWORD = 'password', MASTER_LOG_FILE = 'mysql-bin-changelog.000031', MASTER_LOG_POS = 107; -- And one of these statements depending on your engine version: START SLAVE; -- Aurora MySQL version 2`

Moteur de base de données	Instructions
	`START REPLICA; -- Aurora MySQL version 3`

Si la réplication échoue, cela peut entraîner une augmentation importante des I/O involontaires sur le réplica, ce qui peut dégrader les performances. Si la réplication échoue ou n'est plus nécessaire, vous pouvez exécuter la procédure stockée mysql.rds_reset_external_master (Aurora MySQL version 2) ou mysql.rds_reset_external_source (Aurora MySQL version 3) pour supprimer la configuration de réplication.

Définition d'une position où arrêter la réplication vers un réplica en lecture

Dans Aurora MySQL versions 3.04 et ultérieures, vous pouvez démarrer la réplication, puis l'arrêter à la position spécifiée dans le fichier journal binaire en utilisant la procédure stockée mysql.rds_start_replication_until (Aurora MySQL version 3).

Pour démarrer la réplication vers un réplica en lecture et l'arrêter à une position donnée

1. À l'aide d'un client MySQL, connectez-vous au cluster de bases de données Aurora MySQL du réplica en tant qu'utilisateur principal.
2. Exécutez la procédure stockée mysql.rds_start_replication_until (Aurora MySQL version 3).

   L'exemple suivant lance la réplication et réplique les modifications jusqu'à ce qu'il atteigne la position `120` dans le fichier journal binaire `mysql-bin-changelog.000777`. Dans un scénario de reprise après sinistre, nous supposons que cette position `120` est juste avant le sinistre.

   ```
   call mysql.rds_start_replication_until(
     'mysql-bin-changelog.000777',
     120);
   ```

La réplication s'arrête automatiquement lorsque le point d'arrêt est atteint. L'événement RDS suivant est généré: `Replication has been stopped since the replica reached the stop point specified by the rds_start_replication_until stored procedure`.

Si vous utilisez une réplication basée sur des identifiants de transaction globaux (GTID), utilisez la procédure stockée mysql.rds_start_replication_until_gtid (Aurora MySQL version 3) au lieu de la procédure stockée mysql.rds_start_replication_until (Aurora MySQL version 3). Pour en savoir plus sur les réplications basées sur des identifiants de transaction globaux (GTID), consultez Utilisation de la réplication basée sur des identifiants de transaction globaux (GTID).

7. Surveiller votre réplica

Lorsque vous configurez la réplication MySQL avec un cluster de bases de données Aurora MySQL, vous devez surveiller les événements de basculement du cluster de bases de données Aurora MySQL quand il s'agit de la cible de réplica. En cas de basculement, le cluster de bases de données qui est votre cible de réplica peut alors être recréé sur un nouvel hôte avec une adresse réseau différente. Pour plus d'informations sur la surveillance des événements de basculement, consultez Utiliser la notification d'événements d'Amazon RDS.

Vous pouvez aussi surveiller à quelle distance la cible de réplica se trouve de la source de réplication en vous connectant à la cible de réplica et en exécutant la commande `SHOW SLAVE STATUS` (Aurora MySQL version 2) ou `SHOW REPLICA STATUS` (Aurora MySQL version 3). Dans la sortie de la commande, le champ `Seconds Behind Master` vous indique à quelle distance la cible de réplica se trouve de la source de réplication.

> **Important**
>
> Si vous mettez à niveau votre cluster de bases de données et que vous spécifiez un groupe de paramètres personnalisé, assurez-vous de redémarrer manuellement le cluster une fois la mise à niveau terminée. Cela obligera le cluster à utiliser vos nouveaux paramètres personnalisés et redémarrera la réplication des journaux binaires.

Synchronisation des mots de passe entre la source de réplication et la cible

Lorsque vous modifiez des comptes d'utilisateur et des mots de passe sur la source de réplication à l'aide d'instructions SQL, ces modifications sont automatiquement répliquées sur la cible de réplication.

Si vous utilisez la AWS Management Console, la AWS CLI, ou l'API RDS pour modifier le mot de passe principal sur la source de réplication, ces modifications ne sont pas automatiquement répliquées sur la cible de réplication. Si vous souhaitez synchroniser l'utilisateur principal et le mot

de passe principal entre les systèmes source et cible, vous devez effectuer vous-même la même modification sur la cible de réplication.

Arrêt de la réplication des journaux binaires pour Aurora MySQL

Pour arrêter la réplication des journaux binaires avec une instance de base de données MySQL, une base de données MySQL externe ou un autre cluster Aurora DB, suivez les étapes présentées en détail dans la suite de cette rubrique.

1. Arrêter la réplication des journaux binaires sur la cible de réplica

2. Désactiver la journalisation binaire sur la source de réplication

1. Arrêter la réplication des journaux binaires sur la cible de réplica

Utilisez les instructions suivantes pour arrêter la réplication des journaux binaires pour votre moteur de base de données.

Moteur de base de données	Instructions
Aurora MySQL	Pour arrêter la réplication des journaux binaires sur une cible de réplica de cluster de bases de données Aurora MySQL Connectez-vous au cluster de bases de données Aurora qui est la cible de réplica et appelez la procédure mysql.rds_stop_replication.
RDS for MySQL	Pour arrêter la réplication des journaux binaires sur une instance de base de données Amazon RDS Connectez-vous à l'instance de base de données RDS qui est la cible de réplica et appelez la procédure mysql.rds_stop_replication.
MySQL (externe)	Pour arrêter la réplication des journaux binaires sur une base de données MySQL externe Connectez-vous à la base de données MySQL et exécutez la commande `STOP SLAVE` (version 5.7) ou `STOP REPLICA` (version 8.0).

2. Désactiver la journalisation binaire sur la source de réplication

Utilisez les instructions indiquées dans le tableau suivant pour désactiver la journalisation binaire sur la source de réplication pour votre moteur de base de données.

Moteur de base de données	Instructions
Aurora MySQL	Pour désactiver la journalisation binaire sur un cluster de bases de données Amazon Aurora 1. Connectez-vous au cluster de bases de données Aurora qui est la source de réplication. 2. Utilisez la procédure mysql.rds_set_configuration et spécifiez le paramètre de configuration `binlog retention hours`, avec la valeur `NULL`, comme indiqué dans l'exemple suivant. `CALL mysql.rds_set_configuration('binlog retention hours', NULL);` **Note** Vous ne pouvez pas utiliser la valeur `0` pour `binlog retention hours`. 3. Définissez le paramètre `binlog_format` sur `OFF` sur la source de réplication. Le paramètre `binlog_format` se trouve dans le groupe de paramètres personnal isé du cluster de bases de données associé à votre cluster de bases de données. Après que vous avez modifié la valeur du paramètre `binlog_format` , redémarrez votre cluster de bases de données pour que la modification prenne effet. Pour plus d'informations, consultez Paramètres de cluster de bases de données et d'instance de base de données Amazon Aurora et Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora.

Moteur de base de données	Instructions
RDS for MySQL	Pour désactiver la journalisation binaire sur une instance de base de données Amazon RDS Vous ne pouvez pas désactiver la journalisation binaire directement pour une instance de base de données Amazon RDS, mais vous pouvez la désactiver en procédant comme suit : 1. Désactivez les sauvegardes automatiques de l'instance de base de données. Vous pouvez désactiver les sauvegardes automatiques en modifiant une instance de base de données existante et en affectant la valeur 0 au paramètre Période de rétention des sauvegardes. Pour plus d'informations, consultez Modification d'une instance de base de données Amazon RDS et Utilisation des sauvegardesdans le Guide de l'utilisateur Amazon Relational Database Service. 2. Supprimez tous les réplicas en lecture de l'instance de base de données. Pour plus d'informations, consultez Utilisation des réplicas en lecture des instances de base de données MariaDB, MySQL et PostgreSQL dans le Guide de l'utilisateur Amazon Relational Database Service.

Moteur de base de données	Instructions
MySQL (externe)	Pour désactiver la journalisation binaire sur une base de données MySQL externe Connectez-vous à la base de données MySQL et appelez la commande `STOP REPLICATION` . 1. Depuis un shell de commande, arrêtez le service `mysqld`. ``` sudo service mysqld stop ``` 2. Modifiez le fichier `my.cnf` (qui se trouve généralement sous `/etc`). ``` sudo vi /etc/my.cnf ``` Supprimez les options `log_bin` et `server_id` de la section `[mysqld]`. Pour plus d'informations, consultez Setting the replication source configuration dans la documentation MySQL. 3. Démarrez le service mysql. ``` sudo service mysqld start ```

Mise à l'échelle des lectures pour votre base de données MySQL avec Amazon Aurora

Vous pouvez utiliser Amazon Aurora avec votre instance de base de données MySQL pour tirer parti des capacités de mise à l'échelle en lecture d'Amazon Aurora et développer la charge de travail en lecture de votre instance de base de données MySQL. Pour utiliser Aurora afin de mettre à l'échelle les lectures pour votre instance de base de données MySQL, créez un cluster de bases de données Amazon Aurora MySQL et faites-en un réplica en lecture de votre instance de base de données MySQL. Cela s'applique à une instance de base de données RDS for MySQL ou à une base de données MySQL s'exécutant en dehors de Amazon RDS.

Pour plus d'informations sur la création d'un cluster de bases de données Amazon Aurora, consultez Création d'un cluster de bases de données Amazon Aurora.

Lorsque vous configurez la réplication entre votre instance de base de données MySQL et votre cluster de bases de données Amazon Aurora, veillez à respecter les instructions suivantes :

- Utilisez l'adresse du point de terminaison du cluster de bases de données Amazon Aurora lorsque vous référencez votre cluster de bases de données Amazon Aurora MySQL. Si un basculement se produit, le réplica Aurora qui est promu en instance principale du cluster de bases de données Aurora MySQL continue d'utiliser l'adresse du point de terminaison du cluster de bases de données.
- Tenez à jour les journaux binaires sur votre instance d'enregistreur jusqu'à ce que vous ayez vérifié qu'ils ont été appliqués au réplica Aurora. Cela garantit que vous pouvez restaurer votre instance d'enregistreur en cas de défaillance.

> **Important**
>
> Lorsque vous utilisez une réplication auto-gérée, vous êtes chargé de surveiller et résoudre les problèmes de réplication éventuels. Pour plus d'informations, consultez Diagnostic et résolution du retard entre réplicas en lecture.

> **Note**
>
> Les autorisations requises pour lancer la réplication sur un cluster de bases de données Aurora MySQL sont restreintes et ne sont pas disponibles pour votre utilisateur principal Amazon RDS. Par conséquent, vous devez utiliser les procédures mysql.rds_set_external_master (Aurora MySQL version 2) ou mysql.rds_set_external_source (Aurora MySQL version 3) et mysql.rds_start_replication pour configurer la réplication entre votre cluster de bases de données Aurora MySQL et votre instance de base de données MySQL.

Démarrer la réplication entre une instance source externe et un cluster de bases de données Aurora MySQL

1. Passez l'instance de base de données MySQL source en lecture seule :

```
mysql> FLUSH TABLES WITH READ LOCK;
mysql> SET GLOBAL read_only = ON;
```

2. Exécutez la commande `SHOW MASTER STATUS` sur l'instance de base de données MySQL source pour déterminer l'emplacement du journal binaire. Vous obtenez une sortie similaire à ce qui suit :

```
File                          Position
-----------------------------------
 mysql-bin-changelog.000031      107
-----------------------------------
```

3. Copiez la base de données de l'instance de base de données MySQL externe vers le cluster de bases de données Amazon Aurora MySQL à l'aide de `mysqldump`. Pour les bases de données très volumineuses, vous pouvez utiliser la procédure décrite dans la section Importation de données vers une base de données Amazon RDS for MySQL avec une durée d'indisponibilité réduite du Guide de l'utilisateur Amazon Relational Database Service.

   Pour Linux, macOS ou Unix :

```
mysqldump \
    --databases <database_name> \
    --single-transaction \
    --compress \
    --order-by-primary \
    -u local_user \
    -p local_password | mysql \
        --host aurora_cluster_endpoint_address \
        --port 3306 \
        -u RDS_user_name \
        -p RDS_password
```

   Pour Windows :

```
mysqldump ^
    --databases <database_name> ^
    --single-transaction ^
    --compress ^
    --order-by-primary ^
    -u local_user ^
```

```
    -p local_password | mysql ^
        --host aurora_cluster_endpoint_address ^
        --port 3306 ^
        -u RDS_user_name ^
        -p RDS_password
```

> **Note**
>
> Veillez bien à ce qu'il n'y ait pas d'espace entre l'option `-p` et le mot de passe saisi.

Utilisez les options `--host`, `--user (-u)`, `--port` et `-p` de la commande `mysql` pour spécifier le nom d'hôte, le nom d'utilisateur, le port et le mot de passe pour vous connecter à votre cluster de bases de données Aurora. Le nom d'hôte est le nom DNS du point de terminaison du cluster de bases de données Amazon Aurora, par exempl, `mydbcluster.cluster-123456789012.us-east-1.rds.amazonaws.com`. Vous pouvez trouver la valeur du point de terminaison dans les détails de cluster dans Amazon RDS Management Console.

4. Transformez l'instance de base de données MySQL source en instance accessible de nouveau en écriture :

```
mysql> SET GLOBAL read_only = OFF;
mysql> UNLOCK TABLES;
```

Pour plus d'informations sur les sauvegardes à utiliser avec la réplication, consultez Backing up a source or replica by making it read only dans la documentation MySQL.

5. Dans Amazon RDS Management Console, ajoutez l'adresse IP du serveur qui héberge la base de données MySQL source au groupe de sécurité VPC du cluster de bases de données Amazon Aurora. Pour plus d'informations sur la modification d'un groupe de sécurité de VPC, consultez Groupes de sécurité pour votre VPC dans le Guide de l'utilisateur Amazon Virtual Private Cloud.

   Il se peut aussi que vous ayez besoin de configurer votre réseau local pour autoriser les connexions à partir de l'adresse IP de votre cluster de bases de données Amazon Aurora, de telle sorte qu'elle puisse communiquer avec votre instance MySQL source. Pour rechercher l'adresse IP du cluster de bases de données Amazon Aurora, utilisez la commande `host`.

```
host aurora_endpoint_address
```

Le nom d'hôte est le nom DNS du point de terminaison du cluster de bases de données Amazon Aurora.

6. À l'aide du client de votre choix, connectez-vous à l'instance MySQL externe et créez un utilisateur MySQL à utiliser pour la réplication. Ce compte est utilisé exclusivement pour la réplication et doit être limité à votre domaine pour améliorer la sécurité. Voici un exemple de.

```
CREATE USER 'repl_user'@'example.com' IDENTIFIED BY 'password';
```

7. Pour l'instance MySQL externe, attribuez les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE` à votre utilisateur de réplication. Par exemple, pour accorder les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE` sur toutes les bases de données à l'utilisateur « `repl_user` » de votre domaine, émettez la commande suivante.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'example.com'
    IDENTIFIED BY 'password';
```

8. Avant de configurer la réplication, prenez un instantané manuel du cluster de bases de données Aurora MySQL qui constituera le réplica en lecture. Si vous avez besoin de rétablir la réplication avec le cluster de bases de données en tant que réplica en lecture, vous pouvez restaurer le cluster de bases de données Aurora MySQL à partir de cet instantané au lieu de devoir importer les données depuis votre instance de base de données MySQL vers un nouveau cluster de bases de données Aurora MySQL.
9. Transformez le cluster de bases de données Amazon Aurora en réplica. Connectez-vous au cluster de bases de données Amazon Aurora en tant qu'utilisateur principal et identifiez la base de données MySQL source en tant que source de réplication en utilisant les procédures mysql.rds_set_external_master (Aurora MySQL version 2) ou mysql.rds_set_external_source (Aurora MySQL version 3) et mysql.rds_start_replication.

   Utilisez le nom et la position du fichier binlog que vous avez déterminés à l'étape 2. Voici un exemple.

```
For Aurora MySQL version 2:
CALL mysql.rds_set_external_master ('mymasterserver.example.com', 3306,
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);

For Aurora MySQL version 3:
CALL mysql.rds_set_external_source ('mymasterserver.example.com', 3306,
    'repl_user', 'password', 'mysql-bin-changelog.000031', 107, 0);
```

10. Sur le cluster de bases de données Amazon Aurora, appelez la procédure mysql.rds_start_replication pour démarrer la réplication.

```
CALL mysql.rds_start_replication;
```

Après avoir établi la réplication entre votre instance de base de données MySQL source et votre cluster de bases de données Amazon Aurora, vous pouvez ajouter des réplicas Aurora à votre cluster de bases de données Amazon Aurora. Vous pouvez alors vous connecter aux réplicas Aurora pour dimensionner vos données en lecture. Pour plus d'informations sur la création d'un réplica Aurora, consultez Ajout de réplicas Aurora à un cluster de bases de données.

Optimisation de la réplication des journaux binaires pour Aurora MySQL

Découvrez ci-dessous comment optimiser les performances de réplication des journaux binaires et résoudre les problèmes connexes dans Aurora MySQL.

> **Tip**
>
> Pour continuer, il est nécessaire de connaître le mécanisme de réplication de journaux binaires MySQL et son fonctionnement. Pour en savoir plus, consultez Replication Implementation dans la documentation MySQL.

Réplication des journaux binaires multithread

Avec la réplication de journaux binaires multithreads, un thread SQL lit les événements du journal de relais et les met en file d'attente pour que les threads de travail SQL s'appliquent. Les threads de travail SQL sont gérés par un thread coordinateur. Si cela est possible, les événements du journal binaire sont appliqués en parallèle. Le niveau de parallélisme dépend de facteurs tels que la version, les paramètres, la conception du schéma et les caractéristiques de la charge de travail.

La réplication multithread des journaux binaires est prise en charge dans Aurora MySQL version 3 et dans Aurora MySQL version 2.12.1 ou ultérieure. Pour qu'un réplica multithread traite efficacement les événements du journal binaire en parallèle, vous devez configurer la source pour la réplication des journaux binaires multithread. La source doit également utiliser une version qui inclut les informations de parallélisme sur ses fichiers journaux binaires.

Lorsqu'une instance de base de données Aurora MySQL est configurée pour utiliser la réplication de journaux binaires, l'instance de réplica utilise par défaut la réplication à thread unique pour les

versions d'Aurora MySQL inférieures à 3.04. Pour activer la réplication multithread, mettez à jour le paramètre `replica_parallel_workers` pour que sa valeur soit supérieure à 1 dans votre groupe de paramètres personnalisés.

Pour Aurora MySQL 3.04 et versions ultérieures, la réplication est multithread par défaut. La valeur `replica_parallel_workers` est définie sur 4. Vous pouvez modifier ce paramètre dans votre groupe de paramètres personnalisés.

Pour augmenter la résilience de votre base de données face aux arrêts inattendus, nous vous recommandons d'activer la réplication GTID sur la source et d'autoriser les GTID sur le réplica. Pour autoriser la réplication GTID, définissez `gtid_mode` sur `ON_PERMISSIVE` à la fois sur la source et le réplica. Pour en savoir plus sur les réplications basées sur des identifiants de transaction globaux (GTID), consultez Utilisation de la réplication basée sur des identifiants de transaction globaux (GTID).

Les options de configuration suivantes vous permettent d'optimiser la réplication multithread. Pour plus d'informations, consultez Options et variables de réplication et de journalisation binaire dans le manuel de référence MySQL. Pour plus d'informations sur la réplication multithread, consultez le blog MySQL Improving the Parallel Applier with Writeset-based Dependency Tracking.

Les valeurs optimales des paramètres dépendent de plusieurs facteurs. Par exemple, les performances de la réplication des journaux binaires sont influencées par les caractéristiques de charge de travail de votre base de données, ainsi que la classe d'instance de base de données sur laquelle le réplica s'exécute. Dès lors, nous vous recommandons de tester minutieusement toutes les modifications apportées à ces paramètres de configuration avant d'appliquer de nouveaux paramètres à une instance de production :

- `binlog_format recommended value` : défini sur `ROW`
- `binlog_group_commit_sync_delay`
- `binlog_group_commit_sync_no_delay_count`
- `binlog_transaction_dependency_history_size`
- `binlog_transaction_dependency_tracking` : la valeur recommandée est `WRITESET`
- `replica_preserve_commit_order`
- `replica_parallel_type` : la valeur recommandée est `LOGICAL_CLOCK`
- `replica_parallel_workers`
- `replica_pending_jobs_size_max`

- `transaction_write_set_extraction` : la valeur recommandée est `XXHASH64`

Les caractéristiques de votre schéma et de votre charge de travail sont des facteurs qui influent sur la réplication en parallèle. Les facteurs les plus courants sont les suivants.

- Absence de clés primaires : RDS ne peut pas établir de dépendance entre les ensembles d'écritures pour les tables dépourvues de clés primaires. Avec le format `ROW`, une seule instruction à plusieurs lignes peut être exécutée avec une seule analyse de table complète au niveau de la source, mais il en résulte une analyse de table complète par ligne modifiée sur le réplica. L'absence de clés primaires réduit considérablement le débit de réplication.
- Présence de clés étrangères : s'il existe des clés étrangères, Amazon RDS ne peut pas utiliser la dépendance entre les ensembles d'écritures pour le parallélisme des tables avec la relation FK.
- Taille des transactions : si une seule transaction couvre des dizaines ou des centaines de mégaoctets ou de gigaoctets, le thread coordinateur et l'un des threads de travail peuvent passer beaucoup de temps à traiter uniquement cette transaction. Pendant ce temps, tous les autres threads de travail peuvent rester inactifs une fois qu'ils terminent le traitement de leurs transactions précédentes.

Dans Aurora MySQL version 3.06 et versions ultérieures, vous pouvez améliorer les performances des réplicas de journaux binaires lors de la réplication des transactions pour de tables de grande taille comportant plusieurs index secondaires. Cette fonctionnalité introduit un pool de threads permettant d'appliquer des modifications d'index secondaires en parallèle sur un réplica de journal binaire. Cette fonctionnalité est contrôlée par le paramètre de cluster de bases de données `aurora_binlog_replication_sec_index_parallel_workers`, qui contrôle le nombre total de threads parallèles disponibles pour appliquer les modifications d'index secondaires. Par défaut, ce paramètre est défini sur `0` (désactivé). L'activation de cette fonctionnalité ne nécessite pas le redémarrage de l'instance. Pour activer cette fonctionnalité, arrêtez la réplication en cours, définissez le nombre souhaité de threads de travail parallèles, puis relancez la réplication.

Optimisation de la réplication des journaux binaires

Dans Aurora MySQL versions 2.10 et ultérieures, Aurora applique automatiquement une optimisation connue sous le nom de cache d'I/O de journaux binaires à la réplication des journaux binaires. En mettant en cache les événements de journal binaire les plus récemment validés, cette optimisation est conçue pour améliorer les performances du thread de vidage des journaux binaires tout en limitant l'impact sur les transactions de premier plan sur l'instance source des journaux binaires.

> **Note**
>
> La mémoire utilisée pour cette fonction est indépendante du paramètre `binlog_cache` de MySQL.
> Cette fonction ne s'applique pas aux instances de base de données Aurora qui utilisent les classes d'instance `db.t2` et `db.t3`.

Vous n'avez pas besoin d'ajuster les paramètres de configuration pour activer cette optimisation. En particulier, si vous aviez défini le paramètre de configuration `aurora_binlog_replication_max_yield_seconds` sur une valeur différente de zéro dans des versions antérieures d'Aurora MySQL, redéfinissez-le sur zéro pour les versions actuellement disponibles.

Les variables d'état `aurora_binlog_io_cache_reads` et `aurora_binlog_io_cache_read_requests` vous aident à surveiller la fréquence à laquelle les données sont lues à partir du cache d'E/S des journaux binaires.

- `aurora_binlog_io_cache_read_requests` affiche le nombre de demandes de lecture d'I/O de journaux binaires provenant du cache.
- `aurora_binlog_io_cache_reads` affiche le nombre de lectures d'I/O de journaux binaires qui récupèrent des informations du cache.

La requête SQL suivante calcule le pourcentage de demandes de lecture de journaux binaires qui tirent parti des informations mises en cache. Dans ce cas, plus le ratio est proche de 100, mieux c'est.

```
mysql> SELECT
  (SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    WHERE VARIABLE_NAME='aurora_binlog_io_cache_reads')
  / (SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS
    WHERE VARIABLE_NAME='aurora_binlog_io_cache_read_requests')
  * 100
  as binlog_io_cache_hit_ratio;
+---------------------------+
| binlog_io_cache_hit_ratio |
+---------------------------+
|         99.99847949080622 |
```

```
+---------------------------+
```

La fonction de cache d'I/O de journaux binaires inclut également de nouvelles métriques liées aux threads de vidage des journaux binaires. Les threads de vidage sont les threads créés lorsque de nouveaux réplicas de journaux binaires sont connectés à l'instance source des journaux binaires.

Les métriques de thread de vidage sont imprimées dans le journal de la base de données toutes les 60 secondes avec le préfixe `[Dump thread metrics]`. Les métriques incluent des informations pour chaque réplica de journal binaire, telles que `Secondary_id`, `Secondary_uuid`, le nom du fichier journal binaire et la position que chaque réplica est en train de lire. Les métriques incluent également `Bytes_behind_primary`, qui représente la distance en octets entre la source de réplication et le réplica. Cette métrique mesure le décalage du thread d'I/O du réplica. Cette figure est différente du décalage du thread d'application SQL du réplica, représenté par la métrique `seconds_behind_master` sur le réplica du journal binaire. Vous pouvez déterminer si les réplicas de journaux binaires rattrapent la source ou sont en retard en vérifiant si la distance diminue ou augmente.

Journal de relais en mémoire

Dans Aurora MySQL 3.10 et versions ultérieures, Aurora introduit une optimisation connue sous le nom de « journal de relais en mémoire » afin d'améliorer le débit de réplication. Cette optimisation améliore les performances d'E/S du journal de relais en mettant en cache tout le contenu des journaux de relais intermédiaires en mémoire. Par conséquent, elle réduit la latence de validation en minimisant les opérations d'E/S du stockage, car le contenu du journal de relais reste facilement accessible en mémoire.

Par défaut, la fonctionnalité de journal de relais en mémoire est automatiquement activée pour les scénarios de réplication gérés par Aurora (y compris les déploiements bleu-vert, la réplication Aurora/Aurora et les réplicas entre régions) lorsque le réplica répond à l'une des configurations suivantes :

- Mode de réplication à thread unique (replica_parallel_workers = 0)
- Réplication multithread avec le mode GTID activé :
  - Positionnement automatique activé
  - Mode GTID activé sur le réplica
- Réplication basée sur des fichiers avec replica_preserve_commit_order = ON

La fonctionnalité de journal de relais en mémoire est prise en charge sur les classes d'instance supérieures à t3.large, mais n'est pas disponible sur les instances Aurora Serverless. La mémoire

tampon circulaire du journal de relais a une taille fixe de 128 Mo. Pour contrôler la consommation de mémoire de cette version, vous pouvez exécuter la requête suivante :

```
SELECT event_name, current_alloc FROM sys.memory_global_by_current_bytes WHERE
 event_name = 'memory/sql/relaylog_io_cache';
```

La fonctionnalité de journal de relais en mémoire est contrôlée par le paramètre aurora_in_memory_relaylog, qui peut être défini au niveau du cluster de bases de données ou de l'instance de base de données. Vous pouvez activer ou désactiver cette version de manière dynamique sans avoir à redémarrer l'instance :

1. Arrêt de la réplication en cours
2. Définissez aurora_in_memory_relaylog sur ON (pour l'activer) ou OFF (pour le désactiver) dans le groupe de paramètres.
3. Redémarrage de la réplication

Exemple :

```
CALL mysql.rds_stop_replication;
set aurora_in_memory_relaylog to ON to enable or OFF to disable in cluster parameter
 group
CALL mysql.rds_start_replication;
```

Même quand aurora_in_memory_relaylog est activé, la fonctionnalité de journal de relais en mémoire peut toujours être désactivée dans certaines conditions. Pour vérifier l'état actuel de cette fonctionnalité, vous pouvez utiliser la commande suivante :

```
SHOW GLOBAL STATUS LIKE 'Aurora_in_memory_relaylog_status';
```

Si la fonctionnalité est désactivée de façon inattendue, vous pouvez en identifier la raison en exécutant :

```
SHOW GLOBAL STATUS LIKE 'Aurora_in_memory_relaylog_disabled_reason';
```

Cette commande renvoie un message expliquant pourquoi la fonctionnalité est actuellement désactivée.

Configuration du binlog amélioré pour Aurora MySQL

Le binlog amélioré réduit la surcharge de performances de calcul provoquée par l'activation de binlog, qui peut atteindre 50 % dans certains cas. Avec le binlog amélioré, cette surcharge peut être réduite à environ 13 %. Pour réduire la surcharge, le binlog amélioré écrit les journaux binaires et de transactions sur le stockage en parallèle, ce qui minimise les données écrites au moment de la validation de la transaction.

L'utilisation d'un binlog amélioré améliore également le temps de récupération de la base de données après les redémarrages et les basculements de 99 % par rapport au binlog MySQL communautaire. Le binlog amélioré est compatible avec les charges de travail existantes basées sur binlog et vous interagissez avec lui de la même manière que vous interagissez avec le binlog MySQL communautaire.

Le binlog amélioré est disponible sur les versions Aurora MySQL 3.03.1 et ultérieures.

Rubriques

Configuration des paramètres d'un binlog amélioré

Vous pouvez basculer entre le binlog MySQL communautaire et le binlog amélioré en activant/désactivant les paramètres du binlog amélioré. Les utilisateurs existants de binlog peuvent continuer à lire et à utiliser les fichiers binlog sans aucune interruption dans la séquence des fichiers binlog.

Pour activer le binlog amélioré, définissez les paramètres suivants :

Paramètre	Par défaut	Description
`binlog_format`	–	Définissez le paramètre `binlog_format` au format de journalisation binaire de votre choix pour activer le

Paramètre	Par défaut	Description
		binlog amélioré. Assurez-vous que le `binlog_format parameter` n'est pas réglé sur OFF. Pour plus d'informations, consultez Configuration de la journalisation binaire Aurora MySQL.
`aurora_enhanced_binlog`	`0`	Définissez la valeur de ce paramètre sur 1 dans le groupe de paramètres du cluster de bases de données associé au cluster Aurora MySQL. Lorsque vous modifiez la valeur de ce paramètre, vous devez redémarrer l'instance de rédacteur lorsque la valeur `DBClusterParameterGroupStatus` est affichée comme `pending-reboot` .
`binlog_backup`	`1`	Désactivez ce paramètre pour activer le binlog amélioré. Pour ce faire, définissez la valeur de ce paramètre sur `0`.
`binlog_replication_globaldb`	`1`	Désactivez ce paramètre pour activer le binlog amélioré. Pour ce faire, définissez la valeur de ce paramètre sur `0`.

> **Important**
>
> Vous ne pouvez désactiver les paramètres `binlog_backup` et `binlog_replication_globaldb` que lorsque vous utilisez le binlog amélioré.

Pour désactiver le journal binaire amélioré, définissez les paramètres suivants :

Paramètre	Description
`aurora_enhanced_binlog`	Définissez la valeur de ce paramètre sur `0` dans le groupe de paramètres du cluster de bases de données associé au cluster Aurora MySQL. À chaque fois que vous modifiez la valeur de ce paramètre, vous devez redémarrer l'instance de rédacteur lorsque la valeur `DBClusterParameterGroupStatus` est affichée comme `pending-reboot` .
`binlog_backup`	Activez ce paramètre lorsque vous désactivez le binlog amélioré. Pour ce faire, définissez la valeur de ce paramètre sur 1.
`binlog_replication_globaldb`	Activez ce paramètre lorsque vous désactivez le binlog amélioré. Pour ce faire, définissez la valeur de ce paramètre sur 1.

Pour vérifier si le binlog amélioré est activé, utilisez la commande suivante dans le client MySQL :

```
mysql>show status like 'aurora_enhanced_binlog';

+------------------------+--------+
| Variable_name          | Value  |
+------------------------+--------+
| aurora_enhanced_binlog | ACTIVE |
+------------------------+--------+
1 row in set (0.00 sec)
```

Lorsque le binlog amélioré est activé, la sortie affiche `ACTIVE` pour`aurora_enhanced_binlog`.

Autres paramètres connexes

Lorsque vous activez le binlog amélioré, les paramètres suivants sont affectés :

- Le paramètre `max_binlog_size` est visible mais non modifiable. Sa valeur par défaut `134217728` est automatiquement ajustée sur `268435456` lorsque le binlog amélioré est activé.
- Contrairement au binlog MySQL communautaire, `binlog_checksum` n'agit pas comme un paramètre dynamique lorsque le binlog amélioré est activé. Pour que la modification de ce paramètre soit prise en compte, vous devez redémarrer manuellement le cluster de bases de données, même si la `ApplyMethod` est `immediate`.
- La valeur que vous définissez sur le paramètre `binlog_order_commits` n'a aucun effet sur l'ordre des validations lorsque le binlog amélioré est activé. Les validations sont toujours ordonnées sans aucune autre incidence sur les performances.

Différences entre le binlog amélioré et le binlog MySQL communautaire

Le binlog amélioré interagit différemment avec les clones, les sauvegardes et la base de données globale Aurora par rapport au binlog MySQL communautaire. Nous vous recommandons de comprendre les différences suivantes avant d'utiliser le binlog amélioré.

- Les fichiers binlog améliorés du cluster de bases de données source ne sont pas disponibles sur un cluster de bases de données cloné.
- Les fichiers binlog améliorés ne sont pas inclus dans les sauvegardes Aurora. Par conséquent, les fichiers binlog améliorés du cluster de bases de données source ne sont pas disponibles après la restauration d'un cluster de bases de données, même avec une période de conservation définie.
- Lorsqu'ils sont utilisés avec une base de données globale Aurora, les fichiers binlog améliorés du cluster de bases de données principal ne sont pas répliqués vers le cluster de bases de données des régions secondaires.

Exemples

Les exemples suivants montrent les différences entre le binlog amélioré et le binlog MySQL communautaire.

Sur un cluster de bases de données restauré ou cloné

Lorsque le binlog amélioré est activé, les fichiers binlog historiques ne sont pas disponibles dans le cluster de bases de données restauré ou cloné. Après une opération de restauration ou de clonage, si le binlog est activé, le nouveau cluster de bases de données commence à écrire sa propre séquence de fichiers binlog, en commençant par 1 (mysql-bin-changelog.000001).

Pour activer le binlog amélioré après une opération de restauration ou de clonage, définissez les paramètres du cluster de bases de données requis sur le cluster de bases de données restauré ou cloné. Pour plus d'informations, consultez Configuration des paramètres d'un binlog amélioré.

Example Exemple : opération de clonage ou de restauration effectuée lorsque le binlog amélioré est activé

Cluster de bases de données source :

```
mysql> show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        |
| mysql-bin-changelog.000003 |       156 | No        |
| mysql-bin-changelog.000004 |       156 | No        | --> Enhanced Binlog turned on
| mysql-bin-changelog.000005 |       156 | No        | --> Enhanced Binlog turned on
| mysql-bin-changelog.000006 |       156 | No        | --> Enhanced Binlog turned on
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Sur un cluster de bases de données restauré ou cloné, les fichiers binlog ne sont pas sauvegardés lorsque le journal binaire amélioré est activé. Pour éviter toute discontinuité dans les données du binlog, les fichiers binlog écrits avant l'activation du binlog amélioré ne sont pas non plus disponibles.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        | --> New sequence of Binlog files
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Example Exemple : opération de clonage ou de restauration effectuée lorsque le binlog amélioré est désactivé

Cluster de bases de données source :

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000003 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Le journal binaire amélioré est désactivé après `mysql-bin-changelog.000003`. Sur un cluster de bases de données restauré ou cloné, les fichiers binlog écrits après la désactivation du binlog amélioré sont disponibles.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Sur une Amazon Aurora Global Database

Sur une Amazon Aurora Global Database, les données du binlog du cluster de bases de données principal ne sont pas répliquées vers les clusters de bases de données secondaires. Après un processus de basculement entre régions, les données du binlog ne sont pas disponibles dans le cluster de bases de données principal récemment promu. Si le binlog est activé, le nouveau cluster de bases de données récemment promu commence sa propre séquence de fichiers binlog, en commençant par 1 (mysql-bin-changelog.000001).

Pour activer le binlog amélioré après un basculement, vous devez définir les paramètres de cluster de bases de données requis sur le cluster de bases de données secondaire. Pour plus d'informations, consultez Configuration des paramètres d'un binlog amélioré.

Example Exemple : l'opération de basculement global de la base de données est effectuée lorsque le binlog amélioré est activé

Ancien cluster de bases de données principal (avant le basculement) :

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        |
| mysql-bin-changelog.000003 |       156 | No        |
| mysql-bin-changelog.000004 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000005 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000006 |       156 | No        | --> Enhanced Binlog enabled
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Nouveau cluster de bases de données principal (après le basculement) :

Les fichiers binlog ne sont pas répliqués vers les régions secondaires lorsque le binlog amélioré est activé. Pour éviter toute discontinuité dans les données du binlog, les fichiers binlog écrits avant l'activation du binlog amélioré ne sont pas disponibles.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
```

```
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        | --> Fresh sequence of Binlog
 files
+----------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Example Exemple : l'opération de basculement global de la base de données est effectuée lorsque le binlog amélioré est désactivé

Cluster de bases de données source :

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000001 |       156 | No        |
| mysql-bin-changelog.000002 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000003 |       156 | No        | --> Enhanced Binlog enabled
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
6 rows in set (0.00 sec)
```

Cluster de bases de données restauré ou cloné :

Le journal binaire amélioré est désactivé après `mysql-bin-changelog.000003`. Les fichiers binlog qui sont écrits après la désactivation du binlog amélioré sont répliqués et sont disponibles dans le cluster de bases de données récemment promu.

```
mysql>show binary logs;

+----------------------------+-----------+-----------+
| Log_name                   | File_size | Encrypted |
+----------------------------+-----------+-----------+
| mysql-bin-changelog.000004 |       156 | No        |
| mysql-bin-changelog.000005 |       156 | No        |
```

```
| mysql-bin-changelog.000006 |       156 | No        |
+----------------------------+-----------+-----------+
3 rows in set (0.00 sec)
```

Métriques Amazon CloudWatch pour le binlog amélioré

Les métriques Amazon CloudWatch suivantes sont publiées uniquement lorsque le binlog amélioré est activé.

Métrique CloudWatch	Description	Unités
ChangeLogBytesUsed	Volume de stockage utilisé par le binlog amélioré.	Octets
ChangeLogReadIOPs	Nombre d'opérations d'E/S de lecture réalisées dans le binlog amélioré à 5 minutes d'intervalles.	Compte par 5 minutes
ChangeLogWriteIOPs	Nombre d'opérations d'E/S d'écriture disque réalisées dans le binlog amélioré à 5 minutes d'intervalles.	Compte par 5 minutes

Limites du binlog amélioré

Les limites suivantes s'appliquent aux clusters de bases de données Amazon Aurora lorsque le binlog amélioré est activé.

- Le binlog amélioré est uniquement pris en charge sur les versions Aurora MySQL 3.03.1 et ultérieures.
- Les fichiers binlog améliorés écrits sur le cluster de bases de données principal ne sont pas copiés vers les clusters de bases de données clonés ou restaurés.
- Lorsqu'ils sont utilisés avec Amazon Aurora Global Database, les fichiers binlog améliorés du cluster de bases de données principal ne sont pas répliqués vers les clusters de bases de données secondaires. Par conséquent, après le processus de basculement, les données historiques du binlog ne sont pas disponibles dans le nouveau cluster de bases de données principal.

- Les paramètres de configuration du binlog suivants sont ignorés :
  - `binlog_group_commit_sync_delay`
  - `binlog_group_commit_sync_no_delay_count`
  - `binlog_max_flush_queue_time`
- Vous ne pouvez pas supprimer ou renommer une table corrompue dans une base de données. Pour supprimer ces tables, vous pouvez contacter Support.
- Le cache d'E/S du binlog est désactivé lorsque le binlog amélioré est activé. Pour plus d'informations, consultez Optimisation de la réplication des journaux binaires pour Aurora MySQL.

> **Note**
>
> Le binlog amélioré fournit de meilleures performances de lecture similaires à celles du cache d'E/S du binlog et de meilleures performances d'écriture.

- La fonction de retour en arrière n'est pas prise en charge. Le binlog amélioré ne peut pas être activé dans un cluster de bases de données dans les conditions suivantes :
  - Cluster de bases de données avec la fonctionnalité de retour en arrière actuellement activée.
  - Cluster de bases de données avec la fonctionnalité de retour en arrière précédemment activée mais maintenant désactivée.
  - Cluster de bases de données restauré à partir d'un cluster de bases de données source ou d'un instantané avec la fonction de retour en arrière activée.

Utilisation de la réplication basée sur des identifiants de transaction globaux (GTID)

Le contenu ci-dessous explique comment utiliser les identifiants de transaction globaux (GTID) avec la réplication des journaux binaires (binlog) entre un cluster Aurora MySQL et une source externe.

> **Note**
>
> Pour Aurora, vous pouvez uniquement utiliser cette fonction avec des clusters Aurora MySQL qui utilisent la réplication des journaux binaires vers/à partir d'une base de données MySQL externe. L'autre base de données peut être une instance Amazon RDS MySQL, une base de données MySQL sur site, ou un cluster de bases de données Aurora dans une autre Région AWS. Pour apprendre à configurer ce type de réplication, consultez Réplication entre Aurora

et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

Si vous utilisez la réplication des journaux binaires, mais que vous ne maîtrisez pas la réplication GTID avec MySQL, consultez Réplication avec des identifiants de transaction globaux dans la documentation MySQL.

La réplication GTID est prise en charge pour Aurora MySQL versions 2 et 3.

Rubriques

- Présentation des identifiants de transaction globaux (GTID)
- Paramètres pour la réplication basée sur des identifiants de transaction globaux (GTID)
- Activation de la réplication GTID pour un cluster Aurora MySQL
- Désactivation de la réplication GTID pour un cluster de bases de données Aurora MySQL

Présentation des identifiants de transaction globaux (GTID)

Les identifiants de transaction globaux (GTID) sont des identifiants uniques générés pour des transactions MySQL validées. Vous pouvez utiliser ces identifiants pour simplifier et faciliter la résolution des problèmes liés à la réplication des journaux binaires.

> **Note**
>
> Lorsqu'Aurora synchronise des données entre les instances de base de données d'un cluster, ce mécanisme de réplication n'implique pas le journal binaire (binlog). Pour Aurora MySQL, la réplication GTID s'applique uniquement lorsque vous utilisez également la réplication des journaux binaires pour répliquer à l'intérieur ou à l'extérieur d'un cluster de bases de données Aurora MySQL à partir d'une base de données externe compatible avec MySQL.

MySQL utilise deux types différents de transactions pour la réplication des journaux binaires :

- Transactions GTID – Transactions identifiées par un identifiant de transaction global (GTID).
- Transactions anonymes – Transactions auxquelles aucun identifiant de transaction global (GTID) n'est associé.

Dans une configuration de réplication, les GTID sont uniques parmi toutes les instances de base de données. Les GTID simplifient la configuration de réplication dans la mesure où, lorsque vous les utilisez, vous n'avez pas à vous référer aux positions des fichiers journaux. Les GTID facilitent également le suivi des transactions répliquées et déterminent si l'instance source et les réplicas sont cohérents.

En règle générale, vous utilisez la réplication GTID avec Aurora lorsque vous effectuez une réplication à partir d'une base de données externe compatible avec MySQL dans un cluster Aurora. Vous pouvez procéder à la configuration de cette réplication dans le cadre d'une migration d'une base de données sur site ou Amazon RDS vers Aurora MySQL. Si la base de données externe utilise déjà des identifiants de transaction globaux (GTID), l'utilisation de la réplication GTID pour le cluster Aurora permet de simplifier le processus de réplication.

Vous configurez la réplication GTID pour un cluster Aurora MySQL en commençant par définir les paramètres de configuration appropriés dans un groupe de paramètres de cluster de bases de données. Vous associez ensuite ce groupe de paramètres au cluster.

Paramètres pour la réplication basée sur des identifiants de transaction globaux (GTID)

Utilisez les paramètres suivants pour configurer une réplication GTID.

Paramètre	Valeurs valides	Description
`gtid_mode`	`OFF`, `OFF_PERMISSIVE` , `ON_PERMISSIVE` , `ON`	`OFF` spécifie que les nouvelles transactions sont des transactions anonymes (et n'ont donc pas de GTID), et qu'une transaction doit être anonyme pour être répliquée. `OFF_PERMISSIVE` spécifie que les nouvelles transactions sont des transactions anonymes, mais que toutes les transactions peuvent être répliquées. `ON_PERMISSIVE` spécifie que les nouvelles transactions sont des transactions GTID, mais que toutes les transactions peuvent être répliquées.

Paramètre	Valeurs valides	Description
		ON spécifie que les nouvelles transactions sont des transactions GTID, et qu'une transacti on doit être une transaction GTID pour être répliquée.
enforce_gtid_consistency	OFF, ON, WARN	OFF autorise les transactions à enfreindre la cohérence GTID. ON interdit aux transactions d'enfreindre la cohérence GTID. WARN autorise les transactions à enfreindre la cohérence GTID mais génère un avertissement lorsqu'une infraction se produit.

> **Note**
>
> Dans AWS Management Console, le paramètre gtid_mode apparaît sous la forme gtid-mode.

Pour la réplication GTID, utilisez ces paramètres pour le groupe de paramètres de votre cluster de bases de données Aurora MySQL :

- ON et ON_PERMISSIVE s'appliquent uniquement à la réplication sortante d'un cluster Aurora MySQL. Ces deux valeurs forcent votre cluster de base de données Aurora à utiliser des identifiants de transaction globaux (GTID) pour les transactions répliquées sur une base de données externe. ON exige que la base de données externe utilise également la réplication GTID. ON_PERMISSIVE rend la réplication GTID facultative sur la base de données externe.
- S'il est défini, OFF_PERMISSIVE indique que votre cluster de bases de données Aurora peuvent accepter la réplication entrante à partir d'une base de données externe. que cette dernière utilise la réplication GTID ou non.
- S'il est défini, OFF indique que votre cluster de bases de données Aurora accepte uniquement la réplication entrante à partir de bases de données externes qui n'utilisent pas la réplication GTID.

> **Tip**
>
> La réplication entrante est le scénario de réplication des journaux binaires le plus fréquent pour les clusters Aurora MySQL. Pour la réplication entrante, il est recommandé de définir le mode GTID sur `OFF_PERMISSIVE`. Cette valeur autorise la réplication entrante à partir de bases de données externes, quels que soient les paramètres GTID au niveau de la source de réplication.

Pour plus d'informations sur les groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora.

Activation de la réplication GTID pour un cluster Aurora MySQL

Lorsque la réplication GTID est activée pour un cluster de bases de données Aurora MySQL, les paramètres GTID s'appliquent à la réplication des journaux binaires entrante et sortante.

Pour activer la réplication GTID pour un cluster Aurora MySQL

1. Créez ou modifiez un groupe de paramètres de cluster de bases de données en définissant les valeurs suivantes :
   - `gtid_mode` – `ON` ou `ON_PERMISSIVE`
   - `enforce_gtid_consistency` – `ON`
2. Associez le groupe de paramètres de cluster de bases de données au cluster Aurora MySQL. Pour ce faire, suivez les procédures décrites dans Groupes de paramètres pour Amazon Aurora.
3. (Facultatif) Indiquez comment affecter des GTID à des transactions qui n'en incluent pas. Pour ce faire, appelez la procédure stockée dans mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL version 3).

Désactivation de la réplication GTID pour un cluster de bases de données Aurora MySQL

Vous pouvez désactiver la réplication GTID pour un cluster de bases de données Aurora MySQL. Dans ce cas, le cluster Aurora ne peut pas effectuer de réplication des journaux binaires entrante ou sortante avec des bases de données externes qui utilisent la réplication GTID.

> **Note**
>
> Dans la procédure suivante, un réplica en lecture représente la cible de réplication dans une configuration Aurora avec une réplication des journaux binaires vers/à partir d'une base de données externe. Il ne représente pas les instances de base de données de réplica Aurora en lecture seule. Par exemple, lorsqu'un cluster Aurora accepte la réplication entrante à partir d'une source externe, l'instance principale d'Aurora sert de réplica en lecture pour la réplication des journaux binaires.

Pour plus de détails sur les procédures stockées mentionnées dans la présente section, consultez Référence des procédures stockées Aurora MySQL.

Pour désactiver la réplication GTID pour un cluster de bases de données Aurora MySQL

1. Sur les réplicas Aurora, exécutez la procédure suivante :

   Pour la version 3

   ```
   CALL mysql.rds_set_source_auto_position(0);
   ```

   Pour la version 2

   ```
   CALL mysql.rds_set_master_auto_position(0);
   ```

2. Réinitialisez `gtid_mode` sur `ON_PERMISSIVE`.

   a. Assurez-vous que le groupe de paramètres de cluster de bases de données associé au cluster Aurora MySQL contient le paramètre `gtid_mode` défini sur `ON_PERMISSIVE`.

      Pour plus d'informations sur la définition des paramètres de configuration à l'aide de groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora.

   b. Redémarrez le cluster de bases de données Aurora MySQL.

3. Réinitialisez `gtid_mode` sur `OFF_PERMISSIVE`.

   a. Assurez-vous que le groupe de paramètres de cluster de bases de données associé au cluster Aurora MySQL contient le paramètre `gtid_mode` défini sur `OFF_PERMISSIVE`.

   b. Redémarrez le cluster de base de données Aurora MySQL.

4. Attendez que toutes les transactions GTID soient appliquées sur l'instance principale d'Aurora. Pour vérifier qu'elles ont été appliquées, procédez comme suit :

   a. Sur l'instance Auroraprincipale, exécutez la commande `SHOW MASTER STATUS`.

      Votre sortie doit ressembler à ce qui suit.

      ```
      File                        Position
      ------------------------------------
      mysql-bin-changelog.000031      107
      ------------------------------------
      ```

      Notez le fichier et la position dans votre sortie.

   b. Sur chaque réplica en lecture, utilisez les informations de fichier et de position de l'instance source lors de l'étape précédente pour exécuter la requête suivante :

      Pour la version 3

      ```
      SELECT SOURCE_POS_WAIT('file', position);
      ```

      Pour la version 2

      ```
      SELECT MASTER_POS_WAIT('file', position);
      ```

      Par exemple, si votre fichier se nomme `mysql-bin-changelog.000031` et que sa position est `107`, exécutez l'instruction suivante :

      Pour la version 3

      ```
      SELECT SOURCE_POS_WAIT('mysql-bin-changelog.000031', 107);
      ```

      Pour la version 2

      ```
      SELECT MASTER_POS_WAIT('mysql-bin-changelog.000031', 107);
      ```

5. Réinitialisez les paramètres GTID pour désactiver la réplication GTID.

a. Assurez-vous que le groupe de paramètres de cluster de bases de données associé au cluster Aurora MySQL contient les valeurs suivantes :

   - `gtid_mode` – `OFF`
   - `enforce_gtid_consistency` – `OFF`

b. Redémarrez le cluster de base de données Aurora MySQL.

Utilisation du transfert d’écriture local dans un cluster de bases de données Amazon Aurora MySQL

Le transfert d’écriture local (intracluster) permet à vos applications d’émettre des transactions de lecture/écriture directement sur un réplica Aurora. Ces transactions sont ensuite transférées à l’instance de base de données d’enregistreur pour être validées. Vous pouvez utiliser le transfert d’écriture local lorsque vos applications exigent une cohérence de lecture après écriture, qui est la capacité à lire la dernière écriture dans une transaction.

Les réplicas en lecture reçoivent des mises à jour de manière asynchrone de la part de l’enregistreur. Sans transfert d’écriture, vous devez traiter toutes les lectures qui nécessitent une cohérence de lecture après écriture sur l’instance de base de données d’enregistreur. Ou vous devez développer une logique d’application personnalisée complexe pour tirer parti de plusieurs réplicas en lecture pour assurer la capacité de mise à l’échelle. Vos applications doivent diviser entièrement l’ensemble du trafic de lecture et d’écriture, en conservant deux ensembles de connexions à la base de données pour envoyer le trafic au point de terminaison correct. Cette surcharge de développement complique la conception de l’application lorsque les requêtes font partie d’une seule session logique, ou transaction, au sein de l’application. De plus, comme le retard de réplication peut différer entre les réplicas en lecture, il est difficile d’obtenir une cohérence de lecture globale entre toutes les instances dans la base de données.

Le transfert d’écriture évite d’avoir à diviser ces transactions ou à les envoyer exclusivement à l’enregistreur, ce qui simplifie le développement des applications. Cette nouvelle fonctionnalité permet d’effectuer facilement la mise à l’échelle en lecture des charges de travail qui doivent lire la dernière écriture dans une transaction et qui ne sont pas sensibles à la latence d’écriture.

Le transfert d’écriture local est différent du transfert d’écriture global, qui transfère les écritures d’un cluster de bases de données secondaire vers le cluster de bases de données principal dans une base de données globale Aurora. Vous pouvez utiliser le transfert d’écriture local dans un cluster de bases de données faisant partie d’une base de données globale Aurora. Pour plus d’informations, consultez Utilisation du transfert d'écriture dans une base de données globale Amazon Aurora.

Le transfert d’écriture local nécessite Aurora MySQL version 3.04 ou ultérieure.

Rubriques

- Compatibilité des applications et de SQL avec le transfert d'écriture
- Niveaux d'isolement pour le transfert d'écriture
- Cohérence de lecture pour le transfert d'écriture
- Exécution d'instructions en plusieurs parties avec le transfert d'écriture
- Transactions avec transfert d'écriture
- Paramètres de configuration pour le transfert d'écriture
- Métriques Amazon CloudWatch et variables d'état Aurora MySQL pour le transfert d'écriture
- Identification des transactions et des requêtes transférées

Activation du transfert d'écriture local

Par défaut, le transfert d'écriture local n'est pas activé pour les clusters de bases de données Aurora MySQL. Vous activez le transfert d'écriture local au niveau du cluster, et non au niveau de l'instance.

> **Important**
>
> Vous pouvez également activer le transfert d'écriture local pour les réplicas en lecture entre régions qui utilisent la journalisation binaire, mais les opérations d'écriture ne sont pas transférées vers la Région AWS source. Ils sont transmis à l'instance de base de données d'enregistreur du cluster de réplicas en lecture des journaux binaires.
> Utilisez cette méthode uniquement si vous avez un cas d'utilisation pour écrire dans le réplica en lecture des journaux binaires dans la Région AWS secondaire. Dans le cas contraire, vous risquez de vous retrouver dans un scénario incohérent appelé « split-brain », où les ensembles de données répliqués ne sont pas cohérents les uns avec les autres.
> Nous vous recommandons d'utiliser le transfert d'écriture global avec les bases de données globales à la place du transfert d'écriture local pour les réplicas en lecture entre régions, sauf en cas de nécessité absolue. Pour plus d'informations, consultez Utilisation du transfert d'écriture dans une base de données globale Amazon Aurora.

Console

À l'aide de la AWS Management Console, cochez la case Activer le transfert d'écriture local sous Transfert d'écriture de réplica en lecture lorsque vous créez ou modifiez un cluster de bases de données.

AWS CLI

Pour activer le transfert d'écriture à l'aide de l'interface AWS CLI, utilisez l'option `--enable-local-write-forwarding`. Cette option est utile lorsque vous créez un nouveau cluster de bases de données à l'aide de la commande `create-db-cluster`. Elle est également utile lorsque vous modifiez un cluster de bases de données existant à l'aide de la commande `modify-db-cluster`. Vous pouvez désactiver le transfert d'écriture en utilisant l'option `--no-enable-local-write-forwarding` avec ces mêmes commandes CLI.

L'exemple suivant crée un cluster de bases de données Aurora MySQL avec le transfert d'écriture activé.

```
aws rds create-db-cluster \
  --db-cluster-identifier write-forwarding-test-cluster \
  --enable-local-write-forwarding \
  --engine aurora-mysql \
  --engine-version 8.0.mysql_aurora.3.04.0 \
  --master-username myuser \
  --master-user-password mypassword \
  --backup-retention 1
```

Vous créez ensuite des instances de base de données d'enregistreur et de lecteur afin de pouvoir utiliser le transfert d'écriture. Pour plus d'informations, consultez Création d'un cluster de bases de données Amazon Aurora.

API RDS

Pour activer le transfert d'écriture à l'aide de l'API Amazon RDS, définissez le paramètre `EnableLocalWriteForwarding` sur `true`. Ce paramètre agit lorsque vous créez un nouveau cluster de bases de données à l'aide de l'opération `CreateDBCluster`. Il agit également lorsque vous modifiez un cluster de bases de données existant à l'aide de l'opération `ModifyDBCluster`. Vous pouvez désactiver le transfert d'écriture en définissant le paramètre `EnableLocalWriteForwarding` sur `false`.

Activation du transfert d'écriture pour les sessions de base de données

Le paramètre `aurora_replica_read_consistency` est un paramètre de base de données et un paramètre de cluster de bases de données qui permet le transfert d'écriture. Vous pouvez spécifier `EVENTUAL`, `SESSION` ou `GLOBAL` pour le niveau de cohérence de lecture. Pour en savoir plus sur les niveaux de cohérence, consultez la section Cohérence de lecture pour le transfert d'écriture.

Les règles suivantes s'appliquent à ce paramètre :

- La valeur par défaut est '' (null).
- Le transfert d'écriture est disponible seulement si vous définissez `aurora_replica_read_consistency` sur `EVENTUAL`, `SESSION` ou `GLOBAL`. Ce paramètre n'est pertinent que dans les instances de lecteur de clusters de bases de données dont le transfert d'écriture est activé.
- Vous ne pouvez pas définir ce paramètre (lorsqu'il est vide) ni le désactiver (lorsqu'il est déjà défini) dans une transaction à plusieurs instructions. Vous pouvez le modifier en remplaçant une valeur valide par une autre au cours d'une telle transaction, mais nous ne recommandons pas cette action.

Vérification de l'activation du transfert d'écriture dans un cluster de bases de données

Pour déterminer si vous pouvez utiliser le transfert d'écriture dans un cluster de bases de données, vérifiez que le cluster possède l'attribut `LocalWriteForwardingStatus` réglé sur `enabled`.

Dans la AWS Management Console, dans l'onglet Configuration de la page de détails du cluster, vous pouvez voir l'état Activé pour Transfert local d'écriture de réplica en lecture.

Pour voir l'état du paramètre de transfert d'écriture pour tous vos clusters, exécutez la commande AWS CLI suivante.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,LocalWriteForwardingStatus:LocalWriteForwardingStatus}

[
    {
        "LocalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "write-forwarding-test-cluster-1"
    },
    {
        "LocalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "write-forwarding-test-cluster-2"
    },
```

```
    {
        "LocalWriteForwardingStatus": "requested",
        "DBClusterIdentifier": "test-global-cluster-2"
    },
    {
        "LocalWriteForwardingStatus": "null",
        "DBClusterIdentifier": "aurora-mysql-v2-cluster"
    }
]
```

Un cluster de bases de données peut avoir les valeurs suivantes pour `LocalWriteForwardingStatus` :

- `disabled` : le transfert d'écriture est désactivé.
- `disabling` : le transfert d'écriture est en cours de désactivation.
- `enabled` : le transfert d'écriture est activé.
- `enabling` : le transfert d'écriture est en cours d'activation.
- `null` : le transfert d'écriture n'est pas disponible pour ce cluster de bases de données.
- `requested` : le transfert d'écriture a été demandé, mais n'est pas encore actif.

Compatibilité des applications et de SQL avec le transfert d'écriture

Vous pouvez utiliser les types d'instructions SQL suivants avec le transfert d'écriture :

- Instructions DML (Data Manipulation Language) comme `INSERT`, `DELETE` et `UPDATE`. Il existe certaines restrictions concernant les propriétés de ces instructions que vous pouvez utiliser avec le transfert d'écriture, comme décrit ci-dessous.
- Instructions `SELECT ... LOCK IN SHARE MODE` et `SELECT FOR UPDATE`.
- Instructions `PREPARE` et `EXECUTE`.

Certaines instructions ne sont pas autorisées ou peuvent produire des résultats obsolètes lorsque vous les utilisez dans un cluster de bases de données avec transfert d'écriture. En outre, les fonctions et les procédures définies par l'utilisateur ne sont pas prises en charge. Ainsi, le paramètre `EnableLocalWriteForwarding` est désactivé par défaut pour les clusters de bases de données. Avant de l'activer, vérifiez que votre code d'application n'est affecté par aucune de ces restrictions.

Les restrictions suivantes s'appliquent aux instructions SQL que vous utilisez avec le transfert d'écriture. Dans certains cas, vous pouvez utiliser les instructions sur des clusters de bases de données avec le transfert d'écriture activé. Cette approche fonctionne si le transfert d'écriture n'est pas activé dans la session par le paramètre de configuration `aurora_replica_read_consistency`. Si vous essayez d'utiliser une instruction alors qu'elle n'est pas autorisée en raison du transfert d'écriture, vous verrez un message d'erreur semblable au suivant :

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'operation with write
 forwarding'.
```

Langage de définition de données (DDL)

Connectez-vous à l'instance de base de données d'enregistreur pour exécuter des instructions DDL. Vous ne pouvez pas les exécuter à partir des instances de base de données de lecteur.

Mise à jour d'une table permanente à l'aide des données d'une table temporaire

Vous pouvez utiliser des tables temporaires sur des clusters de bases de données avec le transfert d'écriture activé. Toutefois, vous ne pouvez pas utiliser une instruction DML pour modifier une table permanente si l'instruction fait référence à une table temporaire. Par exemple, vous ne pouvez pas utiliser une instruction `INSERT ... SELECT` qui prend les données d'une table temporaire.

Transactions XA

Vous ne pouvez pas utiliser les instructions suivantes sur un cluster de bases de données lorsque le transfert d'écriture est activé dans la session. Vous pouvez utiliser ces instructions sur des clusters de bases de données pour lesquels le transfert d'écriture n'est pas activé, ou dans des sessions où le paramètre `aurora_replica_read_consistency` est vide. Avant d'activer le transfert d'écriture dans une session, vérifiez si votre code utilise ces instructions.

```
XA {START|BEGIN} xid [JOIN|RESUME]
XA END xid [SUSPEND [FOR MIGRATE]]
XA PREPARE xid
XA COMMIT xid [ONE PHASE]
XA ROLLBACK xid
XA RECOVER [CONVERT XID]
```

Instructions LOAD pour des tables permanentes

Vous ne pouvez pas utiliser les instructions suivantes sur un cluster de bases de données avec le transfert d'écriture activé.

```
LOAD DATA INFILE 'data.txt' INTO TABLE t1;
LOAD XML LOCAL INFILE 'test.xml' INTO TABLE t1;
```

Instructions de plugin

Vous ne pouvez pas utiliser les instructions suivantes sur un cluster de bases de données avec le transfert d'écriture activé.

```
INSTALL PLUGIN example SONAME 'ha_example.so';
UNINSTALL PLUGIN example;
```

Instructions SAVEPOINT

Vous ne pouvez pas utiliser les instructions suivantes sur un cluster de bases de données lorsque le transfert d'écriture est activé dans la session. Vous pouvez utiliser ces instructions sur des clusters de bases de données pour lesquels le transfert d'écriture n'est pas activé, ou dans des sessions où le paramètre `aurora_replica_read_consistency` est vide. Vérifiez si votre code utilise ces instructions avant d'activer le transfert d'écriture dans une session.

```
SAVEPOINT t1_save;
ROLLBACK TO SAVEPOINT t1_save;
RELEASE SAVEPOINT t1_save;
```

Niveaux d'isolement pour le transfert d'écriture

Dans les sessions qui utilisent le transfert d'écriture, vous ne pouvez utiliser uniquement le niveau d'isolement `REPEATABLE READ`. Bien que vous puissiez également utiliser le niveau d'isolement `READ COMMITTED` avec des réplicas Aurora, ce niveau d'isolement ne fonctionne pas avec le transfert d'écriture. Pour plus d'informations sur les niveaux d'isolement `REPEATABLE READ` et `READ COMMITTED`, consultez Niveaux d'isolement Aurora MySQL.

Cohérence de lecture pour le transfert d’écriture

Vous pouvez contrôler le degré de cohérence de lecture sur un cluster de bases de données. Le niveau de cohérence de lecture détermine le temps que le cluster de bases de données doit attendre avant chaque opération de lecture, afin de s’assurer que certaines ou toutes les modifications sont répliquées à partir de l’enregistreur. Vous pouvez ajuster le niveau de cohérence de lecture pour vous assurer que toutes les opérations d’écriture transférées de votre session sont visibles dans le cluster de bases de données avant toute requête ultérieure. Vous pouvez également utiliser ce paramètre pour vous assurer que les requêtes sur le cluster de bases de données voient toujours les mises à jour les plus récentes de l’enregistreur. Ce paramètre s’applique également aux requêtes soumises par d’autres sessions ou d’autres clusters. Pour spécifier ce type de comportement pour votre application, choisissez une valeur pour le paramètre de base de données ou le paramètre de cluster de bases de données `aurora_replica_read_consistency`.

> **Important**
>
> Définissez toujours le paramètre de base de données ou le paramètre de cluster de bases de données `aurora_replica_read_consistency` lorsque vous souhaitez transférer des écritures. Si ce n’est pas le cas, Aurora ne transfère pas les écritures. Ce paramètre a une valeur vide par défaut, alors choisissez une valeur spécifique lorsque vous utilisez ce paramètre. Le paramètre `aurora_replica_read_consistency` affecte uniquement les clusters de bases de données et les instances pour lesquels le transfert d’écriture est activé.

Plus vous augmentez le niveau de cohérence, plus votre application passe de temps à attendre que les modifications se propagent entre les instances de base de données. Avant l’exécution de vos requêtes, vous pouvez choisir l’équilibre entre un temps de réponse rapide et l’assurance que les modifications apportées à d’autres instances de base de données seront entièrement disponibles.

Vous pouvez spécifier les valeurs suivantes pour le paramètre `aurora_replica_read_consistency` :

- `EVENTUAL` : les résultats des opérations d’écriture dans la même session ne sont pas visibles tant que l’opération d’écriture n’est pas effectuée sur l’instance de base de données d’enregistreur. La requête n’attend pas que les résultats mis à jour soient disponibles. Ainsi, elle peut récupérer les données plus anciennes ou les données mises à jour, en fonction de l’heure des instructions et de la durée du retard de réplication. Il s’agit de la même cohérence que pour les clusters de bases de données Aurora MySQL qui n’utilisent pas le transfert d’écriture.

- `SESSION` : toutes les requêtes qui utilisent le transfert d'écriture voient les résultats de toutes les modifications apportées au cours de cette session. Les modifications sont visibles que la transaction soit validée ou non. Si nécessaire, la requête attend que les résultats des opérations d'écriture transférées soient répliqués.
- `GLOBAL` : une session affiche toutes les modifications validées dans toutes les sessions et instances du cluster de bases de données. Chaque requête peut attendre pendant une période qui varie en fonction du décalage de la session. La requête se poursuit lorsque le cluster de bases de données est à jour avec toutes les données validées de l'enregistreur, à compter du début de la requête.

Pour obtenir des informations sur les paramètres de configuration impliqués dans le transfert d'écriture, consultez Paramètres de configuration pour le transfert d'écriture.

> **Note**
>
> Vous pouvez également utiliser `aurora_replica_read_consistency` en tant que variable de session, par exemple :
>
> ```
> mysql> set aurora_replica_read_consistency = 'session';
> ```

Exemples d'utilisation du transfert d'écriture

Les exemples suivants montrent les effets du paramètre `aurora_replica_read_consistency` sur l'exécution d'instructions `INSERT` suivies d'instructions `SELECT`. Les résultats peuvent différer en fonction de la valeur de `aurora_replica_read_consistency` et de l'heure des instructions.

Pour obtenir une plus grande cohérence, vous pouvez attendre brièvement avant d'émettre l'instruction `SELECT`. Sinon, Aurora peut attendre automatiquement la fin de la réplication des résultats avant d'effectuer l'instruction `SELECT`.

Pour obtenir des informations sur la définition des paramètres de base de données, consultez Groupes de paramètres pour Amazon Aurora.

Example avec **`aurora_replica_read_consistency`** défini sur **`EVENTUAL`**

L'exécution d'une instruction `INSERT`, immédiatement suivie d'une instruction `SELECT`, renvoie une valeur pour `COUNT(*)` avec le nombre de lignes avant l'insertion de la nouvelle ligne. L'exécution répétée de l'instruction `SELECT` après une courte période renvoie le nombre de lignes mis à jour. Les instructions `SELECT` n'attendent pas.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> insert into t1 values (6); select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

Example avec **`aurora_replica_read_consistency`** défini sur **`SESSION`**

Une instruction `SELECT` placée immédiatement après une instruction `INSERT` attend que les modifications de l'instruction `INSERT` soient visibles. Les instructions `SELECT` suivantes n'attendent pas.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
```

```
+----------+
1 row in set (0.01 sec)

mysql> insert into t1 values (6); select count(*) from t1; select count(*) from t1;
Query OK, 1 row affected (0.08 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.37 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Le paramètre de cohérence en lecture étant toujours défini sur `SESSION`, l'introduction d'une brève attente après l'exécution d'une instruction `INSERT` rend le nombre de lignes mis à jour disponible au moment de l'exécution de l'instruction `SELECT` suivante.

```
mysql> insert into t1 values (6); select sleep(2); select count(*) from t1;
Query OK, 1 row affected (0.07 sec)
+----------+
| sleep(2) |
+----------+
|        0 |
+----------+
1 row in set (2.01 sec)
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

Example avec **`aurora_replica_read_consistency`** défini sur **`GLOBAL`**

Chaque instruction `SELECT` attend que toutes les modifications de données, depuis l'heure de début de l'instruction, soient visibles avant d'effectuer la requête. Le temps d'attente pour chaque instruction `SELECT` varie en fonction de l'ampleur du retard de réplication.

```
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.75 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.37 sec)

mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.66 sec)
```

Exécution d'instructions en plusieurs parties avec le transfert d'écriture

Une instruction DML peut être composée de plusieurs parties, notamment d'une instruction `INSERT ... SELECT` et d'une instruction `DELETE ... WHERE`. Dans ce cas, l'instruction entière est transférée vers l'instance de base de données d'enregistreur pour y être exécutée.

Transactions avec transfert d'écriture

Si le mode d'accès aux transactions est réglé sur lecture seule, le transfert d'écriture n'est pas utilisé. Vous pouvez spécifier le mode d'accès de la transaction à l'aide de l'instruction `SET TRANSACTION` ou de l'instruction `START TRANSACTION`. Vous pouvez également spécifier le mode d'accès aux transactions en modifiant la valeur de la variable de session transaction_read_only. Vous pouvez modifier cette valeur de session seulement lorsque vous êtes connecté à un cluster de bases de données pour lequel le transfert d'écriture est activé.

Si une transaction de longue durée n'émet aucune instruction pendant une longue période, elle peut dépasser le délai d'inactivité. La valeur par défaut de cette période est d'une minute. Vous pouvez

définir le paramètre `aurora_fwd_writer_idle_timeout` pour l'augmenter jusqu'à un jour. Une transaction qui dépasse le délai d'inactivité est annulée par l'instance d'enregistreur. L'instruction suivante que vous soumettez reçoit une erreur de délai d'attente. Puis Aurora restaure la transaction.

Ce type d'erreur peut se produire dans d'autres cas, lorsque le transfert d'écriture devient indisponible. Par exemple, Aurora annule toutes les transactions qui utilisent le transfert d'écriture si vous redémarrez le cluster de bases de données ou si vous désactivez le transfert d'écriture.

Lorsqu'une instance d'enregistreur dans un cluster utilisant le transfert d'écriture local est redémarrée, toutes les transactions et requêtes actives et transférées sur les instances de lecteur utilisant le transfert d'écriture local sont automatiquement fermées. Une fois que l'instance d'enregistreur est à nouveau disponible, vous pouvez réessayer ces transactions.

Paramètres de configuration pour le transfert d'écriture

Les groupes de paramètres de base de données Aurora incluent des paramètres pour la fonctionnalité de transfert d'écriture. Les détails sur ces paramètres sont résumés dans le tableau suivant, avec des notes d'utilisation après le tableau.

Paramètre	Portée	Type	Valeur par défaut	Valeurs valides
`aurora_fwd_writer_idle_timeout`	Cluster	Entier non signé	60	1–86 400
`aurora_fwd_writer_max_connections_pct`	Cluster	Entier long non signé	10	0–90
`aurora_replica_read_consistency`	Cluster ou instance	Enum	'' (null)	`EVENTUAL`, `SESSION`, `GLOBAL`

Pour contrôler les demandes d'écriture entrantes, utilisez les paramètres suivants :

- `aurora_fwd_writer_idle_timeout` : nombre de secondes pendant lesquelles l'instance de base de données d'enregistreur attend une activité sur une connexion transférée depuis une instance de lecteur avant de la fermer. Si la session reste inactive au-delà de cette période, Aurora l'annule.

- `aurora_fwd_writer_max_connections_pct` : limite supérieure des connexions de base de données qui peuvent être utilisées sur une instance de base de données d'enregistreur pour gérer les requêtes transférées à partir des instances de lecteur. Il est exprimé sous la forme d'un pourcentage du paramètre `max_connections` pour l'enregistreur. Par exemple, si la valeur `max_connections` est 800 et `aurora_fwd_master_max_connections_pct` ou `aurora_fwd_writer_max_connections_pct` 10, l'enregistreur autorise un maximum de 80 sessions transférées simultanées. Ces connexions proviennent du même groupe de connexions géré par le paramètre `max_connections`.

  Ce paramètre s'applique uniquement à l'enregistreur quand le transfert d'écriture est activé. Si vous diminuez la valeur, les connexions existantes ne sont pas affectées. Aurora prend en compte la nouvelle valeur du paramètre lors de la tentative de création d'une nouvelle connexion à partir d'un cluster de bases de données. La valeur par défaut est 10, ce qui représente 10 % de la valeur `max_connections`.

> **Note**
>
> Comme `aurora_fwd_writer_idle_timeout` et `aurora_fwd_writer_max_connections_pct` des paramètres de cluster de bases de données, toutes les instances de base de données de chaque cluster ont les mêmes valeurs pour ces paramètres.

Pour plus d'informations sur `aurora_replica_read_consistency`, consultez Cohérence de lecture pour le transfert d'écriture.

Pour plus d'informations sur les groupes de paramètres de base de données, consultez Groupes de paramètres pour Amazon Aurora.

Métriques Amazon CloudWatch et variables d'état Aurora MySQL pour le transfert d'écriture

Les métriques Amazon CloudWatch et les variables d'état Aurora MySQL suivantes s'appliquent lorsque vous utilisez le transfert d'écriture pour Aurora MySQL. Pour plus d'informations sur les métriques relatives aux instances de base de données de lecteur et d'enregistreur Aurora MySQL, consultez les rubriques suivantes.

Rubriques

Métriques relatives au transfert d’écriture pour les instances de base de données d’enregistreur Aurora MySQL

Les métriques Amazon CloudWatch suivantes s’appliquent lorsque vous utilisez le transfert d’écriture sur un ou plusieurs clusters de bases de données. Ces métriques sont toutes mesurées sur l’instance de base de données d’enregistreur.

Métrique CloudWatch	Unité	Description
`ForwardingWriterDMLLatency`	Millisecondes	Temps moyen pour traiter chaque instruction DML transférée sur l’instance de base de données d’enregistreur. Il n’inclut pas le temps nécessaire au cluster de bases de données pour transférer la demande d’écriture, ni le temps nécessaire pour répliquer les modifications et les renvoyer à l’enregistreur.
`ForwardingWriterDMLThroughput`	Nombre par seconde	Nombre d’instructions DML transférées traitées chaque seconde par cette instance de base de données d’enregistreur.
`ForwardingWriterOpenSessions`	Nombre	Nombre de sessions transférées sur l’instance de base de données d’enregistreur.

Les variables d'état Aurora MySQL suivantes s'appliquent lorsque vous utilisez le transfert d'écriture sur un ou plusieurs clusters de bases de données. Ces variables d'état sont toutes mesurées sur l'instance de base de données d'enregistreur.

Variable d'état Aurora MySQL	Unité	Description
`Aurora_fwd_writer_dml_stmt_count`	Nombre	Nombre total d'instructions DML transférées à cette instance de base de données d'enregistreur.
`Aurora_fwd_writer_dml_stmt_duration`	Microsecondes	Durée totale des instructi ons DML transférées à cette instance de base de données d'enregistreur.
`Aurora_fwd_writer_open_sessions`	Nombre	Nombre de sessions transféré es sur l'instance de base de données d'enregistreur.
`Aurora_fwd_writer_select_stmt_count`	Nombre	Nombre total d'instructions `SELECT` transférées à cette instance de base de données d'enregistreur.
`Aurora_fwd_writer_select_stmt_durati on`	Microsecondes	Durée totale des instructions `SELECT` transférées à cette instance de base de données d'enregistreur.

Métriques relatives au transfert d'écriture pour les instances de base de données de lecteur Aurora MySQL

Les métriques CloudWatch suivantes sont mesurées sur chaque instance de base de données de lecteur dans un cluster de bases de données où le transfert d'écriture est activé.

Métrique CloudWatch	Unité	Description
`ForwardingReplicaDMLLatency`	Millisecondes	Temps de réponse moyen des DML transférées sur le réplica.
`ForwardingReplicaDMLThroughput`	Nombre par seconde	Nombre d'instructions DML transférées traitées par seconde.
`ForwardingReplicaOpenSessions`	Nombre	Nombre de sessions qui utilisent le transfert d'écriture sur une instance de base de données de lecteur.
`ForwardingReplicaReadWaitLatency`	Millisecondes	Temps moyen qu'une instruction `SELECT` sur une instance de base de données de lecteur attend pour rattraper l'enregistreur. La longueur de l'attente de l'instance de base de données du lecteur avant de traiter une requête dépend du paramètre `aurora_replica_read_consistency` .
`ForwardingReplicaReadWaitThroughput`	Nombre par seconde	Nombre total d'instructions `SELECT` traitées chaque seconde dans toutes les sessions qui transportent des écritures.
`ForwardingReplicaSelectLatency`	Millisecondes	Latence d'instruction `SELECT` transférée, dont la moyenne est calculée sur toutes les instructions `SELECT`

Métrique CloudWatch	Unité	Description
		transférées au cours de la période de surveillance.
`ForwardingReplicaSelectThroughput`	Nombre par seconde	Débit d'instruction `SELECT` transférée par seconde, dont la moyenne est calculée au cours de la période de surveillance.

Les variables d'état Aurora MySQL suivantes sont mesurées sur chaque instance de base de données de lecteur dans un cluster de bases de données où le transfert d'écriture est activé.

Variable d'état Aurora MySQL	Unité	Description
`Aurora_fwd_replica_dml_stmt_count`	Nombre	Nombre total d'instructions DML transférées à partir de cette instance de base de données de lecteur.
`Aurora_fwd_replica_dml_stmt_duration`	Microsecondes	Durée totale de toutes les instructions DML transférées à partir de cette instance de base de données de lecteur.
`Aurora_fwd_replica_errors_session_limit`	Nombre	Nombre de sessions rejetées par le cluster principal en raison de l'une des conditions d'erreur suivantes : • enregistreur complet • Trop d'instructions transférées en cours
`Aurora_fwd_replica_open_sessions`	Nombre	Nombre de sessions qui utilisent le transfert d'écriture

Variable d'état Aurora MySQL	Unité	Description
		sur une instance de base de données de lecteur.
`Aurora_fwd_replica_read_wait_count`	Nombre	Nombre total d'attentes en lecture-après écriture sur cette instance de base de données de lecteur.
`Aurora_fwd_replica_read_wait_duration`	Microsecondes	Durée totale des attentes dues au paramètre de cohérence en lecture sur cette instance de base de données de lecteur.
`Aurora_fwd_replica_select_stmt_count`	Nombre	Nombre total d'instructions `SELECT` transférées à partir de cette instance de base de données de lecteur.
`Aurora_fwd_replica_select_stmt_duration`	Microsecondes	Durée totale des instructions `SELECT` transférées à partir de cette instance de base de données de lecteur.

Identification des transactions et des requêtes transférées

Vous pouvez utiliser la table `information_schema.aurora_forwarding_processlist` pour identifier les transactions et les requêtes transférées. Pour plus d'informations sur cette table, consultez information_schema.aurora_forwarding_processlist.

L'exemple suivant montre toutes les connexions transférées sur une instance de base de données d'enregistreur.

```
mysql> select * from information_schema.AURORA_FORWARDING_PROCESSLIST where
 IS_FORWARDED=1 order by REPLICA_SESSION_ID;
```

```
+-----+----------+-------------------+----------+---------+------+-------------
+-----------------------------------------+-------------+-------------------
+---------------------------------+---------------------+----------------+
| ID  | USER     | HOST              | DB       | COMMAND | TIME | STATE        |
 INFO                                    | IS_FORWARDED | REPLICA_SESSION_ID |
 REPLICA_INSTANCE_IDENTIFIER     | REPLICA_CLUSTER_NAME | REPLICA_REGION |
+-----+----------+-------------------+----------+---------+------+-------------
+-----------------------------------------+-------------+-------------------
+---------------------------------+--------------------------------------+
| 648 | myuser   | IP_address:port1   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest58 SET k=k+1 WHERE id=4802579 |            1 |                637 | my-
db-cluster-instance-2        | my-db-cluster        | us-west-2      |
| 650 | myuser   | IP_address:port2   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest54 SET k=k+1 WHERE id=2503953 |            1 |                639 | my-
db-cluster-instance-2        | my-db-cluster        | us-west-2      |
+-----+----------+-------------------+----------+---------+------+-------------
+-----------------------------------------+-------------+-------------------
+---------------------------------+---------------------+----------------+
```

Sur l'instance de base de données du lecteur de transfert, vous pouvez voir les threads associés à ces connexions de base de données d'écriture en exécutant `SHOW PROCESSLIST`. Les valeurs de `REPLICA_SESSION_ID` sur l'enregistreur, 637 et 639, sont les mêmes que les valeurs de `Id` sur le lecteur.

```
mysql> select @@aurora_server_id;

+--------------------------------+
| @@aurora_server_id             |
+--------------------------------+
| my-db-cluster-instance-2       |
+--------------------------------+
1 row in set (0.00 sec)

mysql> show processlist;

+-----+----------+-------------------+----------+---------+------+-------------
+-----------------------------------------+
| Id  | User     | Host              | db       | Command | Time | State        | Info
                                    |
+-----+----------+-------------------+----------+---------+------+-------------
+-----------------------------------------+
| 637 | myuser   | IP_address:port1   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest12 SET k=k+1 WHERE id=4802579  |
```

```
| 639 | myuser   | IP_address:port2   | sysbench | Query   |    0 | async commit |
 UPDATE sbtest61 SET k=k+1 WHERE id=2503953  |
+-----+----------+--------------------+----------+---------+------+--------------
+-------------------------------------------+
12 rows in set (0.00 sec)
```

Intégration d'Amazon Aurora MySQL avec d'autres services AWS

Amazon Aurora MySQL s'intègre avec d'autres services AWS pour vous permettre d'étendre votre cluster de bases de données Aurora MySQL afin d'utiliser des fonctionnalités supplémentaires dans le cloud AWS. Votre cluster de bases de données Aurora MySQL peut utiliser des services AWS pour effectuer les opérations suivantes :

- Appeler de façon synchrone ou asynchrone une fonction AWS Lambda à l'aide des fonctions natives `lambda_sync` ou `lambda_async`. Pour plus d'informations, consultez Appel d'une fonction Lambda à partir d'un cluster de bases de données Amazon Aurora MySQL.
- Charger dans votre cluster de bases de données les données de fichiers texte ou XML stockés dans un compartiment Amazon Simple Storage Service (Amazon S3) à l'aide de la commande `LOAD DATA FROM S3` ou `LOAD XML FROM S3`. Pour plus d'informations, consultez Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3.
- Enregistrer des données dans des fichiers texte stockés dans un compartiment Amazon S3 à partir de votre cluster de bases de données à l'aide de la commande `SELECT INTO OUTFILE S3`. Pour plus d'informations, consultez Enregistrement de données d'un cluster de bases de données Amazon Aurora MySQL dans des fichiers texte stockés dans un compartiment Amazon S3.
- Ajouter ou supprimer automatiquement des réplicas Aurora avec Application Auto Scaling. Pour plus d'informations, consultez Amazon Aurora Auto Scaling avec des réplicas Aurora.
- Exécutez l'analyse du sentiment avec Amazon Comprehend ou une grande diversité d'algorithmes de machine learning avec SageMaker AI. Pour plus d'informations, consultez Utilisation du machine learning Amazon Aurora.

Aurora sécurise l'accès aux autres services AWS en utilisant Gestion des identités et des accès AWS (IAM). Vous accordez l'autorisation d'accès aux autres services AWS en créant un rôle IAM disposant des autorisations nécessaires, puis en associant ce rôle à votre cluster de bases de données. Pour obtenir des informations et des instructions sur la manière d'autoriser votre cluster de bases de données Aurora MySQL à accéder à d'autres services AWS en votre nom, consultez Autorisation d'Amazon Aurora MySQL à accéder à d'autres services AWS en votre nom.

Autorisation d'Amazon Aurora MySQL à accéder à d'autres services AWS en votre nom

Pour permettre à votre cluster de base de données Aurora MySQL d'accéder à d'autres services en votre nom, vous devez créer et configurer un rôle Gestion des identités et des accès AWS (IAM). Ce rôle autorise les utilisateurs de bases de données dans votre cluster de base de données à accéder aux autres services AWS. Pour plus d'informations, consultez Configuration de rôles IAM pour accéder aux services AWS.

Vous devez également configurer votre cluster de base de données Aurora pour autoriser les connexions sortantes au service AWS cible. Pour plus d'informations, consultez Activation de la communication réseau entre Amazon Aurora et d'autres services AWS.

De cette façon, les utilisateurs de base de données peuvent effectuer les actions suivantes à l'aide des autres services AWS :

- Appeler de façon synchrone ou asynchrone une fonction AWS Lambda à l'aide des fonctions natives `lambda_sync` ou `lambda_async`. Ou appeler de façon asynchrone une fonction AWS Lambda en utilisant la procédure `mysql.lambda_async`. Pour plus d'informations, consultez Appel d'une fonction Lambda avec une fonction native Aurora MySQL.
- Charger dans votre cluster de bases de données les données de fichiers texte ou XML stockés dans un compartiment Amazon S3 à l'aide de l'instruction `LOAD DATA FROM S3` ou `LOAD XML FROM S3`. Pour plus d'informations, consultez Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3.
- Enregistrer des données de votre cluster de bases de données dans des fichiers texte stockés dans un compartiment Amazon S3 à l'aide de l'instruction `SELECT INTO OUTFILE S3`. Pour plus d'informations, consultez Enregistrement de données d'un cluster de bases de données Amazon Aurora MySQL dans des fichiers texte stockés dans un compartiment Amazon S3.
- Exportez des données de journaux vers Amazon CloudWatch Logs MySQL. Pour plus d'informations, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs.
- Ajouter ou supprimer automatiquement des réplicas Aurora avec Application Auto Scaling. Pour de plus amples informations, consultez Amazon Aurora Auto Scaling avec des réplicas Aurora.

Configuration de rôles IAM pour accéder aux services AWS

Pour autoriser votre cluster de base de données Aurora à accéder à un autre service AWS, procédez comme suit :

1. Créez une stratégie IAM qui accorde l'autorisation nécessaire au service AWS. Pour plus d'informations, consultez les rubriques suivantes.
   - Création d'une stratégie IAM pour accéder aux ressources Amazon S3
   - Création d'une politique IAM pour accéder aux ressources AWS Lambda
   - Création d'une politique IAM pour accéder aux ressources CloudWatch des journaux
   - Création d'une politique IAM pour accéder aux ressources AWS KMS
2. Créez un rôle IAM et attachez-lui la stratégie que vous avez créée. Pour plus d'informations, consultez Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS.
3. Associez ce rôle IAM à votre cluster de base de données Aurora. Pour de plus amples informations, consultez Association d'un rôle IAM à un cluster de bases de données Amazon Aurora MySQL.

Création d'une stratégie IAM pour accéder aux ressources Amazon S3

Aurora peut accéder aux ressources Amazon S3 pour charger des données ou enregistrer des données à partir d'un cluster de bases de données Aurora. Toutefois, vous devez créer au préalable une stratégie IAM pour fournir les autorisations de compartiment et d'objet permettant à Aurora d'accéder à Amazon S3.

Le tableau ci-dessous répertorie les fonctions Aurora qui peuvent accéder à un compartiment Amazon S3 en votre nom, ainsi que les autorisations de compartiment et d'objet minimales requises pour chaque fonction.

Fonctionnalité	Permissions de compartiment	Autorisations d'objet
`LOAD DATA FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`
`LOAD XML FROM S3`	`ListBucket`	`GetObject` `GetObjectVersion`

Fonctionnalité	Permissions de compartiment	Autorisations d'objet
`SELECT INTO OUTFILE S3`	`ListBucket`	`AbortMultipartUpload` `DeleteObject` `GetObject` `ListMultipartUploadParts` `PutObject`

La stratégie suivante ajoute les autorisations pouvant être requises par Aurora pour accéder à un compartiment Amazon S3 en votre nom.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAuroraToExampleBucket",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:AbortMultipartUpload",
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetObjectVersion",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

> **Note**
>
> Veillez à inclure les deux entrées pour la valeur `Resource`. Aurora a besoin des autorisations sur le compartiment lui-même et sur tous les objets à l'intérieur du compartiment.
> En fonction de votre cas d'utilisation, vous n'avez peut-être pas besoin d'ajouter toutes les autorisations dans l'exemple de stratégie. Aussi, d'autres autorisations peuvent être requises. Par exemple, si votre compartiment Amazon S3 est chiffré, vous devez ajouter les autorisations `kms:Decrypt`.

Vous pouvez effectuer les étapes ci-dessous pour créer une stratégie IAM qui fournit les autorisations minimales nécessaires à Aurora pour accéder à un compartiment Amazon S3 en votre nom. Pour autoriser Aurora à accéder à tous vos compartiments Amazon S3, vous pouvez ignorer ces étapes et utiliser la stratégie IAM prédéfinie `AmazonS3ReadOnlyAccess` ou `AmazonS3FullAccess` au lieu d'en créer une.

Pour créer une stratégie IAM accordant l'accès à vos ressources Amazon S3

1. Ouvrez IAM Management Console.
2. Dans le panneau de navigation, choisissez Politiques.
3. Choisissez Create Policy (Créer une politique).
4. Sous l'onglet Visual editor (Éditeur visuel), choisissez Choose a service (Choisir un service), puis S3.
5. Sous Actions, choisissez Expand all (Développer tout), puis choisissez les autorisations de compartiment et d'objet nécessaires à la stratégie IAM.

   Les autorisations d'objet sont des autorisations pour les opérations d'objet dans Amazon S3. Elles doivent être accordées pour les objets d'un compartiment et non pour le compartiment lui-même. Pour plus d'informations sur les autorisations liées aux opérations d'objet dans Amazon S3, consultez Autorisations pour des opérations d'objet.
6. Choisissez Resources (Ressources), puis Add ARN (Ajouter un ARN) pour bucket (compartiment).
7. Dans la boîte de dialogue Add ARN(s) (Ajouter un ou des ARN), fournissez les détails sur votre ressource, puis choisissez Add (Ajouter).

   Spécifiez le compartiment Amazon S3 auquel autoriser l'accès. Par exemple, si vous voulez autoriser Aurora à accéder au compartiment Amazon S3 nommé *amzn-s3-demo-bucket*,

définissez la valeur de l'Amazon Resource Name (ARN) sur `arn:aws:s3:::amzn-s3-demo-bucket`.

8. Si la ressource object (objet) est répertoriée, choisissez Add ARN (Ajouter un ARN) pour object (objet).
9. Dans la boîte de dialogue Add ARN(s) (Ajouter un ou des ARN), fournissez les détails sur votre ressource.

   Pour le compartiment Amazon S3, spécifiez le compartiment Amazon S3 auquel autoriser l'accès. Pour l'objet, vous pouvez choisir Any (Tous) pour accorder des autorisations à tous les objets du compartiment.

   > **Note**
   >
   > Vous pouvez affecter à Amazon Resource Name (ARN) une valeur d'ARN plus spécifique afin d'autoriser Aurora à accéder uniquement à des fichiers ou des dossiers spécifiques dans un compartiment Amazon S3. Pour plus d'informations sur la définition d'une stratégie d'accès pour Amazon S3, consultez Gestion des autorisations d'accès de vos ressources Amazon S3.

10. (Facultatif) Choisissez Add ARN (Ajouter un ARN) pour bucket (compartiment) pour ajouter un autre compartiment Amazon S3 à la stratégie, puis répétez les étapes précédentes pour le compartiment.

    > **Note**
    >
    > Vous pouvez répéter ces étapes pour ajouter les instructions d'autorisation de compartiment correspondantes à votre stratégie pour chaque compartiment Amazon S3 auquel Aurora doit accéder. Si vous le souhaitez, vous pouvez également accorder l'accès à tous les compartiments et à tous les objets dans Amazon S3.

11. Choisissez Examiner une stratégie.
12. Dans Name (Name), attribuez un nom à votre stratégie IAM, par exemple `AllowAuroraToExampleBucket`. Vous utilisez ce nom lorsque vous créez un rôle IAM à associer à votre cluster de bases de données Aurora. Vous pouvez également ajouter une valeur Description facultative.
13. Choisissez Créer une stratégie.

14. Suivez les étapes de Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS.

Création d'une politique IAM pour accéder aux ressources AWS Lambda

Vous pouvez créer une politique IAM qui fournit les autorisations minimales requises pour qu'Aurora appelle une AWS Lambda fonction en votre nom.

La stratégie suivante ajoute les autorisations qu'Aurora requiert pour appeler une fonction AWS Lambda en votre nom.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAuroraToExampleFunction",
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "arn:aws:lambda:us-
east-1:123456789012:function:example_function"
    }
  ]
}
```

Vous pouvez suivre les étapes suivantes pour créer une politique IAM qui fournit les autorisations minimales requises pour qu'Aurora appelle une AWS Lambda fonction en votre nom. Pour permettre à Aurora d'invoquer toutes vos AWS Lambda fonctions, vous pouvez ignorer ces étapes et utiliser la `AWSLambdaRole` politique prédéfinie au lieu de créer la vôtre.

Pour créer une stratégie IAM accordant l'autorisation d'appeler vos fonctions AWS Lambda

1. Ouvrez la console IAM.
2. Dans le panneau de navigation, choisissez Politiques.
3. Choisissez Create Policy (Créer une politique).

4. Sous l'onglet Visual editor (Éditeur visuel), choisissez Choose a service (Choisir un service), puis Lambda.
5. Sous Actions, choisissez Expand all (Développer tout), puis choisissez les autorisations AWS Lambda nécessaires à la stratégie IAM.

   Assurez-vous que `InvokeFunction` est sélectionné. Il s'agit de l'autorisation minimale requise pour permettre à Amazon Aurora d'appeler une AWS Lambda fonction.
6. Choisissez Resources (Ressources), puis Add ARN (Ajouter un ARN) pour function (fonction).
7. Dans la boîte de dialogue Add ARN(s) (Ajouter un ou des ARN), fournissez les détails sur votre ressource.

   Spécifiez la fonction Lambda auquel autoriser l'accès. Par exemple, si vous voulez autoriser Aurora à accéder à une fonction Lambda nommée `example_function`, attribuez à l'ARN la valeur `arn:aws:lambda:::function:example_function`.

   Pour plus d'informations sur la définition d'une politique d'accès pourAWS Lambda, voir Authentification et contrôle d'accès pour AWS Lambda.
8. Choisissez éventuellement Ajouter des autorisations supplémentaires pour ajouter une autre AWS Lambda fonction à la politique, puis répétez les étapes précédentes pour la fonction.

   > **Note**
   >
   > Vous pouvez répéter cette opération pour ajouter les instructions d'autorisation de fonction correspondantes à votre politique pour chaque AWS Lambda fonction à laquelle vous souhaitez qu'Aurora accède.
9. Choisissez Examiner une politique.
10. Dans Name (Name), attribuez un nom à votre stratégie IAM, par exemple `AllowAuroraToExampleFunction`. Vous utilisez ce nom lorsque vous créez un rôle IAM à associer à votre cluster de base de données Aurora. Vous pouvez également ajouter une valeur Description facultative.
11. Choisissez Créer une stratégie.
12. Suivez les étapes de Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS.

Création d'une politique IAM pour accéder aux ressources CloudWatch des journaux

Aurora peut accéder aux CloudWatch journaux pour exporter les données des journaux d'audit à partir d'un cluster de base de données Aurora. Cependant, vous devez d'abord créer une politique IAM qui fournit les autorisations de groupe de journaux et de flux de journaux qui permettent à Aurora d'accéder aux CloudWatch journaux.

La politique suivante ajoute les autorisations requises par Aurora pour accéder à Amazon CloudWatch Logs en votre nom, ainsi que les autorisations minimales requises pour créer des groupes de journaux et exporter des données.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableCreationAndManagementOfRDSCloudwatchLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/rds/*:log-stream:*"
        },
        {
            "Sid":
 "EnableCreationAndManagementOfRDSCloudwatchLogGroupsAndStreams",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutRetentionPolicy",
                "logs:CreateLogGroup"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/rds/*"
        }
    ]
}
```

Vous pouvez modifier le contenu ARNs de la politique pour restreindre l'accès à une AWS région et à un compte spécifiques.

Vous pouvez suivre les étapes suivantes pour créer une politique IAM qui fournit les autorisations minimales requises pour qu'Aurora accède aux CloudWatch journaux en votre nom. Pour autoriser Aurora à accéder pleinement aux CloudWatch journaux, vous pouvez ignorer ces étapes et utiliser la politique IAM `CloudWatchLogsFullAccess` prédéfinie au lieu de créer la vôtre. Pour plus d'informations, consultez la section Utilisation de politiques basées sur l'identité (politiques IAM) pour les CloudWatch journaux dans le guide de l'utilisateur Amazon CloudWatch .

Pour créer une politique IAM afin d'accorder l'accès à vos ressources CloudWatch Logs

1. Ouvrez la console IAM.
2. Dans le panneau de navigation, choisissez Politiques.
3. Choisissez Create Policy (Créer une politique).
4. Dans l'onglet Éditeur visuel, choisissez Choisir un service, puis CloudWatchLogs.
5. Pour Actions, choisissez Expand all (sur la droite), puis choisissez les autorisations Amazon CloudWatch Logs nécessaires pour la politique IAM.

   Assurez-vous que les autorisations suivantes sont sélectionnées :

   - `CreateLogGroup`
   - `CreateLogStream`
   - `DescribeLogStreams`
   - `GetLogEvents`
   - `PutLogEvents`
   - `PutRetentionPolicy`
6. Choisissez Resources (Ressources), puis Add ARN (Ajouter un ARN) pour log-group.
7. Dans la boîte de dialogue Add ARN(s) (Ajouter un ou des ARN), entrez les valeurs suivantes :

   - Région — Une AWS région ou *
   - Account (Compte) – Numéro de compte ou *
   - Nom du groupe de journaux – `/aws/rds/*`
8. Dans la boîte de dialogue Add ARN(s) (Ajouter un ou des ARN), choisissez Ajouter.
9. Choisissez Add ARN (Ajouter un ARN) pour log-stream.

10. Dans la boîte de dialogue Add ARN(s) (Ajouter un ou des ARN), entrez les valeurs suivantes :
    - Région — Une AWS région ou *
    - Account (Compte) – Numéro de compte ou *
    - Nom du groupe de journaux – `/aws/rds/*`
    - Log Stream Name (Nom du flux de journaux – *
11. Dans la boîte de dialogue Add ARN(s) (Ajouter un ou des ARN), choisissez Ajouter.
12. Choisissez Examiner une stratégie.
13. Dans Name (Name), attribuez un nom à votre stratégie IAM, par exemple `AmazonRDSCloudWatchLogs`. Vous utilisez ce nom lorsque vous créez un rôle IAM à associer à votre cluster de base de données Aurora. Vous pouvez également ajouter une valeur Description facultative.
14. Choisissez Créer une stratégie.
15. Suivez les étapes de Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS.

Création d'une politique IAM pour accéder aux ressources AWS KMS

Aurora peut accéder à la AWS KMS keys utilisée pour chiffrer ses sauvegardes de base de données. Toutefois, vous devez créer au préalable une stratégie IAM pour fournir les autorisations permettant à Aurora d'accéder aux clés KMS.

La stratégie suivante ajoute les autorisations requises à Aurora pour accéder aux clés KMS en votre nom.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAuroraToAccessKey",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
```

```
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-ID"
    }
  ]
}
```

Vous pouvez utiliser les étapes suivantes pour créer une stratégie IAM qui fournit les autorisations minimales requises pour qu'Aurora accède aux clés KMS en votre nom.

Pour créer une stratégie IAM accordant l'accès à vos clés KMS

1. Ouvrez la console IAM.
2. Dans le panneau de navigation, choisissez Politiques.
3. Choisissez Create Policy (Créer une politique).
4. Sous l'onglet Visual Editor (Éditeur visuel), choisissez Choose a service (Choisir un service), puis KMS.
5. Pour Actions, choisissez Write (Écrire), puis choisissez Decrypt (Déchiffrer).
6. Choisissez Resources (Ressources), puis Add ARN (Ajouter un ARN).
7. Dans la boîte de dialogue Add ARN(s) (Ajouter un ou des ARN), entrez les valeurs suivantes :
   - Région : saisissez la AWS région, par exemple`us-west-2`.
   - Account (Compte) – Saisissez le numéro du compte utilisateur.
   - Nom du flux de journalisation – Tapez l'identifiant de clé KMS.
8. Dans la boîte de dialogue Add ARN(s) (Ajouter un ou des ARN), choisissez Ajouter.
9. Choisissez Examiner une stratégie.
10. Dans Name (Name), attribuez un nom à votre stratégie IAM, par exemple `AmazonRDSKMSKey`. Vous utilisez ce nom lorsque vous créez un rôle IAM à associer à votre cluster de base de données Aurora. Vous pouvez également ajouter une valeur Description facultative.
11. Choisissez Créer une stratégie.
12. Suivez les étapes de Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS.

Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS

Après avoir créé une stratégie IAM pour autoriser Aurora à accéder aux ressources AWS, vous devez créer un rôle IAM et attacher la stratégie IAM à ce nouveau rôle IAM.

Pour créer un rôle IAM afin d'autoriser votre cluster Amazon RDS à communiquer avec d'autres services AWS en votre nom, procédez comme suit.

Pour créer un rôle IAM afin d'autoriser Amazon RDS à accéder aux services AWS

1. Ouvrez la console IAM.
2. Dans le panneau de navigation, choisissez Roles (Rôles).
3. Sélectionnez Create role (Créer un rôle).
4. Sous Service AWS, sélectionnez RDS.
5. Pour Select your use case (Sélectionner votre cas d'utilisation), choisissez RDS – Add Role to Database (Ajouter un rôle à la base de données).
6. Choisissez Suivant.
7. Sur la page Permissions policies (Politiques d'autorisations), saisissez le nom de votre politique dans le champ Search (Rechercher).
8. Quand elle s'affiche dans la liste, sélectionnez la stratégie définie précédemment à l'aide des instructions de l'une des sections suivantes :
   - Création d’une stratégie IAM pour accéder aux ressources Amazon S3
   - Création d'une politique IAM pour accéder aux ressources AWS Lambda
   - Création d'une politique IAM pour accéder aux ressources CloudWatch des journaux
   - Création d'une politique IAM pour accéder aux ressources AWS KMS
9. Choisissez Suivant.
10. Pour Role name (Nom du rôle), indiquez le nom de votre rôle IAM, par exemple `RDSLoadFromS3`. Vous pouvez également ajouter une valeur Description facultative.
11. Choisissez Create Role (Créer un rôle).
12. Suivez les étapes de Association d'un rôle IAM à un cluster de bases de données Amazon Aurora MySQL.

Association d'un rôle IAM à un cluster de bases de données Amazon Aurora MySQL

Pour autoriser les utilisateurs d'une base de données dans un cluster de bases de données Amazon Aurora à accéder aux autres services AWS, vous devez associer le rôle IAM que vous avez créé dans Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS à ce cluster de bases de données. Il est également possible qu'AWS crée un nouveau rôle IAM en associant directement le service.

> **Note**
>
> Vous ne pouvez pas associer un rôle IAM à un cluster de base de données Aurora Serverless v1. Pour de plus amples informations, consultez Utilisation d'Amazon Aurora Serverless v1. Vous pouvez associer un rôle IAM à un cluster de bases de données Aurora Serverless v2.

Pour associer un rôle IAM à un cluster de base de données, vous devez effectuer les deux opérations suivantes :

1. Ajoutez le rôle à la liste des rôles associés à un cluster DB en utilisant la console RDS, la commande add-role-to-db-cluster de l'interface AWS CLI ou l'opération AddRoleToDBCluster de l'API RDS.

   Vous pouvez ajouter cinq rôles IAM au maximum pour chaque cluster de base de données Aurora.
2. Définissez l'ARN du rôle IAM associé comme valeur du paramètre de niveau cluster du services AWS concerné.

   Le tableau ci-dessous décrit les noms des paramètres de niveau cluster pour les rôles IAM utilisés pour accéder aux autres servicesAWS.

Paramètre de niveau cluster	Description
`aws_defaul lt_lambda_role`	Utilisé lors de l'appel d'une fonction Lambda à partir de votre cluster de base de données.
`aws_defaul lt_logs_role`	Ce paramètre n'est plus nécessaire pour exporter les données de journaux de votre cluster de base de données vers Amazon CloudWatch Logs. Aurora MySQL utilise

Paramètre de niveau cluster	Description	
	à présent un rôle lié à un service pour les autorisations requises. Pour plus d'informations sur les rôles liés à un service, consultez Utilisation des rôles liés à un service pour Amazon Aurora.	
`aws_default_s3_role`	Utilisé lors de l'appel de l'instruction `LOAD DATA FROM S3`, `LOAD XML FROM S3` ou `SELECT INTO OUTFILE S3` à partir de votre cluster de base de données. Dans Aurora MySQL version 2, le rôle IAM spécifié dans ce paramètre est utilisé si un rôle IAM n'est pas spécifié pour `aurora_load_from_s3_role` ou `aurora_select_into_s3_role` pour l'instruction appropriée. Dans Aurora MySQL version 3, le rôle IAM spécifié pour ce paramètre est toujours utilisé.	
`aurora_load_from_s3_role`	Utilisé lors de l'appel de l'instruction `LOAD DATA FROM S3` ou `LOAD XML FROM S3` à partir de votre cluster de bases de données. Si un rôle IAM n'est pas spécifié pour ce paramètre, le rôle IAM spécifié dans `aws_default_s3_role` est utilisé. Dans Aurora MySQL version 3, ce paramètre n'est pas disponible.	

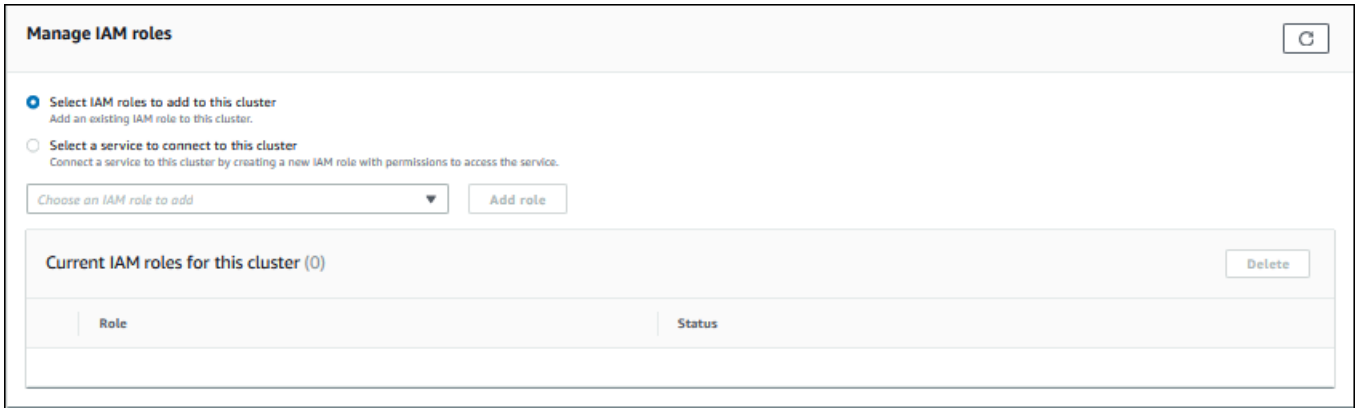
Paramètre de niveau cluster	Description	
`aurora_select_into_s3_role`	Utilisé lors de l'appel de l'instruction `SELECT INTO OUTFILE S3` à partir de votre cluster de bases de données. Si un rôle IAM n'est pas spécifié pour ce paramètre, le rôle IAM spécifié dans `aws_default_s3_role` est utilisé. Dans Aurora MySQL version 3, ce paramètre n'est pas disponible.	

Pour associer un rôle IAM afin d'autoriser votre cluster Amazon RDS à communiquer avec d'autres services AWS en votre nom, procédez comme suit.

console

Pour associer un rôle IAM à un cluster de base de données Aurora à l'aide de la console

1. Ouvrez la console RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez le nom du cluster de base de données Aurora auquel associer un rôle IAM afin d'en afficher les détails.
4. Dans l'onglet Connectivity & security (Connectivité et sécurité), dans la section Manage IAM roles (Gérer les rôles IAM), effectuez l'une des opérations suivantes :
   - Select IAM roles to add to this cluster (Sélectionnez les rôles IAM à ajouter à ce cluster) (par défaut)
   - Select a service to connect to this cluster (Sélectionner un service à connecter à ce cluster)



5. Pour utiliser un rôle IAM existant, sélectionnez-le dans le menu, puis choisissez Add role (Ajouter un rôle).

   Si l'ajout du rôle est réussi, son statut s'affiche alors comme `Pending`, puis `Available`.

6. Pour connecter directement un service :

   a. Choisissez Select a service to connect to this cluster (Sélectionner un service à connecter à ce cluster).

   b. Choisissez le service dans le menu, puis choisissez Connect service (Connecter un service).

   c. Pour (Connect cluster to *`Service Name`*) Connecter le cluster au Nom du service, saisissez l'Amazon Resource Name (ARN) à utiliser pour se connecter au service, puis choisissez Connect service (Connecter un service).

   AWS crée un nouveau rôle IAM pour se connecter au service. Son statut affiche `Pending`, puis `Available`.

7. (Facultatif) Pour arrêter l'association d'un rôle IAM à un cluster de bases de données et supprimer l'autorisation correspondante, choisissez le rôle, puis Delete (Supprimer).

Définir le paramètre de niveau cluster pour le rôle IAM associé

1. Dans la console RDS, choisissez Groupes de paramètres dans le panneau de navigation.
2. Si vous utilisez déjà un groupe de paramètres de base de données personnalisé, vous pouvez sélectionner ce groupe afin de l'utiliser au lieu de créer un groupe de paramètres de cluster de base de données. Si vous utilisez le groupe de paramètres de cluster de base de données par

défaut, créez un nouveau groupe de paramètres de cluster de base de données, comme décrit dans les étapes suivantes :

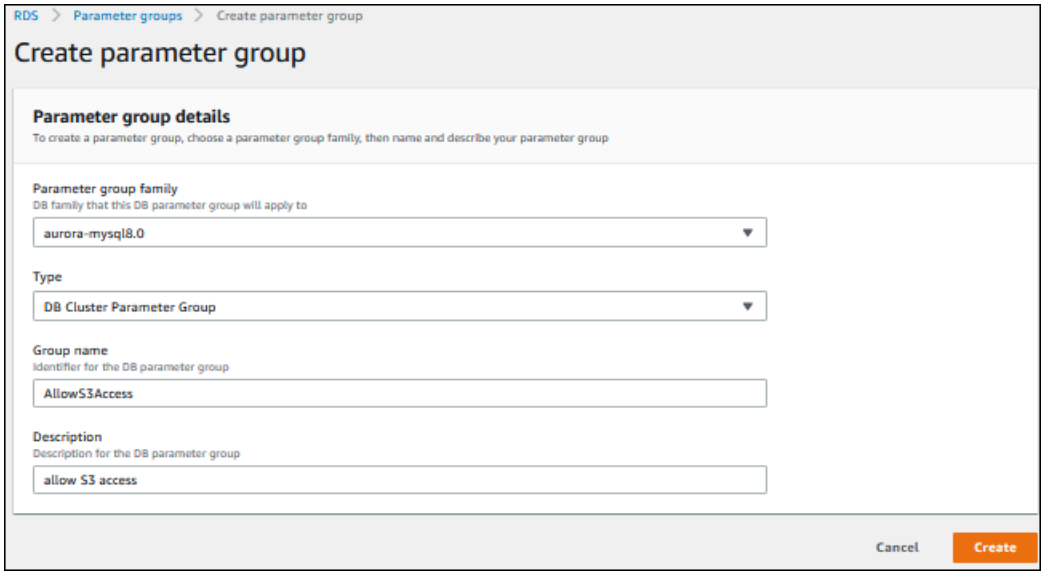
a. Choisissez Créer un groupe de paramètres.

b. Pour Famille de groupes de paramètres, choisissez `aurora-mysql8.0` pour un cluster de base de données compatible avec Aurora MySQL 8.0 ou `aurora-mysql5.7` pour un cluster de base de données compatible avec Aurora MySQL 5.7.

c. Pour Type, choisissez Groupe de paramètres de cluster DB.

d. Pour Nom du groupe, entrez le nom de votre nouveau groupe de paramètres de cluster de bases de données.

e. Dans le champ Description, saisissez une description du nouveau groupe de paramètres de cluster de bases de données.



f. Sélectionnez Créer.

3. Sur la page Groupes de paramètres, sélectionnez votre groupe de paramètres de cluster de bases de données, puis choisissez Modifier pour Parameter group actions (Actions de groupe de paramètres).

4. Affectez aux paramètres appropriés de niveau cluster les valeurs d'ARN des rôles IAM associés.

   Par exemple, vous pouvez affecter au paramètre `aws_default_s3_role` la valeur `arn:aws:iam::123456789012:role/AllowS3Access`.

5. Sélectionnez Save Changes.

6. Pour modifier le groupe de paramètres de cluster DB pour votre cluster DB, effectuez les étapes suivantes :

a. Choisissez Databases (Bases de données), puis sélectionnez votre cluster DB Aurora.

b. Sélectionnez Modify.

c. Faites défiler l'écran jusqu'à Options de base de données et définissez Groupe de paramètres de cluster DB en spécifiant le groupe de paramètres de cluster DB.

d. Choisissez Continuer.

e. Vérifiez vos modifications, puis sélectionnez Appliquer immédiatement.

f. Choisissez Modifier le cluster.

g. Choisissez Databases (Bases de données), puis sélectionnez l'instance principale de votre cluster DB.

h. Pour Actions, choisissez Redémarrer.

   Une fois que l'instance a redémarré, votre rôle IAM est associé à votre cluster de base de données.

   Pour plus d'informations sur les groupes de paramètres de cluster, consultez Paramètres de configuration d'Aurora MySQL.

INTERFACE DE LIGNE DE COMMANDE (CLI)

Pour associer un rôle IAM à un cluster de base de données à l'aide de l AWS CLI

1. Appelez la commande `add-role-to-db-cluster` à partir de l'AWS CLI pour ajouter les ARN de vos rôles IAM au cluster de bases de données, comme ci-dessous.

```
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-
arn arn:aws:iam::123456789012:role/AllowAuroraS3Role
PROMPT> aws rds add-role-to-db-cluster --db-cluster-identifier my-cluster --role-
arn arn:aws:iam::123456789012:role/AllowAuroraLambdaRole
```

2. Si vous utilisez le groupe de paramètres de cluster de base de données par défaut, créez un nouveau groupe de paramètres de cluster de base de données. Si vous utilisez déjà un groupe de paramètres de base de données personnalisé, vous pouvez utiliser ce groupe au lieu de créer un groupe de paramètres de cluster de base de données.

   Pour créer un groupe de paramètres de cluster de bases de données, appelez la commande `create-db-cluster-parameter-group` à partir de l'AWS CLI, comme ci-dessous.

```
PROMPT> aws rds create-db-cluster-parameter-group  --db-cluster-parameter-group-
name AllowAWSAccess \
     --db-parameter-group-family aurora5.7 --description "Allow access to Amazon S3
 and AWS Lambda"
```

Pour un cluster de bases de données compatible avec Aurora MySQL 5.7, spécifiez `aurora-mysql5.7` pour `--db-parameter-group-family`. Pour un cluster de bases de données compatible avec Aurora MySQL 8.0, spécifiez `aurora-mysql8.0` pour `--db-parameter-group-family`.

3. Définissez les paramètres appropriés de niveau cluster et les valeurs d'ARN des rôles IAM associés dans votre groupe de paramètres de cluster de base de données, comme illustré ci-après.

```
PROMPT> aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name
 AllowAWSAccess \
    --parameters
 "ParameterName=aws_default_s3_role,ParameterValue=arn:aws:iam::123456789012:role/
AllowAuroraS3Role,method=pending-reboot" \
    --parameters
 "ParameterName=aws_default_lambda_role,ParameterValue=arn:aws:iam::123456789012:role/
AllowAuroraLambdaRole,method=pending-reboot"
```

4. Modifiez le cluster de base de données afin d'utiliser le nouveau groupe de paramètres de cluster de base de données, puis redémarrez le cluster, comme illustré ci-après.

```
PROMPT> aws rds modify-db-cluster --db-cluster-identifier my-cluster --db-cluster-
parameter-group-name AllowAWSAccess
PROMPT> aws rds reboot-db-instance --db-instance-identifier my-cluster-primary
```

Une fois que l'instance a redémarré, vos rôles IAM sont associés à votre cluster de base de données.

Pour plus d'informations sur les groupes de paramètres de cluster, consultez Paramètres de configuration d’Aurora MySQL.

Activation de la communication réseau entre Amazon Aurora et d'autres services AWS

Pour utiliser certains autres services AWS avec Amazon Aurora, la configuration réseau de votre cluster de bases de données Aurora doit autoriser les connexions sortantes vers des points de terminaison pour ces services. Les opérations suivantes exigent cette configuration réseau.

- Appel de fonctions AWS Lambda Pour plus d'informations sur cette fonctionnalité, consultez Appel d'une fonction Lambda avec une fonction native Aurora MySQL.
- Accès aux fichiers à partir de Amazon S3. Pour plus d'informations sur cette fonctionnalité, consultez Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3 et Enregistrement de données d'un cluster de bases de données Amazon Aurora MySQL dans des fichiers texte stockés dans un compartiment Amazon S3.
- Accès aux points de terminaison AWS KMS. L'accès à AWS KMS est requis pour utiliser des flux d'activité de base de données avec Aurora MySQL. Pour plus d'informations sur cette fonctionnalité, consultez Surveillance d'Amazon Aurora à l'aide des flux d'activité de base de données.
- Accès aux points de terminaison SageMaker AI. L'accès à SageMaker AI est requis pour utiliser le machine learning SageMaker AI avec Aurora MySQL. Pour plus d'informations sur cette fonctionnalité, consultez Utilisation du machine learning Amazon Aurora avec Aurora MySQL.

Aurora renvoie les messages d'erreur suivants s'il ne peut pas se connecter au point de terminaison d'un service.

```
ERROR 1871 (HY000): S3 API returned error: Network Connection
```

```
ERROR 1873 (HY000): Lambda API returned error: Network Connection. Unable to connect to
 endpoint
```

```
ERROR 1815 (HY000): Internal error: Unable to initialize S3Stream
```

Pour les flux d'activité de base de données utilisant Aurora MySQL, le flux d'activité cesse de fonctionner si le cluster de bases de données ne peut pas accéder au point de terminaison AWS KMS. Aurora vous informe de ce problème à l'aide d'événements RDS.

Si vous rencontrez ces messages en utilisant les services AWS correspondants, vérifiez si votre cluster de bases de données Aurora est public ou privé. Si votre cluster de bases de données Aurora est privé, vous devez le configurer pour activer les connexions.

Pour qu'un cluster de bases de données Aurora soit public, il doit être marqué comme accessible publiquement. Auquel cas, l'option Accessible publiquement indique Oui dans les détails du cluster de bases de données affichés dans AWS Management Console. Le cluster de bases de données doit également se trouver dans un sous-réseau public Amazon VPC. Pour plus d'informations sur les instances de base de données accessibles publiquement, consultez Utilisation d'un cluster de bases de données dans un VPC. Pour plus d'informations sur les sous-réseaux Amazon VPC publics, consultez VPC et sous-réseaux.

Si votre cluster de bases de données Aurora n'est pas accessible publiquement et ne se trouve pas dans un sous-réseau public VPC, il est privé. Il se peut que vous ayez un cluster de bases de données privé et que vous souhaitiez utiliser l'une des fonctionnalités qui nécessitent cette configuration réseau. Dans ce cas, configurez le cluster de sorte qu'il puisse se connecter à ces adresses Internet via NAT (Network Address Translation). Comme alternative à Amazon S3, Amazon SageMaker AI et AWS Lambda, vous pouvez configurer le VPC afin qu'il ait un point de terminaison de VPC pour l'autre service associé à la table de routage du cluster de bases de données. Consultez Utilisation d'un cluster de bases de données dans un VPC. Pour plus d'informations sur la configuration de NAT dans votre VPC, consultez Passerelle NAT. Pour plus d'informations sur la configuration des points de terminaison d'un VPC, consultez Points de terminaison d'un VPC. Vous pouvez également créer un point de terminaison de passerelle S3 pour accéder à votre compartiment S3. Pour plus d'informations, consultez Points de terminaison de passerelle pour Amazon S3.

Vous devrez peut-être également ouvrir les ports éphémères de vos listes de contrôle d'accès (ACL) réseau dans les règles sortantes de votre groupe de sécurité VPC. Pour plus d'informations sur les ports éphémères pour les listes ACL réseau, consultez Ports éphémères dans le Guide de l'utilisateur Amazon Virtual Private Cloud.

Rubriques en relation

- Intégration d'Aurora à d'autres AWS services
- Gestion d'un cluster de bases de données Amazon Aurora

Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3

Vous pouvez utiliser l'instruction `LOAD DATA FROM S3` ou `LOAD XML FROM S3` pour charger les données de fichiers stockés dans un compartiment Amazon S3. Dans Aurora MySQL, les fichiers sont d'abord stockés sur le disque local, puis importés dans la base de données. Une fois les importations dans la base de données effectuées, les fichiers locaux sont supprimés.

> **Note**
>
> Le chargement de données dans une table à partir de fichiers texte n'est pas pris en charge pour Aurora Serverless v1. Elle est prise en charge pour Aurora Serverless v2.

Table des matières

Octroi à Aurora d'un accès à Amazon S3

Pour pouvoir charger des données à partir d'un compartiment Amazon S3, vous devez d'abord octroyer à votre cluster de bases de données Aurora MySQL une autorisation d'accès à Amazon S3.

Pour octroyer à Aurora MySQL un accès à Amazon S3

1. Créez une stratégie Gestion des identités et des accès AWS (IAM) pour fournir les autorisations de compartiment et d'objet permettant à votre cluster de bases de données Aurora MySQL d'accéder à Amazon S3. Pour obtenir des instructions, consultez Création d'une stratégie IAM pour accéder aux ressources Amazon S3.

   > **Note**
   >
   > Dans Aurora MySQL 3.05 et versions ultérieures, vous pouvez charger des objets chiffrés à l'aide de AWS KMS keys gérées par le client. Pour ce faire, incluez l'autorisation `kms:Decrypt` dans votre politique IAM. Pour plus d'informations, consultez Création d'une politique IAM pour accéder aux ressources AWS KMS. Vous n'avez pas besoin de cette autorisation pour charger des objets chiffrés à l'aide de Clés gérées par AWS ou de clés gérées par Amazon S3 (SSE-S3).

2. Créez un rôle IAM et attachez la politique IAM que vous avez créée dans Création d'une stratégie IAM pour accéder aux ressources Amazon S3 au nouveau rôle IAM. Pour obtenir des instructions, consultez Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS.
3. Assurez-vous que le cluster de bases de données utilise un groupe de paramètres de cluster de bases de données personnalisé.

   Pour plus d'informations sur la création d'un groupe de paramètres de cluster de bases de données personnalisé, consultez Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora.
4. Pour Aurora MySQL version 2, définissez le paramètre de cluster de bases de données `aurora_load_from_s3_role` ou `aws_default_s3_role` en spécifiant l'Amazon Resource Name (ARN) du nouveau rôle IAM. Si un rôle IAM n'est pas spécifié pour `aurora_load_from_s3_role`, Aurora utilise le rôle IAM spécifié dans `aws_default_s3_role`.

   Pour Aurora MySQL version 3, utilisez `aws_default_s3_role`.

   Si le cluster fait partie d'une base de données globale Aurora, définissez ce paramètre pour chaque cluster Aurora de cette base de données. Bien que seul le cluster principal d'une base de données globale Aurora puisse charger des données, un autre cluster peut être promu en tant que cluster principal par le mécanisme de basculement.

Pour plus d'informations sur les paramètres de cluster de bases de données, consultez Paramètres de cluster de bases de données et d'instance de base de données Amazon Aurora.

5. Pour autoriser les utilisateurs de base de données d'un cluster de bases de données Aurora MySQL à accéder à Amazon S3, associez le rôle que vous avez créé dans Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS au cluster de bases de données. Pour une base de données globale Aurora, associez le rôle à chaque cluster Aurora de cette base de données. Pour plus d'informations sur l'association d'un rôle IAM à un cluster de bases de données, consultez Association d'un rôle IAM à un cluster de bases de données Amazon Aurora MySQL.
6. Configurez votre cluster de bases de données Aurora MySQL de façon à autoriser les connexions sortantes vers Amazon S3. Pour obtenir des instructions, consultez Activation de la communication réseau entre Amazon Aurora et d'autres services AWS.

   Si votre cluster de bases de données n'est pas accessible publiquement et ne se trouve pas dans un sous-réseau public VPC, il est privé. Vous pouvez créer un point de terminaison de passerelle S3 pour accéder à votre compartiment S3. Pour plus d'informations, consultez Points de terminaison de passerelle pour Amazon S3.

   Pour une base de données globale Aurora, activez les connexions sortantes pour chaque cluster Aurora de cette base de données.

Accord de privilèges permettant de charger des données dans Amazon Aurora MySQL

L'utilisateur de base de données qui émet l'instruction `LOAD DATA FROM S3` ou `LOAD XML FROM S3` doit avoir un rôle ou un privilège spécifique pour l'émettre. Dans Aurora MySQL version 3, vous accordez le rôle `AWS_LOAD_S3_ACCESS`. Dans Aurora MySQL version 2, vous accordez le privilège `LOAD FROM S3`. L'utilisateur administratif d'un cluster de bases de données reçoit le rôle ou le privilège approprié par défaut. Vous pouvez accorder le privilège à un autre utilisateur à l'aide des instructions suivantes.

Utilisez l'instruction suivante pour Aurora MySQL version 3 :

```
GRANT AWS_LOAD_S3_ACCESS TO 'user'@'domain-or-ip-address'
```

> **Tip**
>
> Lorsque vous utilisez la technique de rôle dans Aurora MySQL version 3, vous pouvez également activer le rôle en utilisant l'instruction `SET ROLE role_name` ou `SET ROLE ALL`. Si vous n'êtes pas familier avec le système de rôles MySQL 8.0, vous pouvez en apprendre davantage dans Modèle de privilège basé sur les rôles. Pour plus de détails, consultez Using roles dans le manuel de référence de MySQL.
>
> Cela s'applique uniquement à la session active en cours. Lorsque vous vous reconnectez, vous devez à nouveau exécuter l'instruction `SET ROLE` pour accorder des privilèges. Pour plus d'informations, consultez SET ROLE statement dans le manuel MySQL Reference Manual.
>
> Vous pouvez utiliser le paramètre `activate_all_roles_on_login` de cluster de bases de données pour activer automatiquement tous les rôles lorsqu'un utilisateur se connecte à une instance de base de données. Lorsque ce paramètre est défini, vous n'avez pas besoin d'appeler explicitement l'instruction `SET ROLE` pour activer un rôle. Pour plus d'informations, consultez activate_all_roles_on_login dans le manuel MySQL Reference Manual.
>
> Toutefois, vous devez appeler `SET ROLE ALL` explicitement au début d'une procédure stockée pour activer le rôle, lorsque cette procédure est appelée par un autre utilisateur.

Utilisez l'instruction suivante pour Aurora MySQL version 2 :

```
GRANT LOAD FROM S3 ON *.* TO 'user'@'domain-or-ip-address'
```

Le rôle `AWS_LOAD_S3_ACCESS` et le privilège `LOAD FROM S3` sont propres à Amazon Aurora et ne sont pas disponibles pour les bases de données MySQL externes ni les instances de base de données RDS for MySQL. Si vous avez configuré la réplication entre un cluster de bases de données Aurora faisant office de source de réplication et une base de données MySQL faisant office de client de réplication, l'instruction `GRANT` pour le rôle ou le privilège entraîne l'arrêt de la réplication avec une erreur. Vous pouvez ignorer sans risque l'erreur afin de reprendre la réplication. Pour ignorer l'erreur sur une instance de base de données RDS for MySQL, utilisez la procédure mysql_rds_skip_repl_error. Pour ignorer l'erreur sur une base de données MySQL externe, utilisez la variable système slave_skip_errors (Aurora MySQL version 2) ou la variable système replica_skip_errors (Aurora MySQL version 3).

> **Note**
>
> L'utilisateur de la base de données doit disposer de privilèges `INSERT` pour la base de données dans laquelle il charge les données.

Spécification du chemin (URI) à un compartiment Amazon S3

La syntaxe permettant de spécifier le chemin (URI) aux fichiers stockés dans un compartiment Amazon S3 est la suivante.

```
s3-region://amzn-s3-demo-bucket/file-name-or-prefix
```

Le chemin d'accès inclut les valeurs suivantes :

- `region` (facultatif) – Région AWS qui contient le compartiment Amazon S3 à partir duquel effectuer le chargement. Ce champ est facultatif. Si vous ne spécifiez pas de valeur `region`, Aurora charge votre fichier à partir d'Amazon S3 dans la même région que votre cluster de bases de données.
- `bucket-name` – Nom du compartiment Amazon S3 qui contient les données à charger. Les préfixes d'objet qui identifient un chemin d'accès du dossier virtuel sont pris en charge.
- `file-name-or-prefix` – Nom du fichier XML ou du fichier texte Amazon S3, ou préfixe qui identifie un ou plusieurs fichiers XML ou texte à charger. Vous pouvez également spécifier un fichier manifeste qui identifie un ou plusieurs fichiers texte à charger. Pour plus d'informations sur l'utilisation d'un fichier manifeste pour charger des fichiers texte à partir d'Amazon S3, consultez Utilisation d'un manifeste pour spécifier les fichiers de données à charger.

Pour copier l'URI des fichiers dans un compartiment S3

1. Connectez-vous à AWS Management Console et ouvrez la console Amazon S3 à l'adresse https://console.aws.amazon.com/s3/.
2. Dans le panneau de navigation, choisissez Compartiments, puis choisissez le compartiment dont vous souhaitez copier l'URI.
3. Sélectionnez le préfixe ou le fichier que vous souhaitez charger depuis S3.
4. Choisissez Copier l'URI S3.

LOAD DATA FROM S3

Vous pouvez utiliser l'instruction `LOAD DATA FROM S3` pour charger des données à partir de n'importe quel format de fichier texte pris en charge par l'instruction MySQL LOAD DATA INFILE, tel que des données texte séparées par des virgules. Les fichiers compressés ne sont pas pris en charge.

> **Note**
>
> Assurez-vous que votre cluster de bases de données Aurora MySQL autorise les connexions sortantes vers S3. Pour plus d'informations, consultez Activation de la communication réseau entre Amazon Aurora et d'autres services AWS.

Syntaxe

```
LOAD DATA [FROM] S3 [FILE | PREFIX | MANIFEST] 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number {LINES | ROWS}]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

> **Note**
>
> Dans Aurora MySQL 3.05 et versions ultérieures, le mot-clé `FROM` est facultatif.

Paramètres

L'instruction `LOAD DATA FROM S3` utilise les paramètres obligatoires et facultatifs suivants. Pour plus d'informations sur certains de ces paramètres, consultez Instruction LOAD DATA dans la documentation sur MySQL.

FILE | PREFIX | MANIFEST

Indique si les données sont chargées à partir d'un seul fichier, à partir de tous les fichiers qui correspondent à un préfixe donné ou à partir de tous les fichiers contenus dans un manifeste spécifié. `FILE` est la valeur par défaut.

S3-URI

Spécifie l'URI pour un fichier texte ou manifeste à charger, ou un préfixe Amazon S3 à utiliser. Spécifiez l'URI grâce à la syntaxe décrite dans Spécification du chemin (URI) à un compartiment Amazon S3.

REPLACE | IGNORE

Détermine l'action à effectuer si une ligne d'entrée a les mêmes valeurs de clé uniques qu'une ligne existante dans la table de base de données.

- Spécifiez `REPLACE` si vous souhaitez que la ligne d'entrée remplace la ligne existante dans la table.
- Spécifiez `IGNORE` si vous souhaitez supprimer la ligne d'entrée.

INTO TABLE

Identifie le nom de la table de base de données dans laquelle charger les lignes d'entrée.

PARTITION

Exige que toutes les lignes d'entrée soient insérées dans les partitions identifiées par la liste spécifiée de noms de partition séparés par des virgules. Si une ligne d'entrée ne peut pas être insérée dans l'une des partitions spécifiées, l'instruction échoue et une erreur est renvoyée.

CHARACTER SET

Identifie le jeu de caractères des données du fichier d'entrée.

FIELDS | COLUMNS

Identifie la façon dont les champs ou les colonnes sont délimités dans le fichier d'entrée. Par défaut, les champs sont délimités par des tabulations.

LINES

Identifie la façon dont les lignes sont délimitées dans le fichier d'entrée. Les lignes sont délimitées par un caractère de saut de ligne (`'\n'`) par défaut.

IGNORE ***nombre*** LINES | ROWS

Indique qu'un certain nombre de lignes au début du fichier d'entrée doivent être ignorées. Par exemple, vous pouvez utiliser `IGNORE 1 LINES` pour laisser de côté une ligne d'en-tête initiale contenant les noms de colonnes ou `IGNORE 2 ROWS` les deux premières lignes de données dans le fichier d'entrée. Si vous utilisez également `PREFIX`, `IGNORE` ignore un certain nombre de lignes au début du premier fichier d'entrée.

col_name_or_user_var, ...

Spécifie la liste séparée par des virgules d'un ou plusieurs noms de colonne ou de variables utilisateur qui identifient les colonnes à charger par nom. Le nom d'une variable utilisateur utilisé à cette fin doit correspondre au nom d'un élément dans le fichier texte, préfixé par @. Vous pouvez utiliser les variables utilisateur pour stocker les valeurs de champ correspondantes pour une réutilisation ultérieure.

Par exemple, l'instruction suivante charge la première colonne du fichier d'entrée dans la première colonne de `table1` et affecte à la colonne `table_column2` dans `table1` la valeur d'entrée de la deuxième colonne divisée par 100.

```
LOAD DATA FROM S3 's3://amzn-s3-demo-bucket/data.txt'
    INTO TABLE table1
    (column1, @var1)
    SET table_column2 = @var1/100;
```

SET

Spécifie la liste séparée par des virgules des opérations d'affectation qui attribuent aux colonnes de la table des valeurs non incluses dans le fichier d'entrée.

Par exemple, l'instruction suivante affecte aux deux premières colonnes de `table1` les valeurs figurant dans les deux premières colonnes du fichier d'entrée, puis affecte à la colonne `column3` dans `table1` la valeur d'horodatage actuelle.

```
LOAD DATA FROM S3  's3://amzn-s3-demo-bucket/data.txt'
    INTO TABLE table1
    (column1, column2)
```

```
    SET column3 = CURRENT_TIMESTAMP;
```

Vous pouvez utiliser des sous-requêtes dans la partie droite des affectations `SET`. Pour une sous-requête qui renvoie une valeur devant être assignée à une colonne, vous pouvez utiliser uniquement une sous-requête scalaire. En outre, vous ne pouvez pas utiliser une sous-requête pour sélectionner dans la table qui est en cours de chargement.

Vous ne pouvez pas utiliser le mot-clé `LOCAL` de l'instruction `LOAD DATA FROM S3` si vous chargez des données à partir d'un compartiment Amazon S3.

Utilisation d'un manifeste pour spécifier les fichiers de données à charger

Vous pouvez utiliser l'instruction `LOAD DATA FROM S3` avec le mot-clé `MANIFEST` pour spécifier un fichier manifeste au format JSON qui répertorie les fichiers texte à charger dans une table de votre cluster de bases de données.

Le schéma JSON suivant décrit le format et le contenu d'un fichier manifeste.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "additionalProperties": false,
    "definitions": {},
    "id": "Aurora_LoadFromS3_Manifest",
    "properties": {
        "entries": {
            "additionalItems": false,
            "id": "/properties/entries",
            "items": {
                "additionalProperties": false,
                "id": "/properties/entries/items",
                "properties": {
                    "mandatory": {
                        "default": "false",
                        "id": "/properties/entries/items/properties/mandatory",
                        "type": "boolean"
                    },
                    "url": {
                        "id": "/properties/entries/items/properties/url",
                        "maxLength": 1024,
                        "minLength": 1,
                        "type": "string"
                    }
```

```
                },
                "required": [
                    "url"
                ],
                "type": "object"
            },
            "type": "array",
            "uniqueItems": true
        }
    },
    "required": [
        "entries"
    ],
    "type": "object"
}
```

Chaque `url` du manifeste doit spécifier une URL avec le nom de compartiment et le chemin d'objet complet du fichier, pas simplement un préfixe. Vous pouvez utiliser un manifeste pour charger les fichiers de différents compartiments, différentes régions ou les fichiers qui ne partagent pas le même préfixe. Si une région n'est pas spécifiée dans l'URL, la région du cluster de bases de données Aurora cible est utilisée. L'exemple suivant illustre un fichier manifeste qui charge quatre fichiers à partir de trois compartiments différents.

```
{
  "entries": [
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":true
    },
    {
      "url":"s3-us-west-2://aurora-bucket-usw2/2013-10-05-customerdata",
      "mandatory":true
    },
    {
      "url":"s3://aurora-bucket/2013-10-04-customerdata",
      "mandatory":false
    },
    {
      "url":"s3://aurora-bucket/2013-10-05-customerdata"
    }
  ]
}
```

L'indicateur facultatif `mandatory` spécifie si `LOAD DATA FROM S3` doit renvoyer une erreur si le fichier est introuvable. L'indicateur `mandatory` est défini par défaut sur `false`. Quels que soient les paramètres de `mandatory`, `LOAD DATA FROM S3` s'arrête si aucun fichier n'est trouvé.

Les fichiers manifestes peuvent avoir n'importe quelle extension. L'exemple suivant exécute l'instruction `LOAD DATA FROM S3` avec le manifeste de l'exemple précédent, qui s'appelle **`customer.manifest`**.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://aurora-bucket/customer.manifest'
    INTO TABLE CUSTOMER
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL);
```

Une fois l'instruction terminée, une entrée pour chaque fichier chargé avec succès est écrite dans la table `aurora_s3_load_history`.

Vérification des fichiers chargés grâce à la table aurora_s3_load_history

Chaque instruction `LOAD DATA FROM S3` réussie met à jour la table `aurora_s3_load_history` dans le schéma `mysql` avec une entrée pour chaque fichier chargé.

Après avoir exécuté l'instruction `LOAD DATA FROM S3`, vous pouvez vérifier quels fichiers ont été chargés en interrogeant la table `aurora_s3_load_history`. Pour voir les fichiers qui ont été chargés à partir d'une seule itération de l'instruction, utilisez la clause `WHERE` pour filtrer les enregistrements sur l'URI Amazon S3 du fichier manifeste utilisé dans l'instruction. Si vous avez utilisé le même fichier manifeste auparavant, filtrez les résultats grâce au champ `timestamp`.

```
select * from mysql.aurora_s3_load_history where load_prefix = 'S3_URI';
```

La table suivante décrit les champs dans la table `aurora_s3_load_history`.

Champ	Description
`load_prefix`	L'URI spécifié dans l'instruction load. Cet URI peut correspondre à l'un des éléments suivants : • Un fichier de données unique pour une déclaration `LOAD DATA FROM S3 FILE`

Champ	Description
	• Un préfixe Amazon S3 qui correspond à plusieurs fichiers de données pour une déclaration `LOAD DATA FROM S3 PREFIX` • Un fichier manifeste unique qui contient les noms des fichiers à charger pour une déclaration `LOAD DATA FROM S3 MANIFEST`
`file_name`	Le nom d'un fichier qui a été chargé dans Aurora à partir d'Amazon S3 en utilisant l'URI identifiée dans le champ `load_prefix` .
`version_number`	Le numéro de version du fichier identifié par le champ `file_name` qui a été chargé, si le compartiment Amazon S3 possède un numéro de version.
`bytes_loaded`	La taille du fichier chargé en octets.
`load_timestamp`	La valeur d'horodatage lorsque l'instruction `LOAD DATA FROM S3` a fini de s'exécuter.

Exemples

L'instruction suivante charge les données d'un compartiment Amazon S3 situé dans la même région que le cluster de bases de données Aurora. L'instruction lit les données séparées par des virgules du fichier `customerdata.txt` qui se trouve dans le compartiment *`amzn-s3-demo-bucket`* d'Amazon S3, puis charge les données dans la table `store-schema.customer-table`.

```
LOAD DATA FROM S3 's3://amzn-s3-demo-bucket/customerdata.csv'
    INTO TABLE store-schema.customer-table
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, ADDRESS, EMAIL, PHONE);
```

L'instruction suivante charge les données d'un compartiment Amazon S3 situé dans une région différente du cluster de bases de données Aurora. L'instruction lit les données séparées par des virgules de tous les fichiers qui correspondent au préfixe d'objet `employee-data` dans le

compartiment *amzn-s3-demo-bucket* d'Amazon S3 de la région us-west-2, puis charge les données dans la table employees.

```
LOAD DATA FROM S3 PREFIX 's3-us-west-2://amzn-s3-demo-bucket/employee_data'
    INTO TABLE employees
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (ID, FIRSTNAME, LASTNAME, EMAIL, SALARY);
```

L'instruction suivante charge des données à partir des fichiers spécifiés dans un fichier manifeste JSON nommé q1_sales.json dans la table sales.

```
LOAD DATA FROM S3 MANIFEST 's3-us-west-2://amzn-s3-demo-bucket1/q1_sales.json'
    INTO TABLE sales
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    (MONTH, STORE, GROSS, NET);
```

LOAD XML FROM S3

Vous pouvez utiliser l'instruction LOAD XML FROM S3 pour charger les données de fichiers XML stockés dans un compartiment Amazon S3 dans l'un des trois formats XML :

- Les noms de colonnes en tant qu'attributs d'un élément <row>. La valeur d'attribut identifie le contenu du champ de table.

  ```
  <row column1="value1" column2="value2" .../>
  ```

- Les noms de colonnes en tant qu'éléments enfants d'un élément <row>. La valeur de l'élément enfant identifie le contenu du champ de table.

  ```
  <row>
    <column1>value1</column1>
    <column2>value2</column2>
  </row>
  ```

- Les noms de colonnes dans l'attribut name des éléments <field> dans un élément <row>. La valeur de l'élément <field> identifie le contenu du champ de table.

  ```
  <row>
  ```

```
  <field name='column1'>value1</field>
  <field name='column2'>value2</field>
</row>
```

Syntaxe

```
LOAD XML FROM S3 'S3-URI'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [PARTITION (partition_name,...)]
    [CHARACTER SET charset_name]
    [ROWS IDENTIFIED BY '<element-name>']
    [IGNORE number {LINES | ROWS}]
    [(field_name_or_user_var,...)]
    [SET col_name = expr,...]
```

Paramètres

L'instruction `LOAD XML FROM S3` utilise les paramètres obligatoires et facultatifs suivants. Pour plus d'informations sur certains de ces paramètres, consultez Instruction LOAD XML dans la documentation sur MySQL.

FILE | PREFIX

Indique si les données doivent être chargées à partir d'un seul fichier ou à partir de tous les fichiers qui correspondent à un préfixe donné. `FILE` est la valeur par défaut.

REPLACE | IGNORE

Détermine l'action à effectuer si une ligne d'entrée a les mêmes valeurs de clé uniques qu'une ligne existante dans la table de base de données.

- Spécifiez `REPLACE` si vous souhaitez que la ligne d'entrée remplace la ligne existante dans la table.
- Spécifiez `IGNORE` si vous voulez ignorer la ligne d'entrée. `IGNORE` est la valeur par défaut.

INTO TABLE

Identifie le nom de la table de base de données dans laquelle charger les lignes d'entrée.

PARTITION

Exige que toutes les lignes d'entrée soient insérées dans les partitions identifiées par la liste spécifiée de noms de partition séparés par des virgules. Si une ligne d'entrée ne peut pas être insérée dans l'une des partitions spécifiées, l'instruction échoue et une erreur est renvoyée.

CHARACTER SET

Identifie le jeu de caractères des données du fichier d'entrée.

ROWS IDENTIFIED BY

Identifie le nom d'élément qui identifie une ligne dans le fichier d'entrée. La valeur par défaut est `<row>`.

IGNORE ***nombre*** LINES | ROWS

Indique qu'un certain nombre de lignes au début du fichier d'entrée doivent être ignorées. Par exemple, vous pouvez utiliser `IGNORE 1 LINES` pour laisser de côté la première ligne dans le fichier texte, ou `IGNORE 2 ROWS` pour laisser de côté les deux premières lignes de données dans le fichier XML d'entrée.

field_name_or_user_var, ...

Spécifie la liste séparée par des virgules d'un ou plusieurs noms d'élément XML ou de variables utilisateur qui identifient les éléments à charger par nom. Le nom d'une variable utilisateur utilisé à cette fin doit correspondre au nom d'un élément dans le fichier XML, préfixé par @. Vous pouvez utiliser les variables utilisateur pour stocker les valeurs de champ correspondantes pour une réutilisation ultérieure.

Par exemple, l'instruction suivante charge la première colonne du fichier d'entrée dans la première colonne de `table1` et affecte à la colonne `table_column2` dans `table1` la valeur d'entrée de la deuxième colonne divisée par 100.

```
LOAD XML FROM S3 's3://amzn-s3-demo-bucket/data.xml'
   INTO TABLE table1
   (column1, @var1)
   SET table_column2 = @var1/100;
```

SET

Spécifie la liste séparée par des virgules des opérations d'affectation qui attribuent aux colonnes de la table des valeurs non incluses dans le fichier d'entrée.

Par exemple, l'instruction suivante affecte aux deux premières colonnes de `table1` les valeurs figurant dans les deux premières colonnes du fichier d'entrée, puis affecte à la colonne `column3` dans `table1` la valeur d'horodatage actuelle.

```
LOAD XML FROM S3 's3://amzn-s3-demo-bucket/data.xml'
   INTO TABLE table1
   (column1, column2)
   SET column3 = CURRENT_TIMESTAMP;
```

Vous pouvez utiliser des sous-requêtes dans la partie droite des affectations `SET`. Pour une sous-requête qui renvoie une valeur devant être assignée à une colonne, vous pouvez utiliser uniquement une sous-requête scalaire. De plus, vous ne pouvez pas utiliser une sous-requête pour effectuer une sélection dans la table qui est en cours de chargement.

Enregistrement de données d'un cluster de bases de données Amazon Aurora MySQL dans des fichiers texte stockés dans un compartiment Amazon S3

Vous pouvez utiliser l'instruction `SELECT INTO OUTFILE S3` pour interroger les données d'un cluster de bases de données Amazon Aurora MySQL et les enregistrer dans des fichiers texte stockés dans un compartiment Amazon S3. Dans Aurora MySQL, les fichiers sont d'abord stockés sur le disque local, puis exportés vers S3. Une fois les exportations terminées, les fichiers locaux sont supprimés.

Vous pouvez chiffrer le compartiment Amazon S3 à l'aide d'une clé gérée par Amazon S3 (SSE-S3) ou d'une AWS KMS key (SSE-KMS : Clé gérée par AWS ou clé gérée par le client).

L'instruction `LOAD DATA FROM S3` peut utiliser les fichiers créés par l'instruction `SELECT INTO OUTFILE S3` pour charger les données dans un cluster de bases de données Aurora. Pour plus d'informations, consultez Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3.

> **Note**
>
> Cette fonction n'est pas prise en charge pour les clusters de bases de données Aurora Serverless v1. Elle est prise en charge pour les clusters de bases de données Aurora Serverless v2.

Vous pouvez enregistrer les données du cluster de bases de données et de l'instantané du cluster de bases de données sur Amazon S3 à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API Amazon RDS. Pour plus d'informations, consultez Exportation des données du cluster de bases de données vers Amazon S3 et Exportation de données d'instantanés de cluster de bases de données vers Amazon S3.

Table des matières

Octroi à Aurora MySQL d'un accès à Amazon S3

Pour pouvoir enregistrer des données dans un compartiment Amazon S3, vous devez d'abord octroyer à votre cluster de bases de données Aurora MySQL une autorisation d'accès à Amazon S3.

Pour octroyer à Aurora MySQL un accès à Amazon S3

1. Créez une stratégie Gestion des identités et des accès AWS (IAM) pour fournir les autorisations de compartiment et d'objet permettant à votre cluster de bases de données Aurora MySQL d'accéder à Amazon S3. Pour obtenir des instructions, consultez Création d'une stratégie IAM pour accéder aux ressources Amazon S3.

   > **Note**
   >
   > Dans Aurora MySQL 3.05 et versions ultérieures, vous pouvez chiffrer des objets à l'aide de clés AWS KMS gérées par le client. Pour ce faire, incluez l'autorisation `kms:GenerateDataKey` dans votre politique IAM. Pour plus d'informations, consultez Création d'une politique IAM pour accéder aux ressources AWS KMS.

Vous n’avez pas besoin de cette autorisation pour chiffrer des objets à l’aide de clés Clés gérées par AWS ou de clés gérées par Amazon S3 (SSE-S3).

2. Créez un rôle IAM et attachez la politique IAM que vous avez créée dans Création d’une stratégie IAM pour accéder aux ressources Amazon S3 au nouveau rôle IAM. Pour obtenir des instructions, consultez Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS.
3. Pour Aurora MySQL version 2, définissez le paramètre de cluster de bases de données `aurora_select_into_s3_role` ou `aws_default_s3_role` en spécifiant l’Amazon Resource Name (ARN) du nouveau rôle IAM. Si un rôle IAM n’est pas spécifié pour `aurora_select_into_s3_role`, Aurora utilise le rôle IAM spécifié dans `aws_default_s3_role`.

   Pour Aurora MySQL version 3, utilisez `aws_default_s3_role`.

   Si le cluster fait partie d’une base de données globale Aurora, définissez ce paramètre pour chaque cluster Aurora de cette base de données.

   Pour plus d’informations sur les paramètres de cluster de bases de données, consultez Paramètres de cluster de bases de données et d’instance de base de données Amazon Aurora.
4. Pour autoriser les utilisateurs de base de données d’un cluster de bases de données Aurora MySQL à accéder à Amazon S3, associez le rôle que vous avez créé dans Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS au cluster de bases de données.

   Pour une base de données globale Aurora, associez le rôle à chaque cluster Aurora de cette base de données.

   Pour plus d’informations sur l’association d’un rôle IAM à un cluster de bases de données, consultez Association d'un rôle IAM à un cluster de bases de données Amazon Aurora MySQL.
5. Configurez votre cluster de bases de données Aurora MySQL de façon à autoriser les connexions sortantes vers Amazon S3. Pour obtenir des instructions, consultez Activation de la communication réseau entre Amazon Aurora et d’autres services AWS.

   Pour une base de données globale Aurora, activez les connexions sortantes pour chaque cluster Aurora de cette base de données.

Accord de privilèges permettant d'enregistrer des données dans Aurora MySQL

L'utilisateur de base de données qui émet l'instruction `SELECT INTO OUTFILE S3` doit avoir un rôle ou un privilège spécifique. Dans Aurora MySQL version 3, vous accordez le rôle `AWS_SELECT_S3_ACCESS`. Dans Aurora MySQL version 2, vous accordez le privilège `SELECT INTO S3`. L'utilisateur administratif d'un cluster de bases de données reçoit le rôle ou le privilège approprié par défaut. Vous pouvez accorder le privilège à un autre utilisateur à l'aide des instructions suivantes.

Utilisez l'instruction suivante pour Aurora MySQL version 3 :

```
GRANT AWS_SELECT_S3_ACCESS TO 'user'@'domain-or-ip-address'
```

> **Tip**
>
> Lorsque vous utilisez la technique de rôle dans Aurora MySQL version 3, vous pouvez également activer le rôle en utilisant l'instruction `SET ROLE role_name` ou `SET ROLE ALL`. Si vous n'êtes pas familier avec le système de rôles MySQL 8.0, vous pouvez en apprendre davantage dans Modèle de privilège basé sur les rôles. Pour plus de détails, consultez Using roles dans le manuel de référence de MySQL.
>
> Cela s'applique uniquement à la session active en cours. Lorsque vous vous reconnectez, vous devez à nouveau exécuter l'instruction `SET ROLE` pour accorder des privilèges. Pour plus d'informations, consultez SET ROLE statement dans le manuel MySQL Reference Manual.
>
> Vous pouvez utiliser le paramètre `activate_all_roles_on_login` de cluster de bases de données pour activer automatiquement tous les rôles lorsqu'un utilisateur se connecte à une instance de base de données. Lorsque ce paramètre est défini, vous n'avez pas besoin d'appeler explicitement l'instruction `SET ROLE` pour activer un rôle. Pour plus d'informations, consultez activate_all_roles_on_login dans le manuel MySQL Reference Manual.
>
> Toutefois, vous devez appeler `SET ROLE ALL` explicitement au début d'une procédure stockée pour activer le rôle, lorsque cette procédure est appelée par un autre utilisateur.

Utilisez l'instruction suivante pour Aurora MySQL version 2 :

```
GRANT SELECT INTO S3 ON *.* TO 'user'@'domain-or-ip-address'
```

Le rôle `AWS_SELECT_S3_ACCESS` et le privilège `SELECT INTO S3` sont propres à Amazon Aurora MySQL et ne sont pas disponibles pour les bases de données MySQL ou les instances de base de données RDS for MySQL. Si vous avez configuré la réplication entre un cluster de bases de données Aurora MySQL faisant office de source de réplication et une base de données MySQL faisant office de client de réplication, l'instruction `GRANT` pour le rôle ou le privilège entraîne l'arrêt de la réplication avec une erreur. Vous pouvez ignorer sans risque l'erreur afin de reprendre la réplication. Pour ignorer l'erreur sur une instance de base de données RDS for MySQL, utilisez la procédure mysql_rds_skip_repl_error. Pour ignorer l'erreur sur une base de données MySQL externe, utilisez la variable système slave_skip_errors (Aurora MySQL version 2) ou la variable système replica_skip_errors (Aurora MySQL version 3).

Spécification d'un chemin d'accès à un compartiment Amazon S3

La syntaxe permettant de spécifier le chemin de l'emplacement de stockage des données et des fichiers manifeste dans un compartiment Amazon S3 est similaire à celle utilisée dans l'instruction `LOAD DATA FROM S3 PREFIX`, comme indiqué ci-dessous.

```
s3-region://bucket-name/file-prefix
```

Le chemin d'accès inclut les valeurs suivantes :

- `region` (facultatif) – Région AWS qui contient le compartiment Amazon S3 dans lequel enregistrer les données. Ce champ est facultatif. Si vous ne spécifiez pas de valeur `region`, Aurora enregistre vos fichiers dans Amazon S3, dans la même région que votre cluster de bases de données.
- `bucket-name` – Nom du compartiment Amazon S3 où enregistrer les données. Les préfixes d'objet qui identifient un chemin d'accès du dossier virtuel sont pris en charge.
- `file-prefix` – Préfixe d'objet Amazon S3 qui identifie les fichiers à enregistrer dans Amazon S3.

Les fichiers de données créés par l'instruction `SELECT INTO OUTFILE S3` utilisent le chemin suivant, dans lequel *`00000`* représente un nombre entier de base zéro à 5 chiffres.

```
s3-region://bucket-name/file-prefix.part_00000
```

Par exemple, supposez qu'une instruction `SELECT INTO OUTFILE S3` spécifie `s3-us-west-2://bucket/prefix` comme chemin d'accès dans lequel stocker les fichiers de données et crée trois fichiers de données. Le compartiment Amazon S3 spécifié contient les fichiers de données suivants.

- s3-us-west-2://bucket/prefix.part_00000

- s3-us-west-2://bucket/prefix.part_00001
- s3-us-west-2://bucket/prefix.part_00002

Création d'un manifeste pour répertorier les fichiers de données

Vous pouvez utiliser l'instruction `SELECT INTO OUTFILE S3` avec l'option `MANIFEST ON` pour créer un fichier manifeste au format JSON qui répertorie les fichiers texte créés par l'instruction. L'instruction `LOAD DATA FROM S3` peut utiliser le fichier manifeste pour recharger les fichiers de données dans un cluster de bases de données Aurora MySQL. Pour plus d'informations sur l'utilisation d'un manifeste pour charger des fichiers de données dans un cluster de bases de données Aurora MySQL à partir d'Amazon S3, consultez Utilisation d'un manifeste pour spécifier les fichiers de données à charger.

Les fichiers de données inclus dans le manifeste créé par l'instruction `SELECT INTO OUTFILE S3` sont répertoriés dans l'ordre dans lequel ils sont créés par l'instruction. Par exemple, supposez qu'une instruction `SELECT INTO OUTFILE S3` spécifie `s3-us-west-2://bucket/prefix` comme chemin d'accès dans lequel stocker les fichiers de données, et crée trois fichiers de données et un fichier manifeste. Le compartiment Amazon S3 spécifié contient un fichier manifeste nommé `s3-us-west-2://bucket/prefix.manifest`, qui contient les informations suivantes.

```
{
  "entries": [
    {
      "url":"s3-us-west-2://bucket/prefix.part_00000"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00001"
    },
    {
      "url":"s3-us-west-2://bucket/prefix.part_00002"
    }
  ]
}
```

SELECT INTO OUTFILE S3

Vous pouvez utiliser l'instruction `SELECT INTO OUTFILE S3` pour interroger les données d'un cluster de bases de données et les enregistrer directement dans des fichiers texte délimités stockés dans un compartiment Amazon S3.

Les fichiers compressés ne sont pas pris en charge. Les fichiers chiffrés sont pris en charge à partir d'Aurora MySQL version 2.09.0.

Syntaxe

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
        [HIGH_PRIORITY]
        [STRAIGHT_JOIN]
        [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
        [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
        [PARTITION partition_list]
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
        [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
         [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
    [export_options]
    [MANIFEST {ON | OFF}]
    [OVERWRITE {ON | OFF}]
    [ENCRYPTION {ON | OFF | SSE_S3 | SSE_KMS ['cmk_id']}]

export_options:
    [FORMAT {CSV|TEXT} [HEADER]]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
]
```

Paramètres

L'instruction `SELECT INTO OUTFILE S3` utilise les paramètres obligatoires et facultatifs suivants, spécifiques à Aurora.

s3-uri

Spécifie l'URI d'un préfixe Amazon S3 à utiliser. Utilisez la syntaxe décrite dans Spécification d'un chemin d'accès à un compartiment Amazon S3.

FORMAT {CSV|TEXT} [HEADER]

Enregistre éventuellement les données au format CSV.

L'option `TEXT` est la valeur par défaut et produit le format d'exportation MySQL existant.

L'option `CSV` produit des valeurs de données séparées par des virgules. Le format CSV suit la spécification du RFC-4180. Si vous spécifiez le mot-clé facultatif `HEADER`, le fichier de sortie contient une ligne d'en-tête. Les étiquettes de la ligne d'en-tête correspondent aux noms de colonnes de l'instruction `SELECT`. Vous pouvez utiliser les fichiers CSV pour former les modèles de données à utiliser avec les services de ML AWS. Pour plus d'informations sur l'utilisation des données Aurora exportées avec les services de ML AWS, consultez Exportation des données vers Amazon S3 pour l'entraînement des modèles SageMaker AI (avancé).

MANIFEST {ON | OFF}

Indique si un fichier manifeste est créé dans Amazon S3. Le fichier manifeste est un fichier JSON (JavaScript Object Notation) qui peut être utilisé pour charger des données dans un cluster de bases de données Aurora à l'aide de l'instruction `LOAD DATA FROM S3 MANIFEST`. Pour plus d'informations sur `LOAD DATA FROM S3 MANIFEST`, consultez Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3.

Si `MANIFEST ON` est spécifié dans la requête, le fichier manifeste est créé dans Amazon S3 une fois que tous les fichiers de données ont été créés et chargés. Le fichier manifeste est créé à l'aide du chemin suivant :

```
s3-region://bucket-name/file-prefix.manifest
```

Pour plus d'informations sur le format du contenu du fichier manifeste, consultez Création d'un manifeste pour répertorier les fichiers de données.

OVERWRITE {ON | OFF}

Indique si les fichiers existants du compartiment Amazon S3 spécifié sont remplacés. Si `OVERWRITE ON` est spécifié, les fichiers existants qui correspondent au préfixe de fichier dans l'URI spécifié dans `s3-uri` sont remplacés. Dans le cas contraire, une erreur se produit.

ENCRYPTION {ON | OFF | SSE_S3 | SSE_KMS ['*cmk_id*']}

Indique s'il faut utiliser le chiffrement côté serveur avec des clés gérées par Amazon S3 (SSE-S3) ou des clés AWS KMS keys (SSE-KMS, y compris des Clés gérées par AWS et des clés gérées par le client). Les paramètres `SSE_S3` et `SSE_KMS` sont disponibles dans Aurora MySQL 3.05 et versions ultérieures.

Vous pouvez également utiliser la variable de session `aurora_select_into_s3_encryption_default` à la place de la clause `ENCRYPTION`, comme indiqué dans l'exemple suivant. Utilisez la clause SQL ou la variable de session, mais pas les deux.

```
set session set session aurora_select_into_s3_encryption_default={ON | OFF | SSE_S3
 | SSE_KMS};
```

Les paramètres `SSE_S3` et `SSE_KMS` sont disponibles dans Aurora MySQL 3.05 et versions ultérieures.

Lorsque vous définissez `aurora_select_into_s3_encryption_default` sur la valeur suivante :

- `OFF` : la politique de chiffrement par défaut du compartiment S3 est respectée. La valeur par défaut de `aurora_select_into_s3_encryption_default` est `OFF`.
- `ON` ou `SSE_S3` : l'objet S3 est chiffré à l'aide de clés gérées par Amazon S3 (SSE-S3).
- `SSE_KMS` : l'objet S3 est chiffré à l'aide d'une AWS KMS key.

  Dans ce cas, vous incluez également la variable de session `aurora_s3_default_cmk_id`, par exemple :

  ```
  set session aurora_select_into_s3_encryption_default={SSE_KMS};
  set session aurora_s3_default_cmk_id={NULL | 'cmk_id'};
  ```

  - Lorsque la valeur de `aurora_s3_default_cmk_id` est `NULL`, l'objet S3 est chiffré à l'aide d'une Clé gérée par AWS.

- Lorsque `aurora_s3_default_cmk_id` est une chaîne `cmk_id` non vide, l'objet S3 est chiffré à l'aide d'une clé gérée par le client.

  La valeur de `cmk_id` ne peut pas être une chaîne vide.

Lorsque vous utilisez la commande `SELECT INTO OUTFILE S3`, Aurora détermine le chiffrement comme suit :

- Si la clause `ENCRYPTION` est présente dans la commande SQL, Aurora s'appuie uniquement sur la valeur de `ENCRYPTION` et n'utilise pas de variable de session.
- Si la clause `ENCRYPTION` n'est pas présente, Aurora s'appuie sur la valeur de la variable de session.

Pour plus d'informations, consultez Utilisation du chiffrement côté serveur avec des clés gérées par Amazon S3 (SSE-S3) et Utilisation du chiffrement côté serveur avec des clés AWS KMS (SSE-KMS) dans le Guide l'utilisateur d'Amazon Simple Storage Service.

Pour plus d'informations sur les autres paramètres, consultez Instruction SELECT et Instruction LOAD DATA, dans la documentation sur MySQL.

Considérations

Le nombre de fichiers écrits dans le compartiment Amazon S3 dépend de la quantité de données sélectionnées par l'instruction `SELECT INTO OUTFILE S3` et du seuil de taille de fichier défini pour Aurora MySQL. Le seuil de taille de fichier par défaut est de 6 giga-octets (Go). Si les données sélectionnées par l'instruction sont inférieures au seuil de taille de fichier, un fichier unique est créé. Dans le cas contraire, plusieurs fichiers sont créés. Autres considérations relatives aux fichiers créés par cette instruction :

- Aurora MySQL garantit que les lignes des fichiers de données ne sont pas fractionnées à la limite de taille des fichiers. Pour plusieurs fichiers, la taille de chaque fichier de données à l'exception du dernier est en général proche du seuil de taille de fichier. Toutefois, le fait de rester sous le seuil de taille de fichier peut entraîner à l'occasion le fractionnement d'une ligne sur deux fichiers de données. Dans ce cas, Aurora MySQL crée un fichier de données qui conserve la ligne intacte, mais dont la taille peut être supérieure au seuil de taille de fichier.
- Sachant que chaque instruction `SELECT` dans Aurora MySQL s'exécute comme une transaction atomique, l'exécution d'une instruction `SELECT INTO OUTFILE S3` qui sélectionne un ensemble de données volumineux peut prendre un certain temps. Si l'instruction échoue pour une raison quelconque, vous pouvez avoir besoin de recommencer et de réémettre l'instruction. Cependant,

si l'instruction échoue, les fichiers déjà chargés dans Amazon S3 restent dans le compartiment Amazon S3 spécifié. Vous pouvez utiliser une autre instruction pour charger les données restantes au lieu de reprendre du début.

- Si la quantité de données à sélectionner est importante (plus de 25 Go), nous vous recommandons d'utiliser plusieurs instructions `SELECT INTO OUTFILE S3` pour enregistrer les données dans Amazon S3. Chaque instruction doit sélectionner une partie différente des données à enregistrer et doit également spécifier un `file_prefix` différent dans le paramètre `s3-uri` à utiliser lors de l'enregistrement des fichiers de données. Le partitionnement des données à sélectionner avec plusieurs instructions facilite la récupération d'une erreur dans une instruction. Si une erreur se produit pour une instruction, seule une partie des données doit être resélectionnée et téléchargée vers Amazon S3. L'utilisation de plusieurs instructions aide également à éviter une transaction unique de longue durée, ce qui peut améliorer les performances.
- Si plusieurs instructions `SELECT INTO OUTFILE S3` utilisant le même `file_prefix` dans le paramètre `s3-uri` s'exécutent en parallèle pour sélectionner des données dans Amazon S3, le comportement est indéfini.
- Certaines métadonnées, comme les métadonnées de fichier ou de schéma de table, ne sont pas chargées par Aurora MySQL dans Amazon S3.
- Dans certains cas, vous pouvez réexécuter une requête `SELECT INTO OUTFILE S3`, par exemple pour récupérer à partir d'une défaillance. Dans ce cas, vous devez soit supprimer tous les fichiers de données existants du compartiment Amazon S3 qui présentent le préfixe de fichier spécifié dans `s3-uri`, soit inclure `OVERWRITE ON` dans la requête `SELECT INTO OUTFILE S3`.

L'instruction `SELECT INTO OUTFILE S3` retourne une réponse et un numéro d'erreur MySQL standard en cas de réussite ou d'échec. Si vous n'avez pas accès à la réponse et au numéro d'erreur MySQL, la façon la plus simple d'être informé de la conclusion de l'opération consiste à spécifier `MANIFEST ON` dans l'instruction. Le fichier manifeste est le dernier fichier écrit par l'instruction. En d'autres termes, si vous disposez d'un fichier manifeste, cela signifie que l'instruction est terminée.

Actuellement, il n'existe aucun moyen de surveiller directement la progression de l'instruction `SELECT INTO OUTFILE S3` pendant son exécution. Toutefois, supposons que vous écrivez une grande quantité de données d'Aurora MySQL vers Amazon S3 à l'aide de cette instruction et que vous connaissez la taille des données sélectionnées par l'instruction. Dans ce cas, vous pouvez estimer la progression en surveillant la création des fichiers de données dans Amazon S3.

Pour ce faire, vous pouvez utiliser le fait qu'un fichier de données est créé dans le compartiment Amazon S3 spécifié pour environ 6 Go de données sélectionnées par l'instruction. Divisez la taille

des données sélectionnées par 6 Go pour obtenir une estimation du nombre de fichiers de données à créer. Vous pouvez alors estimer la progression de l'instruction en surveillant le nombre de fichiers chargés vers Amazon S3 pendant son exécution.

Exemples

L'instruction ci-dessous sélectionne toutes les données de la table `employees` et les enregistre dans un compartiment Amazon S3 situé dans une région différente de celle du cluster de bases de données Aurora MySQL. L'instruction crée des fichiers de données dans lesquels chaque champ se termine par une virgule (`,`) et chaque ligne par un caractère de saut de ligne (`\n`). L'instruction renvoie une erreur si des fichiers correspondant au préfixe de fichier `sample_employee_data` existent dans le compartiment Amazon S3 spécifié.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n';
```

L'instruction suivante sélectionne toutes les données de la table `employees` et les enregistre dans un compartiment Amazon S3 situé dans la même région que le cluster de bases de données Aurora MySQL. L'instruction crée des fichiers de données dans lesquels chaque champ se termine par une virgule (`,`) et chaque ligne par un caractère de saut de ligne (`\n`), et crée également un fichier manifeste. L'instruction renvoie une erreur si des fichiers correspondant au préfixe de fichier `sample_employee_data` existent dans le compartiment Amazon S3 spécifié.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON;
```

L'instruction ci-dessous sélectionne toutes les données de la table `employees` et les enregistre dans un compartiment Amazon S3 situé dans une région différente de celle du cluster de bases de données Aurora. L'instruction crée des fichiers de données dans lesquels chaque champ se termine par une virgule (`,`) et chaque ligne par un caractère de saut de ligne (`\n`). L'instruction remplace tout fichier existant correspondant au préfixe de fichier `sample_employee_data` dans le compartiment Amazon S3 spécifié.

```
SELECT * FROM employees INTO OUTFILE S3 's3-us-west-2://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    OVERWRITE ON;
```

L'instruction suivante sélectionne toutes les données de la table `employees` et les enregistre dans un compartiment Amazon S3 situé dans la même région que le cluster de bases de données Aurora MySQL. L'instruction crée des fichiers de données dans lesquels chaque champ se termine par une virgule (`,`) et chaque ligne par un caractère de saut de ligne (`\n`), et crée également un fichier manifeste. L'instruction remplace tout fichier existant correspondant au préfixe de fichier `sample_employee_data` dans le compartiment Amazon S3 spécifié.

```
SELECT * FROM employees INTO OUTFILE S3 's3://aurora-select-into-s3-pdx/
sample_employee_data'
    FIELDS TERMINATED BY ','
    LINES TERMINATED BY '\n'
    MANIFEST ON
    OVERWRITE ON;
```

Appel d'une fonction Lambda à partir d'un cluster de bases de données Amazon Aurora MySQL

Vous pouvez appeler une fonction AWS Lambda à partir d'un cluster de bases de données Amazon Aurora Édition compatible avec MySQL avec la fonction native `lambda_sync` ou `lambda_async`. Avant d'appeler une fonction Lambda à partir d'un cluster de bases de données Aurora MySQL, le cluster de bases de données Aurora doit avoir accès à Lambda. Pour plus d'informations sur l'octroi d'un accès à Aurora MySQL, consultez Octroi à Aurora d'un accès à Lambda. Pour plus d'informations sur les fonctions stockées `lambda_sync` et `lambda_async`, consultez Appel d'une fonction Lambda avec une fonction native Aurora MySQL.

Vous pouvez également appeler une fonction AWS Lambda à l'aide d'une procédure stockée. L'utilisation d'une procédure stockée est cependant déconseillée. Il est vivement recommandé d'utiliser une fonction native Aurora MySQL si vous utilisez l'une des versions Aurora MySQL suivantes :

- Aurora MySQL version 2 pour les clusters compatibles avec MySQL 5.7.

- Aurora MySQL version 3.01 et ultérieure, pour les clusters compatibles MySQL 8.0. La procédure stockée n'est pas disponible dans Aurora MySQL version 3.

Pour en savoir plus sur l'accès d'Aurora à Lambda et l'appel d'une fonction Lambda, reportez-vous aux rubriques suivantes.

Rubriques

Octroi à Aurora d'un accès à Lambda

Pour pouvoir appeler des fonctions Lambda à partir d'un cluster de bases de données Aurora MySQL, veillez d'abord à octroyer à votre cluster une autorisation d'accès à Lambda.

Pour octroyer à Aurora MySQL un accès à Lambda

1. Créez une stratégie Gestion des identités et des accès AWS (IAM) pour fournir les autorisations permettant au cluster de base de données Aurora MySQL d'appeler des fonctions Lambda. Pour obtenir des instructions, consultez Création d'une politique IAM pour accéder aux ressources AWS Lambda.
2. Créez un rôle IAM et attachez la stratégie IAM que vous avez créée dans Création d'une politique IAM pour accéder aux ressources AWS Lambda au nouveau rôle IAM. Pour obtenir des instructions, consultez Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS.
3. Définissez le paramètre de cluster de bases de données `aws_default_lambda_role` sur l'Amazon Resource Name (ARN) du nouveau rôle IAM.

   Si le cluster fait partie d'une base de données globale Aurora, appliquez le même paramètre pour chaque cluster Aurora de cette base de données.

   Pour plus d'informations sur les paramètres de cluster de base de données, consultez Paramètres de cluster de bases de données et d'instance de base de données Amazon Aurora.
4. Pour autoriser les utilisateurs de base de données d'un cluster de bases de données Aurora MySQL à appeler des fonctions Lambda, associez le rôle que vous avez créé dans Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS au cluster de bases

de données. Pour plus d'informations sur l'association d'un rôle IAM à un cluster de bases de données, consultez Association d'un rôle IAM à un cluster de bases de données Amazon Aurora MySQL.

Si le cluster fait partie d'une base de données globale Aurora, associez le rôle à chaque cluster Aurora de cette base de données.

5. Configurez votre cluster de bases de données Aurora MySQL de façon à autoriser les connexions sortantes vers Lambda. Pour obtenir des instructions, consultez Activation de la communication réseau entre Amazon Aurora et d'autres services AWS.

   Si le cluster fait partie d'une base de données globale Aurora, activez les connexions sortantes pour chaque cluster Aurora de cette base de données.

Appel d'une fonction Lambda avec une fonction native Aurora MySQL

> **Note**
>
> Vous pouvez appeler les fonctions natives `lambda_sync` et `lambda_async` quand vous utilisez Aurora MySQL version 2 ou Aurora MySQL version 3.01 ou ultérieure. Pour plus d'informations sur les versions d'Aurora MySQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL.

Vous pouvez appeler une fonction AWS Lambda à partir d'un cluster de bases de données Aurora MySQL en appelant les fonctions natives `lambda_sync` et `lambda_async`. Cette approche peut être utile lorsque vous souhaitez intégrer votre base de données s'exécutant sur Aurora MySQL avec d'autres services AWS. Par exemple, vous pouvez souhaiter envoyer une notification avec Amazon Simple Notification Service (Amazon SNS) chaque fois qu'une ligne est insérée dans une table spécifique de votre base de données.

Table des matières

- Exemple de la fonction lambda_sync
- Syntaxe de la fonction lambda_async
- Paramètres de la fonction lambda_async
- Exemple de la fonction lambda_async
- Appel d'une fonction Lambda dans un déclencheur

Utilisation d'une fonction Lambda avec une fonction native

Les fonctions natives `lambda_sync` et `lambda_async` sont des fonctions natives intégrées qui appellent une fonction Lambda de façon synchrone ou asynchrone. Lorsque vous devez connaître le résultat de la fonction Lambda avant de passer à une autre action, utilisez la fonction synchrone `lambda_sync`. Lorsque vous n'avez pas besoin de connaître le résultat de la fonction Lambda avant de passer à une autre action, utilisez la fonction asynchrone `lambda_async`.

Octroi du rôle dans Aurora MySQL version 3

Dans Aurora MySQL version 3, l'utilisateur qui appelle une fonction native doit disposer du rôle `AWS_LAMBDA_ACCESS`. Pour accorder ce rôle à un utilisateur, connectez-vous à l'instance de base de données en tant qu'utilisateur administratif, puis exécutez l'instruction suivante.

```
GRANT AWS_LAMBDA_ACCESS TO user@domain-or-ip-address
```

Vous pouvez révoquer ce rôle en exécutant l'instruction suivante.

```
REVOKE AWS_LAMBDA_ACCESS FROM user@domain-or-ip-address
```

> **Tip**
>
> Lorsque vous utilisez la technique de rôle dans Aurora MySQL version 3, vous pouvez également activer le rôle en utilisant l'instruction `SET ROLE role_name` ou `SET ROLE ALL`. Si vous n'êtes pas familier avec le système de rôles MySQL 8.0, vous pouvez en apprendre davantage dans Modèle de privilège basé sur les rôles. Pour plus de détails, consultez Using roles dans le manuel de référence de MySQL.
>
> Cela s'applique uniquement à la session active en cours. Lorsque vous vous reconnectez, vous devez à nouveau exécuter l'instruction `SET ROLE` pour accorder des privilèges. Pour plus d'informations, consultez SET ROLE statement dans le manuel MySQL Reference Manual.

Vous pouvez utiliser le paramètre `activate_all_roles_on_login` de cluster de bases de données pour activer automatiquement tous les rôles lorsqu'un utilisateur se connecte à une instance de base de données. Lorsque ce paramètre est défini, vous n'avez pas besoin d'appeler explicitement l'instruction `SET ROLE` pour activer un rôle. Pour plus d'informations, consultez activate_all_roles_on_login dans le manuel MySQL Reference Manual. Toutefois, vous devez appeler `SET ROLE ALL` explicitement au début d'une procédure stockée pour activer le rôle, lorsque cette procédure est appelée par un autre utilisateur.

Si vous obtenez une erreur telle que la suivante lorsque vous essayez d'invoquer une fonction Lambda, exécutez une instruction `SET ROLE`.

```
SQL Error [1227] [42000]: Access denied; you need (at least one of) the Invoke Lambda
 privilege(s) for this operation
```

Assurez-vous d'attribuer le rôle à l'utilisateur approprié, comme indiqué dans les entrées de la table `mysql.users`. Plusieurs utilisateurs peuvent avoir le même nom, mais sur des hôtes différents. En fonction de l'application ou de l'hôte qui invoque la fonction `lambda_sync`, MySQL sélectionne l'utilisateur présentant la meilleure correspondance selon les entrées de colonne `host`.

Octroi du privilège dans Aurora MySQL version 2

Dans Aurora MySQL version 2, l'utilisateur qui appelle une fonction native doit se voir accorder le privilège `INVOKE LAMBDA`. Pour accorder ce privilège à un utilisateur, connectez-vous à l'instance de base de données en tant qu'utilisateur administratif, puis exécutez l'instruction suivante.

```
GRANT INVOKE LAMBDA ON *.* TO user@domain-or-ip-address
```

Vous pouvez révoquer ce privilège en exécutant l'instruction suivante.

```
REVOKE INVOKE LAMBDA ON *.* FROM user@domain-or-ip-address
```

Syntaxe de la fonction lambda_sync

Vous appelez la fonction `lambda_sync` de façon synchrone avec le type d'appel `RequestResponse`. La fonction renvoie le résultat de l'appel Lambda dans des données utiles JSON. La fonction a la syntaxe suivante.

```
lambda_sync (
  lambda_function_ARN,
  JSON_payload
)
```

Paramètres de la fonction lambda_sync

La fonction `lambda_sync` possède les paramètres suivants.

lambda_function_ARN

Amazon Resource Name (ARN) de la fonction Lambda à appeler.

JSON_payload

Charge utile de la fonction Lambda appelée au format JSON.

> **Note**
>
> Aurora MySQL version 3 prend en charge les fonctions d'analyse JSON de MySQL 8.0. Toutefois, Aurora MySQL version 2 n'inclut pas ces fonctions. L'analyse JSON n'est pas requise lorsqu'une fonction Lambda renvoie une valeur atomique, telle qu'un numéro ou une chaîne.

Exemple de la fonction lambda_sync

La requête suivante basée sur `lambda_sync` appelle la fonction Lambda `BasicTestLambda` de façon synchrone en utilisant l'ARN de la fonction. La charge utile de la fonction est `{"operation": "ping"}`.

```
SELECT lambda_sync(
    'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
    '{"operation": "ping"}');
```

Syntaxe de la fonction lambda_async

Vous appelez la fonction `lambda_async` de façon asynchrone avec le type d'appel `Event`. La fonction renvoie le résultat de l'appel Lambda dans des données utiles JSON. La fonction a la syntaxe suivante.

```
lambda_async (
  lambda_function_ARN,
  JSON_payload
)
```

Paramètres de la fonction lambda_async

La fonction `lambda_async` possède les paramètres suivants.

lambda_function_ARN

Amazon Resource Name (ARN) de la fonction Lambda à appeler.

JSON_payload

Charge utile de la fonction Lambda appelée au format JSON.

> **Note**
>
> Aurora MySQL version 3 prend en charge les fonctions d'analyse JSON de MySQL 8.0. Toutefois, Aurora MySQL version 2 n'inclut pas ces fonctions. L'analyse JSON n'est pas requise lorsqu'une fonction Lambda renvoie une valeur atomique, telle qu'un numéro ou une chaîne.

Exemple de la fonction lambda_async

La requête suivante basée sur `lambda_async` appelle la fonction Lambda `BasicTestLambda` de façon asynchrone en utilisant l'ARN de la fonction. La charge utile de la fonction est `{"operation": "ping"}`.

```
SELECT lambda_async(
    'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
    '{"operation": "ping"}');
```

Appel d'une fonction Lambda dans un déclencheur

Vous pouvez utiliser des déclencheurs pour appeler Lambda sur les instructions de modification de données. L'exemple suivant utilise la fonction native `lambda_async` et stocke le résultat dans une variable.

```
mysql>SET @result=0;
mysql>DELIMITER //
mysql>CREATE TRIGGER myFirstTrigger
      AFTER INSERT
          ON Test_trigger FOR EACH ROW
      BEGIN
      SELECT lambda_async(
          'arn:aws:lambda:us-east-1:123456789012:function:BasicTestLambda',
          '{"operation": "ping"}')
          INTO @result;
      END; //
mysql>DELIMITER ;
```

> **Note**
>
> Les déclencheurs ne sont pas exécutés une fois par instruction SQL, mais une fois par ligne modifiée, une ligne à la fois. Lorsqu'un déclencheur s'exécute, le processus est synchrone. L'instruction de modification des données n'est retournée que lorsque le déclencheur est terminé.
>
> Soyez prudent lorsque vous appelez une fonction AWS Lambda à partir de déclencheurs sur des tables qui connaissent un trafic d'écriture élevé. Les déclencheurs `INSERT`, `UPDATE` et `DELETE` sont activés par ligne. Une charge de travail d'écriture intensive sur une table avec des déclencheurs `INSERT`, `UPDATE` ou `DELETE` produit un grand nombre d'appels de votre fonction AWS Lambda.

Appel d'une fonction Lambda avec une procédure stockée Aurora MySQL (obsolète)

Vous pouvez appeler une fonction AWS Lambda à partir d'un cluster de base de données Aurora MySQL en appelant la procédure `mysql.lambda_async`. Cette approche peut être utile lorsque vous souhaitez intégrer votre base de données s'exécutant sur Aurora MySQL avec d'autres services AWS. Par exemple, vous pouvez souhaiter envoyer une notification avec Amazon Simple Notification Service (Amazon SNS) chaque fois qu'une ligne est insérée dans une table spécifique de votre base de données.

Table des matières

Remarques relatives aux versions Aurora MySQL

Depuis Aurora MySQL version 2, vous pouvez utiliser la méthode des fonctions natives à la place de ces procédures stockées pour appeler une fonction Lambda. Pour de plus amples informations sur les fonctions natives, veuillez consulter Utilisation d'une fonction Lambda avec une fonction native.

Dans Aurora MySQL version 2, la procédure stockée `mysql.lambda_async` n'est plus prise en charge. Nous vous recommandons vivement de travailler plutôt avec des fonctions Lambda natives.

Dans Aurora MySQL version 3, la procédure stockée n'est pas disponible.

Utilisation de la procédure mysql.lambda_async pour appeler une fonction Lambda (obsolète)

La procédure `mysql.lambda_async` est une procédure stockée intégrée qui appelle une fonction Lambda de manière asynchrone. Pour utiliser cette procédure, votre utilisateur de base de données doit disposer du privilège `EXECUTE` sur la procédure stockée `mysql.lambda_async`.

Syntaxe

La procédure `mysql.lambda_async` possède la syntaxe suivante.

```
CALL mysql.lambda_async (
  lambda_function_ARN,
  lambda_function_input
)
```

Paramètres

La procédure `mysql.lambda_async` possède les paramètres suivants.

lambda_function_ARN

Amazon Resource Name (ARN) de la fonction Lambda à appeler.

lambda_function_input

Chaîne d'entrée, au format JSON, pour la fonction Lambda appelée.

Exemples

Au titre de bonne pratique, nous vous recommandons d'encapsuler les appels de la procédure `mysql.lambda_async` dans une procédure stockée qui peut être appelée depuis différentes sources, telles que des déclencheurs ou le code client. Cette approche peut permettre d'éviter les problèmes d'incohérence d'impédance et faciliter l'appel des fonctions Lambda.

> **Note**
>
> Soyez prudent lorsque vous appelez une fonction AWS Lambda à partir de déclencheurs sur des tables qui connaissent un trafic d'écriture élevé. Les déclencheurs `INSERT`, `UPDATE` et `DELETE` sont activés par ligne. Une charge de travail d'écriture intensive sur une table avec des déclencheurs `INSERT`, `UPDATE` ou `DELETE` produit un grand nombre d'appels de votre fonction AWS Lambda.
>
> Même si les appels à la procédure `mysql.lambda_async` sont asynchrones, les déclencheurs sont synchrones. Une instruction qui produit un grand nombre d'activation de déclencheur n'attend pas la fin de l'appel de la fonction AWS Lambda, elle attend que les déclencheurs se terminent avant de rendre le contrôle au client.

Example Exemple : invoquer une fonction AWS Lambda pour envoyer un e-mail

L'exemple suivant crée une procédure stockée que vous pouvez appeler dans votre code de base de données pour envoyer un e-mail à l'aide d'une fonction Lambda.

Fonction AWS Lambda

```
import boto3

ses = boto3.client('ses')

def SES_send_email(event, context):

    return ses.send_email(
        Source=event['email_from'],
        Destination={
            'ToAddresses': [
            event['email_to'],
            ]
        },
```

```
        Message={
            'Subject': {
            'Data': event['email_subject']
            },
            'Body': {
                'Text': {
                    'Data': event['email_body']
                }
            }
        }
    )
```

Procédure stockée

```
DROP PROCEDURE IF EXISTS SES_send_email;
DELIMITER ;;
  CREATE PROCEDURE SES_send_email(IN email_from VARCHAR(255),
                                  IN email_to VARCHAR(255),
                                  IN subject VARCHAR(255),
                                  IN body TEXT) LANGUAGE SQL
  BEGIN
    CALL mysql.lambda_async(
         'arn:aws:lambda:us-west-2:123456789012:function:SES_send_email',
         CONCAT('{"email_to" : "', email_to,
             '", "email_from" : "', email_from,
             '", "email_subject" : "', subject,
             '", "email_body" : "', body, '"}')
     );
  END
  ;;
DELIMITER ;
```

Appel de la procédure stockée pour appeler la fonction AWS Lambda

```
mysql> call SES_send_email('example_from@amazon.com', 'example_to@amazon.com', 'Email
 subject', 'Email content');
```

Example Exemple : invoquer une fonction AWS Lambda pour publier un événement à partir d'un déclencheur

L'exemple suivant crée une procédure stockée qui publie un événement avec Amazon SNS. Le code appelle la procédure à partir d'un déclencheur lorsqu'une ligne est ajoutée à une table.

Fonction AWS Lambda

```
import boto3

sns = boto3.client('sns')

def SNS_publish_message(event, context):

    return sns.publish(
        TopicArn='arn:aws:sns:us-west-2:123456789012:Sample_Topic',
        Message=event['message'],
        Subject=event['subject'],
        MessageStructure='string'
    )
```

Procédure stockée

```
DROP PROCEDURE IF EXISTS SNS_Publish_Message;
DELIMITER ;;
CREATE PROCEDURE SNS_Publish_Message (IN subject VARCHAR(255),
                                      IN message TEXT) LANGUAGE SQL
BEGIN
  CALL mysql.lambda_async('arn:aws:lambda:us-
west-2:123456789012:function:SNS_publish_message',
     CONCAT('{ "subject" : "', subject,
            '", "message" : "', message, '" }')
     );
END
;;
DELIMITER ;
```

Tableau

```
CREATE TABLE 'Customer_Feedback' (
  'id' int(11) NOT NULL AUTO_INCREMENT,
  'customer_name' varchar(255) NOT NULL,
  'customer_feedback' varchar(1024) NOT NULL,
  PRIMARY KEY ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Déclencheur

```
DELIMITER ;;
CREATE TRIGGER TR_Customer_Feedback_AI
  AFTER INSERT ON Customer_Feedback
  FOR EACH ROW
BEGIN
  SELECT CONCAT('New customer feedback from ', NEW.customer_name),
 NEW.customer_feedback INTO @subject, @feedback;
  CALL SNS_Publish_Message(@subject, @feedback);
END
;;
DELIMITER ;
```

Insertion d'une ligne dans la table pour déclencher la notification

```
mysql> insert into Customer_Feedback (customer_name, customer_feedback) VALUES ('Sample
 Customer', 'Good job guys!');
```

Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs

Vous pouvez configurer votre cluster de bases de données Aurora MySQL afin qu'il publie les données de journaux généraux, de journaux de requêtes lentes, de journaux d'audit et de journaux d'erreurs dans un groupe de journaux dans Amazon CloudWatch Logs. CloudWatch Logs vous permet d'effectuer une analyse en temps réel des données de journaux et d'utiliser CloudWatch pour créer des alarmes et afficher des métriques. CloudWatch Logs permet de conserver les enregistrements des journaux dans une solution de stockage hautement durable.

Pour pouvoir publier des journaux dans CloudWatch Logs, il faut que ces journaux soient activés. Les journaux d'erreurs sont activés par défaut mais vous devez activer explicitement les autres types de journaux. Pour plus d'informations sur l'activation de journaux dans MySQL, consultez Selecting General Query and Slow Query Log Output Destinations dans la documentation MySQL. Pour plus d'informations sur l'activation des journaux d'audit Aurora MySQL, consultez Activation de l'Audit avancé.

> **Note**
>
> - Si l'exportation des données de journaux est désactivée, Aurora ne supprime pas les groupes de journaux ou les flux de journaux existants. Si l'exportation des données de journaux est désactivée, les données de journaux existantes restent disponibles dans

CloudWatch Logs en fonction de la rétention de journaux ; des frais peuvent être appliqués pour le stockage des données de journaux d'audit. Vous pouvez supprimer des flux de journaux et des groupes de journaux à partir de la console CloudWatch Logs, de l'AWS CLI ou de l'API CloudWatch Logs.

- Une autre façon de publier des journaux d'audit dans CloudWatch Logs consiste à activer la fonction d'audit avancé, puis à créer un groupe de paramètres de cluster de bases de données et à définir le paramètre `server_audit_logs_upload` sur `1`. La valeur par défaut du paramètre de cluster de bases de données `server_audit_logs_upload` est `0`. Pour plus d'informations sur l'activation de l'audit avancé, consultez Utilisation de l'Audit avancé avec un cluster de bases de données Amazon Aurora MySQL.

  Si vous utilisez cette autre méthode, vous devez disposer d'un rôle IAM pour accéder à CloudWatch Logs et définir le paramètre de niveau cluster `aws_default_logs_role` sur l'ARN de ce rôle. Pour plus d'informations sur la création d'un rôle , consultez Configuration de rôles IAM pour accéder aux services AWS. Cependant, si vous disposez du rôle lié à un service `AWSServiceRoleForRDS`, il donne accès à CloudWatch Logs et remplace les rôles personnalisés. Pour plus d'informations sur les rôles liés à un service pour Amazon RDS, consultez Utilisation des rôles liés à un service pour Amazon Aurora.
- Si vous ne souhaitez pas exporter les journaux d'audit vers CloudWatch Logs, veillez à ce que toutes les méthodes d'exportation de journaux d'audit soient désactivées. Ces méthodes reposent sur AWS Management Console, l'AWS CLI, l'API RDS et le paramètre `server_audit_logs_upload`.
- La procédure est légèrement différente pour les clusters de bases de données Aurora Serverless v1 que pour les clusters de bases de données avec des instances de base de données Aurora Serverless v2 ou provisionnées. Les clusters Aurora Serverless v1 chargent automatiquement tous les types de journaux que vous activez via les paramètres de configuration.

  Ainsi, pour activer ou désactiver le chargement des journaux pour les clusters de bases de données Aurora Serverless v1, vous devez activer ou désactiver différents types de journaux dans le groupe de paramètres de cluster de bases de données. Vous ne modifiez pas les paramètres du cluster lui-même via AWS Management Console, l'AWS CLI ou l'API RDS. Pour obtenir plus d'informations sur l'activation et la désactivation des journaux MySQL pour les clusters Aurora Serverless v1, consultez Groupes de paramètres pour Aurora Serverless v1.

Console

Vous pouvez publier des journaux Aurora MySQL pour les clusters alloués dans CloudWatch Logs avec la console.

Pour publier des journaux Aurora MySQL à partir de la console

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster de bases de données Aurora MySQL dont vous voulez publier les données de journaux.
4. Sélectionnez Modify.
5. Dans la section Exportations des journaux, choisissez les journaux que vous voulez commencer à publier dans CloudWatch Logs.
6. Choisissez Continuer, puis Modifier le cluster DB sur la page récapitulative.

AWS CLI

Vous pouvez publier des journaux Aurora MySQL pour les clusters alloués avec l'AWS CLI. Pour cela, vous exécutez la commande modify-db-cluster de l'AWS CLI avec les options suivantes :

- `--db-cluster-identifier` — Identifiant du cluster de bases de données.
- `--cloudwatch-logs-export-configuration` — Paramètre de configuration des types de journaux à activer pour l'exportation vers CloudWatch Logs pour le cluster de bases de données.

Vous pouvez également publier des journaux Aurora MySQL en exécutant l'une des commandes de l'AWS CLI suivantes :

- create-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-snapshot
- restore-db-cluster-to-point-in-time

Exécutez l'une de ces commandes de l'AWS CLI avec les options suivantes :

- `--db-cluster-identifier` — Identifiant du cluster de bases de données.

- `--engine` — Moteur de base de données.
- `--enable-cloudwatch-logs-exports` — Paramètre de configuration des types de journaux à activer pour l'exportation vers CloudWatch Logs pour le cluster de bases de données.

D'autres options peuvent être requises en fonction de la commande d'AWS CLI que vous exécutez.

Example

La commande suivante modifie un cluster de bases de données Aurora MySQL existant afin qu'il publie les fichiers journaux dans CloudWatch Logs.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["error","general","slowquery","audit","instance"]}'
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["error","general","slowquery","audit","instance"]}'
```

Example

La commande suivante crée un cluster de bases de données Aurora MySQL qui publie les fichiers journaux dans CloudWatch Logs.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine aurora \
    --enable-cloudwatch-logs-exports
 '["error","general","slowquery","audit","instance"]'
```

Pour Windows :

```
aws rds create-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine aurora ^
    --enable-cloudwatch-logs-exports
 '["error","general","slowquery","audit","instance"]'
```

API RDS

Vous pouvez publier des journaux Aurora MySQL pour les clusters alloués avec l'API RDS. Pour cela, vous exécutez l'opération ModifyDBCluster avec les options suivantes :

- `DBClusterIdentifier` — Identifiant du cluster de bases de données.
- `CloudwatchLogsExportConfiguration` — Paramètre de configuration des types de journaux à activer pour l'exportation vers CloudWatch Logs pour le cluster de bases de données.

Vous pouvez également publier des journaux Aurora MySQL avec l'API RDS en exécutant l'une des opérations d'API RDS suivantes :

- CreateDBCluster
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime

Exécutez l'opération d'API RDS avec les paramètres suivants :

- `DBClusterIdentifier` — Identifiant du cluster de bases de données.
- `Engine` — Moteur de base de données.
- `EnableCloudwatchLogsExports` — Paramètre de configuration des types de journaux à activer pour l'exportation vers CloudWatch Logs pour le cluster de bases de données.

D'autres paramètres peuvent être requis en fonction de la commande d'AWS CLI que vous exécutez.

Surveillance des événements de journaux dans Amazon CloudWatch

Après avoir activé les événements de journaux Aurora MySQL, vous pouvez les surveiller dans Amazon CloudWatch Logs. Un nouveau groupe de journaux est automatiquement créé pour le cluster de bases de données Aurora sous le préfixe suivant, dans lequel *`cluster-name`* représente le nom du cluster de bases de données et *`log_type`* le type de journal.

```
/aws/rds/cluster/cluster-name/log_type
```

Par exemple, si vous configurez la fonction d'exportation de sorte à inclure le journal de requêtes lentes pour un cluster de bases de données nommé `mydbcluster`, les données de requêtes lentes sont stockées dans le groupe de journaux `/aws/rds/cluster/mydbcluster/slowquery`.

Les événements de toutes les instances dans votre cluster sont transmis en mode push vers un groupe de journaux par l'intermédiaire de différents flux de journaux. Le comportement dépend des conditions suivante qui sont vraies :

- Un groupe de journaux avec le nom spécifié existe.

  Aurora utilise le groupe de journaux existant pour exporter les données de journal du cluster. Pour créer des groupes de journaux avec des périodes de rétention de journaux, des filtres de métriques et des accès client prédéfinis, vous pouvez utiliser une configuration automatisée, telle que AWS CloudFormation.
- Aucun groupe de journaux avec le nom spécifié n'existe.

  Lorsqu'une entrée de journal correspondante est détectée dans le fichier journal de l'instance, Aurora MySQL crée automatiquement un nouveau groupe de journaux dans CloudWatch Logs. Le groupe de journaux utilise la période de rétention de journaux par défaut Never Expire (N'expire jamais).

  Pour modifier la période de rétention des journaux, vous pouvez utiliser la console CloudWatch Logs, la AWS CLI ou l'API CloudWatch Logs. Pour plus d'informations sur la modification des périodes de rétention des journaux dans CloudWatch Logs, consultez Modification de la rétention des données de journaux dans CloudWatch Logs.

Pour rechercher des informations dans les événements du journal pour un cluster de bases de données, utilisez la console CloudWatch Logs, la AWS CLI ou l'API CloudWatch Logs. Pour plus

d’informations sur la recherche et le filtrage des données de journaux, consultez Recherche et filtrage des données de journaux.

Mode Lab Amazon Aurora MySQL

> **Important**
>
> Le mode lab a été introduit dans la version 2 d'Aurora MySQL pour permettre l'optimisation Fast DDL, qui améliore l'efficacité de certaines opérations DDL. Dans Aurora MySQL version 3, le mode lab a été supprimé et Fast DDL a été remplacé par la fonctionnalité Instant DDL de MySQL 8.0.

Le mode Lab Aurora permet d'activer les fonctions Aurora qui sont disponibles dans la version de base de données Aurora actuelle, mais qui ne sont pas activées par défaut. Si l'utilisation des fonctions du mode Lab Aurora n'est pas recommandée dans les clusters de bases de données de production, vous pouvez utiliser le mode Lab Aurora pour activer ces fonctions pour les clusters de bases de données de vos environnements de développement et de test. Pour plus d'informations sur les fonctions Aurora disponibles lorsque le mode Lab Aurora est activé, consultez Fonctions du mode Lab Aurora.

Le paramètre `aurora_lab_mode` est un paramètre de niveau instance qui figure dans le groupe de paramètres par défaut. Le paramètre est défini sur `0` (désactivé) dans le groupe de paramètres par défaut. Pour activer le mode Lab Aurora, créez un groupe de paramètres personnalisé, définissez le paramètre `aurora_lab_mode` sur `1` (activé) dans le groupe de paramètres personnalisé, puis modifiez une ou plusieurs instances de votre cluster Aurora pour utiliser le groupe de paramètres personnalisé. Ensuite, connectez-vous au point de terminaison approprié pour tester les fonctionnalités du mode Lab. Pour plus d'informations sur la modification d'un groupe de paramètres DB, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora. Pour plus d'informations sur les groupes de paramètres et Amazon Aurora, consultez Paramètres de configuration d'Aurora MySQL.

Fonctions du mode Lab Aurora

Le tableau suivant répertorie les fonctionnalités Aurora actuellement disponibles lorsque le mode lab Aurora est activé. Vous devez activer le mode Lab Aurora afin de pouvoir utiliser ces fonctions.

Fast DDL

Cette fonctionnalité vous permet d'exécuter une opération `ALTER TABLE` *`tbl_name`* `ADD COLUMN` *`col_name column_definition`* presque instantanément. L'opération s'effectue sans

nécessiter la copie de la table et sans impact matériel sur les autres instructions DML. Comme cette opération ne consomme pas de stockage temporaire pour une copie de table, elle rend les instructions DDL pratiques même pour des tables volumineuses sur des classes d'instance de petite taille.

FAST DLL prend actuellement uniquement en charge l'ajout d'une colonne acceptant la valeur null, sans valeur par défaut, à la fin d'une table. Pour plus d'informations sur l'utilisation de cette fonction, consultez Modification de tables dans Amazon Aurora à l'aide de Fast DDL.

Bonnes pratiques avec Amazon Aurora MySQL

Cette rubrique contient des informations sur les bonnes pratiques et les options en matière d'utilisation ou de migration de données vers un cluster de bases de données Amazon Aurora MySQL. Les informations contenues dans cette rubrique résument et réitèrent certaines des lignes directrices et procédures que vous pouvez trouver dans Gestion d'un cluster de bases de données Amazon Aurora.

Table des matières

- Configuration de la fréquence à laquelle le tampon du journal est vidé
- Minimisation et résolution des blocages d'Aurora MySQL
  - Minimisation des blocages InnoDB
  - Surveillance des blocages InnoDB
- Évaluation de l'utilisation de l'instance de base de données pour Aurora MySQL à l'aide des métriques Amazon CloudWatch

Détermination de l'instance de base de données à laquelle vous êtes connecté

Pour déterminer à quelle instance de base de données d'un cluster de bases de données Aurora MySQL une connexion est établie, vérifiez la variable globale `innodb_read_only` représentée dans l'exemple suivant.

```
SHOW GLOBAL VARIABLES LIKE 'innodb_read_only';
```

La variable `innodb_read_only` est définie sur `0N` si vous êtes connecté à une instance de base de données de lecteur. Ce paramètre est `0FF` si vous êtes connecté à une instance DB d'écriture, telle qu'une instance principale dans un cluster provisionné.

Cette approche peut s'avérer utile si vous voulez ajouter de la logique au code de votre application pour équilibrer la charge de travail ou pour garantir qu'une opération d'écriture utilise la connexion appropriée.

Bonnes pratiques pour les performances et la mise à l'échelle d'Aurora MySQL

Vous pouvez appliquer les bonnes pratiques suivantes afin d'améliorer les performances et la capacité de mise à l'échelle de vos clusters Aurora MySQL.

Rubriques

- Utilisation de classes d'instance T pour le développement et les tests
- Optimisation des requêtes de jointure indexées Aurora MySQL avec lecture anticipée asynchrone des clés
- Optimisation des requêtes de jointure MySQL Aurora volumineuses avec des jointures de hachage

- Utilisation d'Amazon Aurora pour dimensionner les lectures de votre base de données MySQL
- Optimisation des opérations d'horodatage
- Erreurs de débordement des identifiants d'index virtuels

Utilisation de classes d'instance T pour le développement et les tests

Les instances Amazon Aurora MySQL qui utilisent les classes d'instance de base de données `db.t2`, `db.t3` ou `db.t4g` sont particulièrement bien adaptées aux applications qui ne peuvent pas prendre en charge une charge de travail élevée de façon prolongée. Les instances T sont conçues pour offrir des performances de base modérées et la possibilité d'émettre en rafale pour atteindre des performances nettement supérieures si votre charge de travail l'exige. Elles sont prévues pour des charges de travail qui n'utilisent pas souvent ou de manière continue l'intégralité de l'UC, mais qui ont parfois besoin d'émettre en rafale. Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à la production. Pour plus d'informations sur les classes d'instance T, consultez Instances à performances extensibles.

Si votre cluster Aurora est supérieur à 40 To, n'utilisez pas les classes d'instance T. Lorsque votre base de données contient un volume important de données, la surcharge de mémoire pour la gestion des objets du schéma peut dépasser la capacité d'une instance T.

N'activez pas le schéma de performance MySQL sur des instances T Amazon Aurora MySQL. S'il est activé, l'instance risque de manquer de mémoire.

> **Tip**
>
> Si votre base de données est parfois inactive mais qu'à d'autres moments, elle gère une charge de travail importante, vous pouvez utiliser Aurora Serverless v2 comme alternative aux instances T. Avec Aurora Serverless v2, vous définissez une plage de capacité et Aurora augmente ou réduit automatiquement votre base de données en fonction de la charge de travail actuelle. Pour plus de détails sur l'utilisation, consultez Utiliser Aurora Serverless v2. Pour connaître les versions du moteur de base de données que vous pouvez utiliser avec Aurora Serverless v2, consultez Exigences et limites relatives à Aurora Serverless v2.

Lorsque vous utilisez une instance T en tant qu'instance de base de données dans un cluster de bases de données Aurora MySQL, nous vous recommandons la procédure suivante :

- Utilisez la même classe d'instance de base de données pour toutes les instances dans votre cluster de bases de données. Par exemple, si vous utilisez `db.t2.medium` pour votre instance d'enregistreur, nous vous recommandons d'utiliser `db.t2.medium` pour vos instances de lecteur également.
- N'ajustez pas les paramètres de configuration liés à la mémoire, tels que `innodb_buffer_pool_size`. Aurora utilise un ensemble hautement réglé de valeurs par défaut pour les tampons de mémoire sur les instances T. Ces valeurs par défaut spéciales sont nécessaires pour que Aurora s'exécute sur des instances limitées en mémoire. Si vous modifiez des paramètres liés à la mémoire sur une instance T, vous risquez beaucoup plus de rencontrer des conditions de mémoire insuffisante, même si votre modification vise à augmenter la taille de la mémoire tampon.
- Surveillez votre solde de crédits UC (`CPUCreditBalance`) pour vous assurer qu'il est à un niveau viable. Autrement dit, les crédits UC sont accumulés au même rythme qu'ils sont utilisés.

  Lorsque vous avez épuisé les crédits UC pour une instance, vous voyez une baisse immédiate de l'UC disponible et une augmentation de la latence de lecture et d'écriture de l'instance. Cela se traduit par une diminution drastique des performances globales de l'instance.

  Si votre solde de crédits CPU n'est pas à un niveau viable, nous vous conseillons de modifier votre instance de base de données pour utiliser l'une des classes d'instance de base de données R prises en charge (dimensionnement du calcul).

  Pour obtenir plus d'informations sur les métriques de supervision, consultez Affichage des métriques dans la console Amazon RDS.
- Surveillez le retard de réplica (`AuroraReplicaLag`) entre l'instance d'enregistreur et les instances de lecteur.

  Si une instance de lecteur manque de crédits CPU avant l'instance d'enregistreur, le décalage qui en résulte peut entraîner le redémarrage fréquent de l'instance de lecteur. Ceci est commun lorsqu'une application a une lourde charge d'opérations de lecture répartie entre les instances de lecteur, au même moment où l'instance d'enregistreur a une charge minimale d'opérations d'écriture.

  Si vous notez une augmentation soutenue du décalage de réplica, assurez-vous que votre solde de crédits CPU pour les instances de lecteur de votre cluster de bases de données n'est pas épuisé.

Si votre solde de crédits CPU n'est pas à un niveau viable, nous vous conseillons de modifier votre instance de base de données pour utiliser l'une des classes d'instance de base de données R prises en charge (dimensionnement du calcul).

- Maintenez le nombre d'insertions par transaction sous 1 million pour les clusters de bases de données dont la journalisation binaire est activée.

  Si le groupe de paramètres de cluster de bases de données de votre cluster de bases de données a le paramètre `binlog_format` défini sur une valeur autre que `OFF`, ce cluster peut connaître des conditions de mémoire insuffisante s'il reçoit des transactions qui contiennent plus de 1 million de lignes à insérer. Vous pouvez surveiller la mesure de mémoire libérable (`FreeableMemory`) pour déterminer si votre cluster de bases de données est à cours de mémoire disponible. Vous pouvez ensuite vérifier la mesure des opérations d'écriture (`VolumeWriteIOPS`) pour savoir si une instance de dispositif d'écriture reçoit une lourde charge d'opérations d'écriture. Si tel est le cas, nous vous recommandons de mettre à jour votre application afin de limiter le nombre d'insertions dans une opération à moins de 1 million. Il est également possible de modifier votre instance pour utiliser l'une des classes d'instance de base de données R prises en charge (dimensionnement du calcul).

Optimisation des requêtes de jointure indexées Aurora MySQL avec lecture anticipée asynchrone des clés

Aurora MySQL peut utiliser la fonctionnalité de lecture anticipée asynchrone des clés (AKP) pour améliorer les performances des requêtes qui joignent des tables par le biais des index. Cette fonction améliore les performances en anticipant les lignes nécessaires à l'exécution des requêtes dans lesquelles une requête JOIN exige l'utilisation de l'algorithme Join d'accès par lots aux clés (Batched Key Access ou BKA) et des fonctions d'optimisation de la lecture multiplage (Multi-Range Read ou MRR). Pour plus d'informations sur BKA et MRR, consultez Block Nested-Loop and Batched Key Access Joins et Multi-Range Read Optimization dans la documentation MySQL.

Pour profiter de la fonction AKP, une requête doit utiliser à la fois BKA et MRR. Une telle requête se produit normalement lorsque la clause JOIN d'une requête utilise un index secondaire, mais nécessite également quelques colonnes pour l'index principal. Par exemple, vous pouvez utiliser AKP lorsque la clause JOIN représente une équijointure sur les valeurs d'index entre une petite table externe et une grande table interne, et que l'index de la grande table est très sélectif. AKP fonctionne en association avec BKA et MRR pour procéder à une recherche d'index secondaire à principal pendant l'évaluation de la clause JOIN. AKP identifie les lignes nécessaires pour exécuter la requête

pendant l'évaluation de la clause JOIN. Elle utilise ensuite un thread d'arrière-plan pour charger de manière asynchrone des pages contenant ces lignes en mémoire avant d'exécuter la requête.

La lecture anticipée asynchrone des clés (AKP) est disponible pour Aurora MySQL versions 2.10 et ultérieures et version 3. Pour plus d'informations sur les versions d'Aurora MySQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL.

Activation de la lecture anticipée asynchrone des clés

Vous pouvez activer la fonction AKP en paramétrant `aurora_use_key_prefetch`, une variable de serveur MySQL, sur `on`. Par défaut, cette valeur indique `on`. Néanmoins, l'AKP ne peut pas être activée tant que l'algorithme de jointure BKA n'a pas été activé et que la fonctionnalité MRR basée sur le coût n'a pas été désactivée. Pour cela, vous devez spécifier les valeurs suivantes pour `optimizer_switch`, une variable du serveur MySQL :

- Définissez `batched_key_access` sur `on`. Cette valeur contrôle l'utilisation de l'algorithme Join BKA. Par défaut, cette valeur indique `off`.
- Définissez `mrr_cost_based` sur `off`. Cette valeur contrôle l'utilisation de la fonctionnalité MRR basée sur le coût. Par défaut, cette valeur indique `on`.

Actuellement, vous pouvez uniquement configurer ces valeurs au niveau de la session. L'exemple suivant illustre la configuration de ces valeurs de manière à activer AKP pour la session en cours en exécutant les instructions SET.

```
mysql> set @@session.aurora_use_key_prefetch=on;
mysql> set @@session.optimizer_switch='batched_key_access=on,mrr_cost_based=off';
```

De la même manière, vous pouvez utiliser des instructions SET pour désactiver la fonction AKP et l'algorithme Join BKA, et réactiver la fonctionnalité MRR basée sur le coût pour la session actuelle, comme indiqué dans l'exemple suivant.

```
mysql> set @@session.aurora_use_key_prefetch=off;
mysql> set @@session.optimizer_switch='batched_key_access=off,mrr_cost_based=on';
```

Pour plus d'informations sur les commutateurs d'optimiseur batched_key_access et mrr_cost_based, consultez Switchable Optimizations dans la documentation MySQL.

Optimisation des requêtes pour la lecture anticipée asynchrone des clés

Vous pouvez confirmer si une requête doit pouvoir profiter des avantages de la fonction AKP. Pour cela, utilisez l'instruction `EXPLAIN` afin de profiler la requête avant de l'exécuter. L'instruction `EXPLAIN` fournit des informations sur le plan d'exécution à utiliser pour une requête déterminée.

Dans la sortie pour l'instruction `EXPLAIN`, la colonne `Extra` décrit les informations supplémentaires comprises avec le plan d'exécution. Si la fonction AKP s'applique à une table utilisée dans la requête, cette colonne inclut l'une des valeurs suivantes :

- `Using Key Prefetching`
- `Using join buffer (Batched Key Access with Key Prefetching)`

L'exemple suivant présente l'utilisation de l'instruction `EXPLAIN` pour visualiser le plan d'exécution d'une requête qui peut bénéficier d'AKP.

```
mysql> explain select sql_no_cache
    ->     ps_partkey,
    ->     sum(ps_supplycost * ps_availqty) as value
    -> from
    ->     partsupp,
    ->     supplier,
    ->     nation
    -> where
    ->     ps_suppkey = s_suppkey
    ->     and s_nationkey = n_nationkey
    ->     and n_name = 'ETHIOPIA'
    -> group by
    ->     ps_partkey having
    ->         sum(ps_supplycost * ps_availqty) > (
    ->             select
    ->                 sum(ps_supplycost * ps_availqty) * 0.0000003333
    ->             from
    ->                 partsupp,
    ->                 supplier,
    ->                 nation
    ->             where
    ->                 ps_suppkey = s_suppkey
    ->                 and s_nationkey = n_nationkey
    ->                 and n_name = 'ETHIOPIA'
    ->         )
```

```
    -> order by
    ->     value desc;
+----+-------------+----------+------+-----------------------+--------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
| id | select_type | table    | type | possible_keys         | key           | key_len
 | ref                               | rows | filtered | Extra
                            |
+----+-------------+----------+------+-----------------------+--------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
|  1 | PRIMARY     | nation   | ALL  | PRIMARY               | NULL          | NULL
 | NULL                              |   25 |   100.00 | Using where; Using temporary;
 Using filesort                 |
|  1 | PRIMARY     | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5
 | dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                            |
|  1 | PRIMARY     | partsupp | ref  | i_ps_suppkey          | i_ps_suppkey  | 4
 | dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key
 Access with Key Prefetching) |
|  2 | SUBQUERY    | nation   | ALL  | PRIMARY               | NULL          | NULL
 | NULL                              |   25 |   100.00 | Using where
                            |
|  2 | SUBQUERY    | supplier | ref  | PRIMARY,i_s_nationkey | i_s_nationkey | 5
 | dbt3_scale_10.nation.n_nationkey | 2057 |   100.00 | Using index
                            |
|  2 | SUBQUERY    | partsupp | ref  | i_ps_suppkey          | i_ps_suppkey  | 4
 | dbt3_scale_10.supplier.s_suppkey |   42 |   100.00 | Using join buffer (Batched Key
 Access with Key Prefetching) |
+----+-------------+----------+------+-----------------------+--------------
+---------+-----------------------------------+------+----------
+-------------------------------------------------------------+
6 rows in set, 1 warning (0.00 sec)
```

Pour plus d'informations sur le format de sortie EXPLAIN, consultez Format de sortie EXPLAIN étendu (langue française non garantie) dans la documentation MySQL.

Optimisation des requêtes de jointure MySQL Aurora volumineuses avec des jointures de hachage

La jointure par hachage peut améliorer les performances de requêtes lorsque vous devez joindre une grande quantité de données au moyen d'une équijointure. Vous pouvez activer les jointures par hachage pour Aurora MySQL.

Une colonne de jointure par hachage peut être une expression complexe. Dans une colonne de jointure par hachage, vous pouvez effectuer des comparaisons dans les types de données des manières suivantes :

- Vous pouvez comparer n'importe quoi dans la catégorie des types de données numériques précises, tels que `int`, `bigint`, `numeric` et `bit`.
- Vous pouvez comparer n'importe quoi dans la catégorie des types de données numériques approximatives, tels que `float` et `double`.
- Vous pouvez comparer des éléments dans des types de chaînes si ces types de chaînes ont le même jeu de caractères et le même classement.
- Vous pouvez comparer des éléments avec des types de données de date et d'horodatage si les types sont identiques.

> **Note**
>
> Vous ne pouvez pas comparer les types de données de différentes catégories.

Les restrictions suivantes s'appliquent aux jointures par hachage pour Aurora MySQL :

- Les jointures externes gauche-droite ne sont pas prises en charge pour Aurora MySQL version 2, mais elles sont prises en charge pour la version 3.
- Les semi-jointures telles que les sous-requêtes ne sont pas prises en charge, sauf si les sous-requêtes sont matérialisées en premier.
- Les mises à jour et les suppressions sur plusieurs tables ne sont pas prises en charge.

  > **Note**
  >
  > Les mises à jour et les suppressions à table unique sont prises en charge.

- Les colonnes de types de données spatiales et BLOB ne peuvent pas constituer de colonnes de jointure dans une jointure par hachage.

Activation des jointures par hachage

Pour activer les jointures par hachage :

- Aurora MySQL version 2 – Définissez le paramètre de base de données ou le paramètre de cluster de bases de données `aurora_disable_hash_join` sur `0`. La désactivation de `aurora_disable_hash_join` définit `optimizer_switch` sur la valeur `hash_join=on`.
- Aurora MySQL version 3 – Définissez le paramètre du serveur MySQL `optimizer_switch` sur `block_nested_loop=on`.

Les jointures par hachage sont activées par défaut dans Aurora MySQL version 3 et désactivées par défaut dans Aurora MySQL version 2. L'exemple suivant montre comment activer les jointures par hachage pour Aurora MySQL version 3. Vous pouvez commencer par publier l'instruction `select @@optimizer_switch` pour voir les autres paramètres présents dans la chaîne de paramètre `SET`. La mise à jour d'un paramètre du paramètre `optimizer_switch` n'efface ni ne modifie les autres paramètres.

```
mysql> SET optimizer_switch='block_nested_loop=on';
```

> **Note**
>
> Pour Aurora MySQL Version 3, la prise en charge de la jointure par hachage est disponible dans toutes les versions mineures et est activée par défaut.
> Pour Aurora MySQL version 2, la prise en charge des jointures par hachage est disponible dans toutes les versions mineures. Dans Aurora MySQL version 2, la fonction de jointure par hachage est toujours contrôlée par la valeur `aurora_disable_hash_join`.

Avec ce paramètre, l'optimiseur choisit d'utiliser la jointure par hachage sur la base du coût, des caractéristiques de requête et de la disponibilité des ressources. Si l'estimation de coût est incorrecte, vous pouvez forcer l'optimiseur à choisir une jointure par hachage. Il suffit pour cela de paramétrer `hash_join_cost_based`, une variable de serveur MySQL, sur `off`. L'exemple suivant montre comment forcer l'optimiseur à choisir une jointure par hachage.

```
mysql> SET optimizer_switch='hash_join_cost_based=off';
```

> **Note**
>
> Ce paramètre remplace les décisions de l'optimiseur basé sur les coûts. Bien que ce paramètre puisse être utile à des fins de test et de développement, nous vous recommandons de ne pas l'utiliser en production.

Optimisation des requêtes pour les jointures par hachage

Pour savoir si une requête peut tirer parti d'une jointure par hachage, utilisez l'instruction `EXPLAIN` pour profiler la requête en premier. L'instruction `EXPLAIN` fournit des informations sur le plan d'exécution à utiliser pour une requête déterminée.

Dans la sortie pour l'instruction `EXPLAIN`, la colonne `Extra` décrit les informations supplémentaires comprises avec le plan d'exécution. Si une jointure par hachage s'applique aux tables utilisées dans la requête, cette colonne inclut des valeurs similaires aux suivantes :

- `Using where; Using join buffer (Hash Join Outer table` *`table1_name`*`)`
- `Using where; Using join buffer (Hash Join Inner table` *`table2_name`*`)`

L'exemple suivant présente l'utilisation d'EXPLAIN pour visualiser le plan d'exécution d'une requête de jointure par hachage.

```
mysql> explain SELECT sql_no_cache * FROM hj_small, hj_big, hj_big2
    ->     WHERE hj_small.col1 = hj_big.col1 and hj_big.col1=hj_big2.col1 ORDER BY 1;
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
| id | select_type | table    | type | possible_keys | key  | key_len | ref  | rows |
 Extra                                                          |
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
|  1 | SIMPLE      | hj_small | ALL  | NULL          | NULL | NULL    | NULL |    6 |
 Using temporary; Using filesort                                |
|  1 | SIMPLE      | hj_big   | ALL  | NULL          | NULL | NULL    | NULL |   10 |
 Using where; Using join buffer (Hash Join Outer table hj_big)  |
|  1 | SIMPLE      | hj_big2  | ALL  | NULL          | NULL | NULL    | NULL |   15 |
 Using where; Using join buffer (Hash Join Inner table hj_big2) |
+----+-------------+----------+------+---------------+------+---------+------+------
+----------------------------------------------------------------+
```

```
3 rows in set (0.04 sec)
```

Dans la sortie, `Hash Join Inner table` représente la table utilisée pour construire la table de hachage, et `Hash Join Outer table` représente la table employée pour sonder la table de hachage.

Pour plus d'informations sur le format de sortie étendu d'`EXPLAIN`, consultez Extended EXPLAIN Output Format dans la documentation du produit MySQL.

Dans les versions 2.08 et ultérieures d'Aurora MySQL, vous pouvez utiliser des indicateurs SQL pour déterminer si une requête utilise ou non la jointure de hachage, et quelles tables utiliser pour les côtés construction et sonde de la jointure. Pour en savoir plus, consultez Indicateurs Aurora MySQL.

Utilisation d'Amazon Aurora pour dimensionner les lectures de votre base de données MySQL

Vous pouvez utiliser Amazon Aurora avec votre instance de base de données MySQL pour tirer parti des capacités de mise à l'échelle en lecture d'Amazon Aurora et développer la charge de travail en lecture de votre instance de base de données MySQL. Pour utiliser Aurora afin de mettre à l'échelle en lecture votre instance de base de données MySQL, créez un cluster de bases de données Aurora MySQL et faites-en un réplica en lecture de votre instance de base de données MySQL. Ensuite, connectez-vous au cluster Aurora MySQL pour traiter les requêtes en lecture. La base de données source peut être une instance de base de données RDS for MySQL ou une base de données MySQL s'exécutant en dehors de Amazon RDS. Pour plus d'informations, consultez Mise à l'échelle des lectures pour votre base de données MySQL avec Amazon Aurora.

Optimisation des opérations d'horodatage

Lorsque la valeur de la variable système `time_zone` est définie sur `SYSTEM`, chaque appel de fonction MySQL qui nécessite un calcul de fuseau horaire effectue un appel à la bibliothèque système. Lorsque vous exécutez des instructions SQL qui renvoient ou modifient de telles valeurs `TIMESTAMP` avec un taux de simultanéité élevé, vous pouvez constater une augmentation de la latence, de la contention des verrou et de l'utilisation du processeur. Pour plus d'informations, consultez time_zone dans la documentation MySQL.

Pour éviter ce comportement, nous vous recommandons de modifier la valeur du paramètre `time_zone` de cluster de bases de données sur `UTC`. Pour plus d'informations, consultez Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

Le paramètre `time_zone` est dynamique (il ne nécessite pas de redémarrage du serveur de base de données), mais la nouvelle valeur est utilisée uniquement pour les nouvelles connexions. Pour vous assurer que toutes les connexions sont mises à jour pour utiliser la nouvelle valeur `time_zone`, nous vous recommandons de recycler les connexions de votre application après avoir mis à jour le paramètre du cluster de bases de données.

Erreurs de débordement des identifiants d'index virtuels

Aurora MySQL limite les valeurs des identifiants d'index virtuels à 8 bits pour éviter un problème causé par le format d'annulation dans MySQL. Si un index dépasse cette limite, il se peut que votre cluster ne soit pas disponible. Lorsqu'un index approche la limite d'ID d'index virtuel ou lorsque vous tentez de créer un index supérieur à la limite d'ID d'index virtuel, RDS peut générer un code d'erreur `63955` ou un code d'avertissement `63955`. Pour corriger une erreur liée à une limite d'ID d'index virtuel, nous vous recommandons de recréer votre base de données avec une opération de vidage et de restauration logiques.

Pour plus d'informations sur le vidage et la restauration logiques pour Amazon Aurora MySQL, consultez Migration de très grandes bases de données vers Amazon Aurora MySQL à l'aide de MyDumper et MyLoader. Pour plus d'informations sur l'accès aux journaux d'erreurs dans Amazon Aurora, consultez Surveillance des fichiers journaux Amazon Aurora.

Bonnes pratiques pour la haute disponibilité Aurora MySQL

Vous pouvez appliquer les bonnes pratiques suivantes afin d'améliorer la disponibilité de vos clusters Aurora MySQL.

Rubriques

- Utilisation d'Amazon Aurora pour la reprise après sinistre avec vos bases de données MySQL
- Migration depuis MySQL vers Amazon Aurora MySQL avec une interruption réduite
- Contournement des performances lentes, du redémarrage automatique et du basculement pour les instances de base de données Aurora MySQL

Utilisation d'Amazon Aurora pour la reprise après sinistre avec vos bases de données MySQL

Vous pouvez utiliser Amazon Aurora avec votre instance de base de données MySQL pour créer une sauvegarde hors site pour la reprise après sinistre. Pour utiliser Aurora pour la reprise après

sinistre de votre instance de base de données MySQL, créez un cluster de bases de données Amazon Aurora et faites-en un réplica en lecture de votre instance de base de données MySQL. Cela s'applique à une instance de base de données RDS for MySQL ou à une base de données MySQL s'exécutant en dehors de Amazon RDS.

> **Important**
>
> Lorsque vous configurez la réplication entre une instance de base de données MySQL et un cluster de bases de données Amazon Aurora MySQL, vous devez surveiller la réplication pour vous assurer qu'elle reste saine et la réparer si nécessaire.

Pour obtenir des instructions sur la façon de créer un cluster de bases de données Amazon Aurora MySQL et d'en faire un réplica en lecture de votre instance de base de données MySQL, suivez la procédure décrite dans Utilisation d'Amazon Aurora pour dimensionner les lectures de votre base de données MySQL.

Pour plus d'informations sur les modèles de reprise après sinistre, consultez How to choose the best disaster recovery option for your Amazon Aurora MySQL cluster (Comment choisir la meilleure option de reprise après sinistre pour votre cluster Amazon Aurora MySQL).

Migration depuis MySQL vers Amazon Aurora MySQL avec une interruption réduite

Lors de l'importation de données depuis une base de données MySQL prenant en charge une application active vers un cluster de bases de données Amazon Aurora MySQL, vous pouvez souhaiter réduire la durée d'interruption du service de vos données pendant la migration. Pour ce faire, vous pouvez utiliser la procédure documentée dans Importation de données vers une instance de base de données Amazon RDS for MySQL avec une durée d'indisponibilité réduite dans le Guide de l'utilisateur Amazon Relational Database Service. Cette procédure peut s'avérer tout spécialement utile si vous travaillez avec une base de données très volumineuse. Elle vous permet de réduire le coût de l'importation en diminuant la quantité de données transmises à AWS via le réseau.

La procédure répertorie les étapes à suivre pour transférer une copie des données de votre base de données vers une EC2 instance Amazon et importer les données dans une nouvelle instance de base de données RDS for MySQL. Comme Amazon Aurora est compatible avec MySQL, vous pouvez utiliser à la place un cluster de bases de données Amazon Aurora pour l'instance de base de données Amazon RDS MySQL cible.

Contournement des performances lentes, du redémarrage automatique et du basculement pour les instances de base de données Aurora MySQL

Si vous exécutez une charge de travail importante ou des charges de travail qui dépassent les ressources allouées à votre instance de base de données, vous pouvez épuiser les ressources sur lesquelles vous exécutez votre application et votre base de données Aurora. Pour obtenir des statistiques sur votre instance de base de données, telles que l'utilisation du processeur, l'utilisation de la mémoire et le nombre de connexions de base de données utilisées, vous pouvez vous référer aux métriques fournies par Amazon CloudWatch, Performance Insights et Enhanced Monitoring. Pour plus d'informations sur la surveillance de votre instance de base de données, consultez Surveillance des métriques d'un cluster de bases de données Amazon Aurora.

Si votre charge de travail épuise les ressources que vous utilisez, votre instance de base de données peut ralentir, redémarrer ou même basculer vers une autre instance de base de données. Pour éviter cela, surveillez l'utilisation de vos ressources, examinez la charge de travail exécutée sur votre instance de base de données et effectuez des optimisations si nécessaire. Si les optimisations n'améliorent pas les métriques de l'instance et n'atténuent pas l'épuisement des ressources, envisagez d'augmenter votre instance de base de données avant d'atteindre ses limites. Pour plus d'informations sur les classes d'instance de base de données disponibles et leurs spécifications, consultez Classes d'instance de base de données Amazon Aurora.

Recommandations pour les fonctionnalités MySQL dans Aurora MySQL

Les fonctions suivantes sont disponibles dans Aurora MySQL pour la compatibilité MySQL. Cependant, ils présentent des problèmes de performance, de capacité de mise à l'échelle, de stabilité ou de compatibilité dans l'environnement Aurora. Nous vous recommandons donc de suivre certaines directives dans l'utilisation de ces fonctionnalités. Par exemple, nous vous recommandons de ne pas utiliser certaines fonctions pour les déploiements Aurora en production.

Rubriques

- Utilisation de la réplication multithread dans Aurora MySQL
- Appeler des fonctions AWS Lambda à l'aide des fonctions MySQL natives
- Éviter les transactions XA avec Amazon Aurora MySQL
- Maintenir les clés étrangères activées pendant les instructions DML
- Configuration de la fréquence à laquelle le tampon du journal est vidé
- Minimisation et résolution des blocages d'Aurora MySQL

Utilisation de la réplication multithread dans Aurora MySQL

Avec la réplication de journaux binaires multithreads, un thread SQL lit les événements du journal de relais et les met en file d'attente pour que les threads de travail SQL s'appliquent. Les threads de travail SQL sont gérés par un thread coordinateur. Si cela est possible, les événements du journal binaire sont appliqués en parallèle.

La réplication multithread est prise en charge dans Aurora MySQL version 3 et dans Aurora MySQL version 2.12.1 ou ultérieure.

Pour les versions d'Aurora MySQL inférieures à 3.04, Aurora utilise la réplication à thread unique par défaut lorsqu'un cluster de bases de données Aurora MySQL est utilisé comme réplica en lecture pour la réplication de journaux binaires.

Les versions antérieures d'Aurora MySQL 2 ont hérité de plusieurs problèmes concernant la réplication multithread à partir de MySQL Community Edition. Pour ces versions, nous vous recommandons de ne pas utiliser la réplication multithread en production.

Si vous utilisez la réplication multithread, nous vous recommandons de la tester pleinement.

Pour plus d'informations sur l'utilisation de la réplication dans Amazon Aurora, consultez Réplication avec Amazon Aurora. Pour plus d'informations sur la réplication multithread dans Aurora MySQL, consultez Réplication des journaux binaires multithread.

Appeler des fonctions AWS Lambda à l'aide des fonctions MySQL natives

Nous vous recommandons d'utiliser les fonctions MySQL natives `lambda_sync` et `lambda_async` pour invoquer des fonctions Lambda.

Si vous employez la procédure obsolète `mysql.lambda_async`, nous vous recommandons d'encapsuler les appels de la procédure `mysql.lambda_async` dans une procédure stockée. Vous pouvez appeler cette procédure stockée à partir de différentes sources, telles que des déclencheurs ou du code client. Cette approche contribue à éviter les problèmes de discordance d'impédance et permet aux programmeurs de base de données d'appeler plus facilement les fonctions Lambda.

Pour plus d'informations sur l'appel de fonctions Lambda à partir d'Amazon Aurora, consultez Appel d'une fonction Lambda à partir d'un cluster de bases de données Amazon Aurora MySQL.

Éviter les transactions XA avec Amazon Aurora MySQL

Nous vous déconseillons d'utiliser des transactions eXtended Architecture (XA) avec Aurora MySQL, car elles peuvent allonger les temps de récupération si l'architecture XA était à l'état `PREPARED`.

Si vous devez utiliser des transactions XA avec Aurora MySQL, observez les bonnes pratiques suivantes :

- ne laissez pas de transaction XA ouverte en état `PREPARED` ;
- gardez les transactions XA aussi petites que possible.

Pour plus d'informations sur l'utilisation des transactions XA avec MySQL, consultez XA Transactions dans la documentation MySQL.

Maintenir les clés étrangères activées pendant les instructions DML

Nous vous recommandons vivement de ne pas exécuter d'instruction de langage de définition de données (Data Definition Language, DDL) lorsque la variable `foreign_key_checks` a pour valeur `0` (désactivé).

Si vous avez besoin d'insérer ou de mettre à jour des lignes qui requièrent une violation transitoire des clés étrangères, procédez comme suit :

1. Définissez `foreign_key_checks` sur `0`.
2. Apportez vos modifications de langage de manipulation de données (DML).
3. Assurez-vous que les modifications que vous avez apportées ne vont à l'encontre d'aucune contrainte de clé étrangère.
4. Affectez à `foreign_key_checks` la valeur `1` (activé).

De plus, suivez ces autres bonnes pratiques relatives aux contraintes de clé étrangère :

- Assurez-vous que vos applications clientes n'affectent pas la valeur `foreign_key_checks` à la variable `0` dans le cadre de la variable `init_connect`.
- Si une restauration à partir d'une sauvegarde logique telle que `mysqldump` échoue ou est incomplète, assurez-vous que la valeur `foreign_key_checks` soit affectée à `1` avant de commencer toute autre opération dans la même session. Une sauvegarde logique affecte la valeur `foreign_key_checks` à `0` lorsqu'elle commence.

Configuration de la fréquence à laquelle le tampon du journal est vidé

Dans MySQL Community Edition, pour rendre les transactions durables, le tampon du journal InnoDB doit être vidé vers un stockage durable. Vous utilisez le paramètre

`innodb_flush_log_at_trx_commit` pour configurer la fréquence à laquelle le tampon du journal est vidé vers un disque.

Lorsque vous définissez le paramètre `innodb_flush_log_at_trx_commit` sur la valeur par défaut de 1, le tampon du journal est vidé à chaque validation de transaction. Ce paramètre permet de maintenir la conformité ACID de la base de données. Nous vous recommandons de conserver le paramètre par défaut sur 1.

Le remplacement d'`innodb_flush_log_at_trx_commit` par une valeur autre que celle par défaut peut contribuer à réduire la latence du langage de manipulation des données (DML), mais compromet la durabilité des enregistrements du journal. Ce manque de durabilité rend la base de données ACID non conforme. Nous recommandons que vos bases de données soient conformes à ACID pour éviter le risque de perte de données en cas de redémarrage du serveur. Pour plus d'informations sur ce paramètre, consultez innodb_flush_log_at_trx_commit dans la documentation MySQL.

Dans Aurora MySQL, le traitement des fichiers de reprise est déchargé vers la couche de stockage, de sorte qu'aucun vidage des fichiers journaux ne se produit sur l'instance de base de données. Lorsqu'une écriture est émise, les journaux de reprise sont envoyés depuis l'instance de base de données d'écriture directement vers le volume de cluster Aurora. Les seules écritures qui transitent par le réseau sont les enregistrements de journaux de reprise. Aucune page n'est jamais écrite à partir du niveau de la base de données.

Par défaut, chaque thread qui valide une transaction attend la confirmation du volume du cluster Aurora. Cette confirmation indique que cet enregistrement et tous les enregistrements de journaux de reprise précédents sont écrits et ont atteint le quorum. La conservation des enregistrements du journal et l'atteinte du quorum rendent la transaction durable, que ce soit par le biais d'une validation automatique ou d'une validation explicite. Pour plus d'informations sur l'architecture de stockage Aurora, consultez Stockage Amazon Aurora démystifié.

Aurora MySQL ne vide pas les journaux dans les fichiers de données comme le fait MySQL Community Edition. Vous pouvez toutefois utiliser le paramètre `innodb_flush_log_at_trx_commit` pour assouplir les contraintes de durabilité lors de l'écriture d'enregistrements de journaux de reprise sur le volume de cluster Aurora.

Pour Aurora MySQL version 2 :

- `innodb_flush_log_at_trx_commit` = 0 ou 2 : la base de données n'attend pas la confirmation que les enregistrements des journaux de reprise sont écrits sur le volume de cluster Aurora.

- `innodb_flush_log_at_trx_commit` = 1 : la base de données attend la confirmation que les enregistrements des journaux de reprise sont écrits sur le volume du cluster Aurora.

Pour Aurora MySQL version 3 :

- `innodb_flush_log_at_trx_commit` = 0 : la base de données n'attend pas la confirmation que les enregistrements des journaux de reprise sont écrits sur le volume de cluster Aurora.
- `innodb_flush_log_at_trx_commit` = 1 ou 2 : la base de données attend la confirmation que les enregistrements des journaux de reprise sont écrits sur le volume du cluster Aurora.

Par conséquent, pour obtenir le même comportement, autre que celui par défaut, dans Aurora MySQL version 3 que celui que vous obtiendriez avec une valeur définie sur 0 ou 2 dans Aurora MySQL version 2, définissez le paramètre sur 0.

Bien que ces paramètres puissent réduire la latence DML pour le client, ils peuvent également entraîner une perte de données en cas de basculement ou de redémarrage. Par conséquent, nous vous recommandons de conserver la valeur par défaut de 1 pour le paramètre `innodb_flush_log_at_trx_commit`.

Bien que des pertes de données puissent se produire à la fois dans MySQL Community Edition et Aurora MySQL, le comportement diffère dans chaque base de données en raison de leurs architectures différentes. Ces différences architecturales peuvent entraîner des pertes de données à des degrés divers. Pour vous assurer que votre base de données est conforme à la norme ACID, définissez toujours une valeur 1 pour `innodb_flush_log_at_trx_commit`.

> **Note**
>
> Dans Aurora MySQL version 3, avant de pouvoir remplacer `innodb_flush_log_at_trx_commit` par une valeur autre que 1, vous devez d'abord remplacer la valeur d'`innodb_trx_commit_allow_data_loss` par 1. Ce faisant, vous reconnaissez le risque de perte de données.

Minimisation et résolution des blocages d'Aurora MySQL

Les utilisateurs exécutant des charges de travail qui rencontrent régulièrement des violations de contraintes sur des index secondaires uniques ou des clés étrangères lorsqu'ils modifient simultanément des enregistrements sur la même page de données, peuvent être confrontés à

des blocages et à des délais d'attente plus longs. Ces blocages et délais d'attente sont dus à une correction de bogue de MySQL Community Edition.

Ce correctif est inclus dans les versions 5.7.26 et ultérieures de MySQL Community Edition, et a été rétroporté aux versions 2.10.3 et ultérieures d'Aurora MySQL. Le correctif est nécessaire pour mettre en vigueur la sérialisation, en implémentant un verrouillage supplémentaire pour ces types d'opérations en langage de manipulation de données (DML) sur les modifications apportées aux enregistrements d'une table InnoDB. Ce problème a été découvert dans le cadre d'une enquête sur les problèmes de blocage introduits par une précédente correction de bogue de MySQL Community Edition.

Le correctif a modifié la gestion interne de l'annulation partielle d'une mise à jour de tuple (ligne) dans le moteur de stockage InnoDB. Les opérations qui génèrent des violations de contraintes sur des clés étrangères ou des index secondaires uniques entraînent une annulation partielle. Cela inclut, sans toutefois s'y limiter, les instructions `INSERT...ON DUPLICATE KEY UPDATE`, `REPLACE INTO`, et `INSERT IGNORE` simultanées (mises à jour/insertions).

Dans ce contexte, l'annulation partielle ne fait pas référence à l'annulation des transactions au niveau de l'application, mais plutôt à une annulation interne à InnoDB des modifications apportées à un index organisé en cluster, lorsqu'une violation de contrainte est détectée. Par exemple, une valeur de clé dupliquée est détectée lors d'une opération de mise à jour/d'insertion.

Dans une opération d'insertion normale, InnoDB crée de manière atomique des entrées d'index organisés en cluster et secondaires pour chaque index. Si InnoDB détecte une valeur dupliquée sur un index secondaire unique lors d'une opération de mise à jour/d'insertion, l'entrée insérée dans l'index organisé en cluster doit être annulée (annulation partielle) et la mise à jour doit ensuite être appliquée à la ligne dupliquée existante. Au cours de cette étape d'annulation partielle interne, InnoDB doit verrouiller chaque enregistrement considéré dans le cadre de l'opération. Le correctif garantit la sérialisation des transactions en introduisant un verrouillage supplémentaire après l'annulation partielle.

Minimisation des blocages InnoDB

Vous pouvez adopter les approches suivantes pour réduire la fréquence des blocages dans votre instance de base de données. Vous trouverez d'autres exemples dans la documentation MySQL.

1. Pour réduire les risques de blocages, validez les transactions immédiatement après avoir apporté un ensemble de modifications connexes. Vous pouvez le faire en divisant les transactions volumineuses (mises à jour de plusieurs lignes entre les validations) en transactions plus petites.

Si vous insérez des lignes par lots, essayez de réduire la taille des insertions par lots, en particulier lorsque vous utilisez les opérations de mise à jour/d'insertion mentionnées précédemment.

Pour réduire le nombre d'annulations partielles possibles, vous pouvez essayer l'une des approches suivantes :

a. Remplacez les opérations d'insertion par lots en insérant une ligne à la fois. Cela peut réduire la durée pendant laquelle les verrous sont bloqués en raison de transactions susceptibles de présenter des conflits.

b. Au lieu d'utiliser `REPLACE INTO`, réécrivez l'instruction SQL sous la forme d'une transaction à plusieurs instructions, comme suit :

```
BEGIN;
DELETE conflicting rows;
INSERT new rows;
COMMIT;
```

c. Au lieu d'utiliser `INSERT...ON DUPLICATE KEY UPDATE`, réécrivez l'instruction SQL sous la forme d'une transaction à plusieurs instructions, comme suit :

```
BEGIN;
SELECT rows that conflict on secondary indexes;
UPDATE conflicting rows;
INSERT new rows;
COMMIT;
```

2. Évitez les transactions de longue durée, actives ou inactives, qui pourraient bloquer les verrous. Cela inclut les sessions client MySQL interactives qui peuvent être ouvertes pendant une période prolongée avec une transaction non validée. Lors de l'optimisation de la taille des transactions ou de la taille des lots, l'impact peut varier en fonction d'un certain nombre de facteurs tels que la simultanéité, le nombre de doublons et la structure de la table. Toute modification doit être mise en œuvre et testée en fonction de votre charge de travail.
3. Dans certains cas, des blocages peuvent survenir lorsque deux transactions tentent d'accéder aux mêmes jeux de données, dans une ou plusieurs tables, dans des ordres différents. Pour éviter cela, vous pouvez modifier les transactions pour accéder aux données dans le même ordre, sérialisant ainsi l'accès. Par exemple, créez une file d'attente de transactions à terminer. Cette approche permet d'éviter les blocages lorsque plusieurs transactions se produisent simultanément.
4. L'ajout d'index soigneusement sélectionnés à vos tables peut améliorer la sélectivité et réduire le besoin d'accéder aux lignes, ce qui permet de réduire le verrouillage.

5. En cas de verrouillage des écarts, vous pouvez modifier le niveau d'isolement de la transaction sur `READ COMMITTED` afin que la session ou la transaction l'empêche. Pour plus d'informations sur les niveaux d'isolement d'InnoDB et leurs comportements, consultez Niveaux d'isolement des transactions dans la documentation MySQL.

> **Note**
>
> Bien que vous puissiez prendre des précautions pour réduire les risques de blocages, les blocages sont un comportement normal des bases de données et peuvent toujours se produire. Les applications doivent disposer de la logique nécessaire pour gérer les blocages lorsqu'ils se présentent. Par exemple, implémentez une logique de nouvelle tentative et de retrait dans l'application. Il est préférable de s'attaquer à la cause racine du problème, mais en cas de blocage, l'application a la possibilité d'attendre et de réessayer.

Surveillance des blocages InnoDB

Des blocages peuvent survenir dans MySQL lorsque des transactions d'application tentent de se verrouiller au niveau des tables et des lignes, ce qui entraîne une attente circulaire. Un blocage occasionnel d'InnoDB n'est pas nécessairement un problème, car le moteur de stockage InnoDB détecte immédiatement la situation et annule automatiquement l'une des transactions. Si vous rencontrez fréquemment des blocages, nous vous recommandons de revoir et de modifier votre application pour atténuer les problèmes de performances et éviter les blocages. Lorsque la détection des blocages est activée (par défaut), InnoDB détecte automatiquement les blocages de transactions et annule une ou plusieurs transactions pour sortir du blocage. InnoDB essaie de sélectionner les petites transactions à annuler, la taille d'une transaction étant déterminée par le nombre de lignes insérées, mises à jour ou supprimées.

- Instruction `SHOW ENGINE` : l'instruction `SHOW ENGINE INNODB STATUS \G` contient des informations sur le blocage le plus récent rencontré sur la base de données depuis le dernier redémarrage.
- Journal des erreurs MySQL : si vous rencontrez fréquemment des blocages lorsque la sortie de l'instruction `SHOW ENGINE` est inadéquate, vous pouvez activer le paramètre de cluster de bases de données innodb_print_all_deadlocks.

  Lorsque ce paramètre est activé, les informations relatives à tous les blocages dans les transactions utilisateur d'InnoDB sont enregistrées dans le journal des erreurs Aurora MySQL.

- Métriques Amazon CloudWatch : nous vous recommandons également de surveiller de manière proactive les blocages à l'aide de la métrique CloudWatch `Deadlocks`. Pour plus d'informations, consultez Métriques de niveau instance pour Amazon Aurora.
- Amazon CloudWatch Logs : CloudWatch Logs vous permet d'afficher des métriques, d'analyser des données de journaux et de créer des alarmes en temps réel. Pour plus d'informations, consultez Monitor errors in Amazon Aurora MySQL and Amazon RDS for MySQL using Amazon CloudWatch and send notifications using Amazon SNS (Surveiller les erreurs dans Amazon Aurora MySQL et Amazon RDS for MySQL à l'aide d'Amazon CloudWatch et envoyer des notifications via Amazon SNS).

  Lorsque `innodb_print_all_deadlocks` est activé sur CloudWatch Logs, vous pouvez configurer des alarmes qui vous avertiront lorsque le nombre de blocages dépasse un seuil donné. Pour définir un seuil, nous vous recommandons d'observer vos tendances et d'utiliser une valeur basée sur votre charge de travail normale.
- Performances Insights : lorsque vous utilisez Performance Insights, vous pouvez surveiller les métriques `innodb_deadlocks` et `innodb_lock_wait_timeout`. Pour obtenir plus d'informations sur ces métriques, consultez Compteurs non natifs pour Aurora MySQL.

Évaluation de l'utilisation de l'instance de base de données pour Aurora MySQL à l'aide des métriques Amazon CloudWatch

Vous pouvez utiliser les métriques CloudWatch pour surveiller le débit de votre instance de base de données et déterminer si votre classe d'instance fournit suffisamment de ressources à vos applications. Pour en savoir plus sur les limites des classes d'instance de base de données, consultez Spécifications matérielles pour les classes d'instance de base de données pour Aurora. Consultez les spécifications de votre classe d'instance de base de données afin de connaître les performances du réseau.

Si l'utilisation de votre instance de base de données est proche de la limite de classe d'instance, les performances peuvent commencer à ralentir. Les métriques CloudWatch peuvent confirmer cette situation afin que vous puissiez planifier une augmentation verticale manuelle vers une classe d'instance plus importante.

Combinez les valeurs des métriques CloudWatch suivantes pour déterminer si vous approchez de la limite de classes d'instance :

- NetworkThroughput : la quantité de débit réseau reçue des clients et transmise à ces derniers par chaque instance du cluster de bases de données Aurora. Cette valeur du débit n'inclut pas le trafic réseau entre les instances du cluster de bases de données et le volume de cluster.
- StorageNetworkThroughput : la quantité de débit réseau reçue du et envoyée au sous-système de stockage Aurora par chaque instance du cluster de bases de données Aurora.

Ajoutez NetworkThroughput à StorageNetworkThroughput pour découvrir la quantité de débit réseau reçue du et envoyée au sous-système de stockage Aurora par chaque instance du cluster de bases de données Aurora. La limite de classe d'instance pour votre instance doit être supérieure à la somme de ces deux métriques combinées.

Vous pouvez utiliser les métriques suivantes pour consulter des informations supplémentaires sur le trafic réseau provenant de vos applications clientes lors de l'envoi et de la réception :

- NetworkReceiveThroughput : quantité de débit réseau reçue des clients par chaque instance de base de données du cluster de bases de données Aurora MySQL. Ce débit n'inclut pas le trafic réseau entre les instances du cluster de bases de données et le volume de cluster.
- NetworkTransmitThroughput : quantité de débit réseau envoyée aux clients par chaque instance du cluster de bases de données Aurora. Ce débit n'inclut pas le trafic réseau entre les instances du cluster de bases de données et le volume de cluster.
- StorageNetworkReceiveThroughput : quantité de débit réseau reçue du sous-système de stockage Aurora par chaque instance du cluster de bases de données.
- StorageNetworkTransmitThroughput : la quantité de débit réseau envoyée au sous-système de stockage Aurora par chaque instance du cluster de bases de données.

Ajoutez toutes ces métriques pour évaluer l'utilisation de votre réseau par rapport à la limite de classe d'instance de base de données. La limite de classe d'instance doit être supérieure à la somme de ces métriques combinées.

Les limites du réseau et l'utilisation du processeur pour le stockage sont directement liés. Lorsque le débit réseau augmente, l'utilisation du processeur augmente également. La surveillance de l'utilisation du processeur et du réseau fournit des informations sur comment et pourquoi les ressources sont épuisées.

Pour contribuer à minimiser l'utilisation du réseau, vous pouvez envisager les actions suivantes :

- Utiliser une classe d'instance de base de données plus grande.

- Diviser les demandes d’écriture par lots afin de réduire le nombre total de transactions.
- Rediriger la charge de travail en lecture seule vers une instance en lecture seule.
- Supprimer tous les index inutilisés.

Résolution des problèmes de performances de base de données Amazon Aurora MySQL

Cette rubrique se concentre sur certains problèmes courants de performance des bases de données Aurora MySQL, ainsi que sur la manière de résoudre ces problèmes ou de collecter des informations pour y remédier rapidement. Nous divisons les performances des bases de données en deux catégories principales :

- Performances du serveur : l'ensemble du serveur de base de données fonctionne plus lentement.
- Performances des requêtes : l'exécution d'une ou de plusieurs requêtes prend plus de temps.

Options de surveillance AWS

Nous vous recommandons d'utiliser les options de surveillance AWS suivantes pour faciliter la résolution des problèmes :

- Amazon CloudWatch : Amazon CloudWatch surveille vos ressources AWS et les applications que vous exécutez sur AWS en temps réel. Vous pouvez utiliser CloudWatch pour recueillir et suivre les métriques, qui sont des variables que vous pouvez mesurer pour vos ressources et applications. Pour plus d'informations, consultez Qu'est-ce qu'Amazon CloudWatch ?

  Vous pouvez afficher toutes les métriques système et les informations de processus de vos instances de base de données RDS dans la AWS Management Console. Vous pouvez configurer votre cluster de bases de données Aurora MySQL afin qu'il publie les données de journaux généraux, de journaux de requêtes lentes, de journaux d'audit et de journaux d'erreurs dans un groupe de journaux dans Amazon CloudWatch Logs. Cela vous permet de visualiser les tendances, de gérer les journaux si un hôte est impacté et de créer une base de référence pour les performances « normales » afin d'identifier facilement les anomalies ou les modifications. Pour plus d'informations, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs.
- Surveillance améliorée : pour bénéficier de métriques Amazon CloudWatch supplémentaires pour une base de données Aurora MySQL, activez la surveillance améliorée. Lorsque vous créez ou modifiez un cluster de bases de données Aurora, sélectionnez Activer la surveillance améliorée. Cela permet à Aurora de publier des métriques de performance sur CloudWatch. Parmi les métriques clés disponibles, citons l'utilisation du processeur, les connexions aux bases de

données, l'utilisation du stockage et la latence des requêtes. Ces métriques peuvent vous aider à identifier les goulots d'étranglement au niveau des performances.

La quantité d'informations transférées pour une instance de base de données est directement proportionnelle à la granularité définie pour la fonction de surveillance améliorée. Plus l'intervalle de surveillance est court, plus la fréquence des rapports sur les métriques du système d'exploitation est élevée, ce qui augmente les coûts de surveillance. Pour gérer les coûts, définissez différentes granularités pour différentes instances de vos Comptes AWS. La granularité par défaut lors de la création d'une instance est de 60 secondes. Pour plus d'informations, consultez Coût de la surveillance améliorée.

- Performance Insights : vous pouvez consulter toutes les métriques relatives aux appels de base de données. Cela inclut les blocages de base de données, les temps d'attente et le nombre de lignes traitées, qui sont tous des informations que vous pouvez utiliser pour la résolution des problèmes. Lorsque vous créez ou modifiez un cluster de bases de données Aurora, sélectionnez Activer Performance Insights. Par défaut, Performance Insights dispose d'une période de conservation des données de 7 jours. Toutefois, cette période peut être personnalisée pour permettre une analyse des tendances de performance à long terme. Pour une période de conservation supérieure à 7 jours, vous devez passer à l'offre payante. Pour plus d'informations sur les coûts, consultez Tarification de Performance Insights. Vous pouvez définir la période de conservation des données pour chaque instance de base de données Aurora séparément. Pour plus d'informations, consultez Surveillance de la charge de la base de données avec Performance Insights sur .

Causes fréquentes des problèmes de performances des bases de données Aurora MySQL

Vous pouvez suivre les étapes ci-dessous pour résoudre les problèmes de performances dans votre base de données Aurora MySQL. Ces étapes figurent dans l'ordre le plus logique à suivre pour identifier la cause du problème, mais elles ne sont pas censées être linéaires. Une seule découverte peut franchir plusieurs étapes et ouvrir ainsi la voie à une série de pistes d'investigation.

1. Charge de travail : comprenez la charge de travail de votre base de données.
2. Journalisation : passez en revue tous les journaux de base de données.
3. Connexions de base de données : assurez-vous que les connexions entre vos applications et votre base de données sont fiables.
4. Performances des requêtes : examinez vos plans d'exécution des requêtes pour déterminer s'ils ont changé. Toute modification du code peut entraîner la modification des plans.

Résolution des problèmes de charge de travail pour les bases de données Aurora MySQL

La charge de travail des bases de données peut être considérée sous forme de lectures et d'écritures. En comprenant la charge de travail « normale » des bases de données, vous pouvez ajuster les requêtes et le serveur de base de données en fonction de l'évolution de la demande. Les performances peuvent changer pour différentes raisons. La première étape consiste donc à comprendre ce qui a changé.

- Y a-t-il eu une mise à niveau de version majeure ou mineure ?

  Une mise à niveau de version majeure inclut des modifications du code du moteur, en particulier dans l'optimiseur, qui peuvent avoir un impact sur le plan d'exécution des requêtes. Lorsque vous mettez à niveau des versions de base de données, notamment des versions majeures, il est très important d'analyser la charge de travail de la base de données et de l'ajuster en conséquence. L'ajustement peut impliquer l'optimisation et la réécriture des requêtes, ou l'ajout et la mise à jour de paramètres, en fonction des résultats des tests. Comprendre la cause de l'impact vous permettra de commencer à vous concentrer sur ce domaine spécifique.

  Pour plus d'informations, consultez Nouveautés de MySQL 8.0 et Ajout, abandon ou suppression des variables et options de serveur et d'état dans MySQL 8.0 dans la documentation MySQL, et Comparaison d'Aurora MySQL version 2 et Aurora MySQL version 3.
- Y a-t-il eu une augmentation du nombre de données traitées (nombre de lignes) ?
- D'autres requêtes sont-elles exécutées simultanément ?
- Y a-t-il des modifications du schéma ou de la base de données ?
- Y a-t-il eu des défauts ou des corrections de code ?

Table des matières

- Exemple 2 : pics de mémoire transitoires
- Exemple 3 : la mémoire libérable diminue continuellement et n'est pas récupérée
- Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL

Métriques de l'hôte de l'instance

Surveillez les métriques de l'hôte de l'instance telles que le processeur, la mémoire et l'activité réseau afin de déterminer si la charge de travail a changé. Deux concepts principaux permettent de comprendre l'évolution de la charge de travail :

- Utilisation : utilisation d'un périphérique, tel qu'un processeur ou un disque. Elle peut être basée sur le temps ou sur la capacité.
  - Basée sur le temps : durée pendant laquelle une ressource est occupée au cours d'une période d'observation donnée.
  - Basée sur la capacité : débit qu'un système ou un composant peut fournir, en pourcentage de sa capacité.
- Saturation : mesure dans laquelle une ressource demande plus de travail qu'elle ne peut en traiter. Lorsque l'utilisation basée sur la capacité atteint 100 %, le travail supplémentaire ne peut pas être traité et doit être mis en file d'attente.

Utilisation de l'UC

Vous pouvez utiliser les outils suivants pour identifier l'utilisation et la saturation du processeur :

- CloudWatch fournit la métrique `CPUUtilization`. Si ce chiffre atteint 100 %, l'instance est saturée. Cependant, les métriques CloudWatch se basent sur une moyenne calculée sur une minute et manquent donc de granularité.

  Pour plus d'informations sur les métriques CloudWatch, consultez Métriques de niveau instance pour Amazon Aurora.
- La surveillance améliorée fournit les métriques renvoyées par la commande `top` du système d'exploitation. Elle affiche les moyennes de charge et les états de processeur suivants, avec une granularité d'une seconde :
  - `Idle (%)` = délai d'inactivité
  - `IRQ (%)` = interruptions logicielles
  - `Nice (%)` = moment parfait pour les processus avec une bonne priorité.

- `Steal (%)` = temps passé à desservir les autres locataires (lié à la virtualisation)
- `System (%)` = heure du système
- `User (%)` = heure de l'utilisateur
- `Wait (%)` = attente d'E/S

Pour plus d'informations sur les métriques de surveillance améliorée, consultez Métriques de système d'exploitation pour Aurora.

Utilisation de la mémoire

Si le système est soumis à une forte charge de mémoire et que la consommation des ressources atteint la saturation, vous devriez observer un taux élevé d'erreurs liées à l'analyse, à la pagination, à l'échange et à l'épuisement de la mémoire.

Vous pouvez utiliser les outils suivants pour identifier l'utilisation et la saturation de la mémoire :

CloudWatch fournit la métrique `FreeableMemory` qui indique la quantité de mémoire pouvant être récupérée en vidant certains caches du système d'exploitation et la mémoire disponible actuellement.

Pour plus d'informations sur les métriques CloudWatch, consultez Métriques de niveau instance pour Amazon Aurora.

La surveillance améliorée fournit les métriques suivantes qui peuvent vous aider à identifier les problèmes d'utilisation de la mémoire :

- `Buffers (KB)` : quantité de mémoire utilisée pour la mise en mémoire tampon des demandes d'E/S avant écriture dans le périphérique de stockage, en kilo-octets.
- `Cached (KB)` : quantité de mémoire utilisée pour la mise en cache des E/S basées sur le système de fichiers.
- `Free (KB)` : quantité de mémoire non attribuée, en kilo-octets.
- `Swap` : en cache, gratuit et total.

Par exemple, si vous constatez que votre instance de base de données utilise la mémoire `Swap`, la quantité totale de mémoire pour votre charge de travail est supérieure à celle dont dispose actuellement votre instance. Nous vous recommandons d'augmenter la taille de votre instance de base de données ou d'ajuster votre charge de travail pour qu'elle utilise moins de mémoire.

Pour plus d'informations sur les métriques de surveillance améliorée, consultez Métriques de système d'exploitation pour Aurora.

Pour des informations plus détaillées sur l'utilisation du schéma de performance et du schéma `sys` afin de déterminer les connexions et les composants qui utilisent de la mémoire, consultez Résolution des problèmes d'utilisation de la mémoire pour les bases de données Aurora MySQL.

Débit réseau

CloudWatch fournit les métriques suivantes concernant le débit total du réseau, toutes calculées en moyenne sur une minute :

- `NetworkReceiveThroughput` : quantité de débit réseau reçue des clients par chaque instance du cluster de bases de données Aurora.
- `NetworkTransmitThroughput` : quantité de débit réseau envoyée aux clients par chaque instance du cluster de bases de données Aurora.
- `NetworkThroughput` : quantité de débit réseau reçue des clients et transmise à ces derniers par chaque instance du cluster de bases de données Aurora.
- `StorageNetworkReceiveThroughput` : quantité de débit réseau reçue du sous-système de stockage Aurora par chaque instance du cluster de bases de données.
- `StorageNetworkTransmitThroughput` : quantité de débit réseau envoyée au sous-système de stockage Aurora par chaque instance du cluster de bases de données Aurora.
- `StorageNetworkThroughput` : quantité de débit réseau reçue du sous-système de stockage Aurora et envoyée à celui-ci par chaque instance du cluster de bases de données Aurora.

Pour plus d'informations sur les métriques CloudWatch, consultez Métriques de niveau instance pour Amazon Aurora.

La surveillance améliorée fournit les graphiques `network` reçus (RX) et transmis (TX), avec une granularité allant jusqu'à une seconde.

Pour plus d'informations sur les métriques de surveillance améliorée, consultez Métriques de système d'exploitation pour Aurora.

Métriques de base de données

Examinez les métriques CloudWatch suivantes pour identifier les modifications de la charge de travail :

- `BlockedTransactions` : nombre moyen de transactions de la base de données bloquées par seconde.
- `BufferCacheHitRatio` : pourcentage de demandes traitées par le cache de tampon.
- `CommitThroughput` : nombre moyen d'opérations de validation par seconde.
- `DatabaseConnections` : nombre de connexions réseau client à l'instance de base de données.
- `Deadlocks` : nombre moyen de blocages de la base de données par seconde.
- `DMLThroughput` : nombre moyen d'insertions, de mises à jour et de suppressions par seconde.
- `ResultSetCacheHitRatio` : pourcentage de demandes traitées par le cache de requêtes.
- `RollbackSegmentHistoryListLength` : journaux d'annulation qui enregistrent les transactions validées avec des enregistrements marqués pour la suppression.
- `RowLockTime` : temps total passé à acquérir des verrous de ligne pour les tables InnoDB.
- `SelectThroughput` : nombre moyen de requêtes SELECT par seconde.

Pour plus d'informations sur les métriques CloudWatch, consultez Métriques de niveau instance pour Amazon Aurora.

Lorsque vous examinez la charge de travail, posez-vous les questions suivantes :

1. Y a-t-il eu des changements récents dans la classe d'instance de base de données, par exemple une réduction de la taille de l'instance qui serait passé de 8xlarge à 4xlarge, ou le passage de db.r5 à db.r6 ?
2. Peut-on créer un clone et reproduire le problème, ou cela se produit-il uniquement sur cette instance ?
3. Les ressources du serveur sont-elles épuisées, le processeur est-il trop élevé ou la mémoire est-elle saturée ? Dans l'affirmative, cela peut signifier que du matériel supplémentaire est nécessaire.
4. Une ou plusieurs requêtes prennent-elles plus de temps ?
5. Les modifications sont-elles causées par une mise à niveau, en particulier une mise à niveau de version majeure ? Dans l'affirmative, comparez les métriques avant et après la mise à niveau.
6. Y a-t-il des changements du nombre d'instances de base de données en écriture ?
7. Avez-vous activé la journalisation générale, la journalisation d'audit ou la journalisation binaire ? Pour plus d'informations, consultez Journalisation pour les bases de données Aurora MySQL.
8. Avez-vous activé, désactivé ou modifié votre utilisation de la réplication des journaux binaires (binlog) ?

9. Existe-t-il des transactions de longue durée comportant un grand nombre de verrous de ligne ? Examinez la longueur de la liste d'historique (HLL) d'InnoDB pour obtenir des indications sur les transactions de longue durée.

   Pour plus d'informations, consultez La longueur de la liste d'historique InnoDB a considérablement augmenté et le billet de blog Pourquoi ma requête SELECT s'exécute-t-elle lentement sur mon cluster de bases de données Amazon Aurora MySQL ?.

   a. Si une grande longueur de la liste d'historique est causée par une transaction d'écriture, cela signifie que les journaux `UNDO` s'accumulent (ils ne sont donc pas nettoyés régulièrement). Dans le cas d'une transaction d'écriture de grande taille, cette accumulation peut augmenter rapidement. Dans MySQL, `UNDO` est stocké dans l'espace de table SYSTEM. La taille de l'espace de table `SYSTEM` ne peut pas être réduite. Le journal `UNDO` peut entraîner une augmentation de l'espace de table `SYSTEM` pouvant atteindre plusieurs Go, voire plusieurs To. Après la purge, libérez l'espace alloué en effectuant une sauvegarde logique (vidage) des données, puis importez le vidage dans une nouvelle instance de base de données.

   b. Si une grande longueur de la liste d'historique est causée par une transaction de lecture (requête de longue durée), cela peut signifier que la requête utilise une grande quantité d'espace temporaire. Libérez l'espace temporaire en effectuant un redémarrage. Examinez les métriques de la base de données Performance Insights pour détecter toute modification dans la section `Temp`, telle que `created_tmp_tables`. Pour plus d'informations, consultez Surveillance de la charge de la base de données avec Performance Insights sur .

10. Pouvez-vous diviser les transactions de longue durée en transactions plus petites qui modifient moins de lignes ?

11. Y a-t-il des changements des transactions bloquées ou une augmentation des blocages ? Examinez les métriques de la base de données Performance Insights pour détecter toute modification apportée aux variables d'état dans la section `Locks`, telles que `innodb_row_lock_time`, `innodb_row_lock_waits` et `innodb_dead_locks`. Utilisez des intervalles de 1 minute ou de 5 minutes.

12. Y a-t-il une augmentation des temps d'attente ? Examinez les événements et les types d'attente de Performance Insights avec des intervalles de 1 minute ou de 5 minutes. Analysez les principaux événements d'attente et déterminez s'ils sont corrélés à des modifications de la charge de travail ou à des conflits de base de données. Par exemple, `buf_pool mutex` indique un conflit du pool de tampons. Pour plus d'informations, consultez Réglage d'Aurora MySQL avec des événements d'attente.

Résolution des problèmes d'utilisation de la mémoire pour les bases de données Aurora MySQL

CloudWatch, Enhanced Monitoring et Performance Insights fournissent une bonne vue d'ensemble de l'utilisation de la mémoire au niveau du système d'exploitation, notamment de la quantité de mémoire utilisée par le processus de base de données. Cependant, ils ne vous permettent pas de déterminer les connexions ou les composants du moteur susceptibles d'être à l'origine de cette utilisation de mémoire.

Pour résoudre ce problème, vous pouvez utiliser le schéma de performance et le schéma `sys`. Dans Aurora MySQL version 3, l'instrumentation de la mémoire est activée par défaut lorsque le schéma de performance est activé. Dans Aurora MySQL version 2, seule l'instrumentation de la mémoire pour l'utilisation de la mémoire du schéma de performance est activée par défaut. Pour en savoir plus sur les tables disponibles dans le schéma de performance afin de suivre l'utilisation de la mémoire et sur l'activation de l'instrumentation de la mémoire du schéma de performance, consultez Memory Summary Tables dans la documentation MySQL. Pour plus d'informations sur l'utilisation du schéma de performance avec Performance Insights, consultez Présentation du schéma de performance pour Performance Insights sur Aurora MySQL.

Bien que des informations détaillées soient disponibles dans le schéma de performance pour suivre l'utilisation actuelle de la mémoire, le schéma système MySQL inclut des vues au-dessus des tables du schéma de performance. Ces vues vous permettent d'identifier rapidement où la mémoire est utilisée.

Dans le schéma `sys`, les vues suivantes sont disponibles pour suivre l'utilisation de la mémoire par connexion, composant et requête.

Vue	Description
memory_by_host_by_current_bytes	Fournit des informations sur l'utilisation de la mémoire du moteur par hôte. Cette vue peut être utile pour identifier les serveurs d'applica tions ou les hôtes clients qui consomment de la mémoire.
memory_by_thread_by_current_bytes	Fournit des informations sur l'utilisation de la mémoire du moteur par ID de thread. L'ID de thread dans MySQL peut être une connexion

Vue	Description
	client ou un thread d'arrière-plan. Vous pouvez mapper les ID de thread aux ID de connexion MySQL en utilisant la vue sys.processlist ou la table performance_schema.threads.
memory_by_user_by_current_bytes	Fournit des informations sur l'utilisation de la mémoire du moteur par utilisateur. Cette vue peut être utile pour identifier les comptes utilisateurs ou les clients consommant de la mémoire.
memory_global_by_current_bytes	Fournit des informations sur l'utilisation de la mémoire du moteur par composant du moteur. Cette vue peut être utile pour identifier l'utilisation globale de la mémoire par les tampons ou les composants du moteur. Par exemple, vous pouvez voir l'événement `memory/innodb/buf_buf_pool` pour le pool de tampons InnoDB ou l'événement `memory/sql/Prepared_statement::main_mem_root` pour les instructions préparées.
memory_global_total	Fournit une vue d'ensemble de l'utilisation totale de la mémoire suivie dans le moteur de base de données.

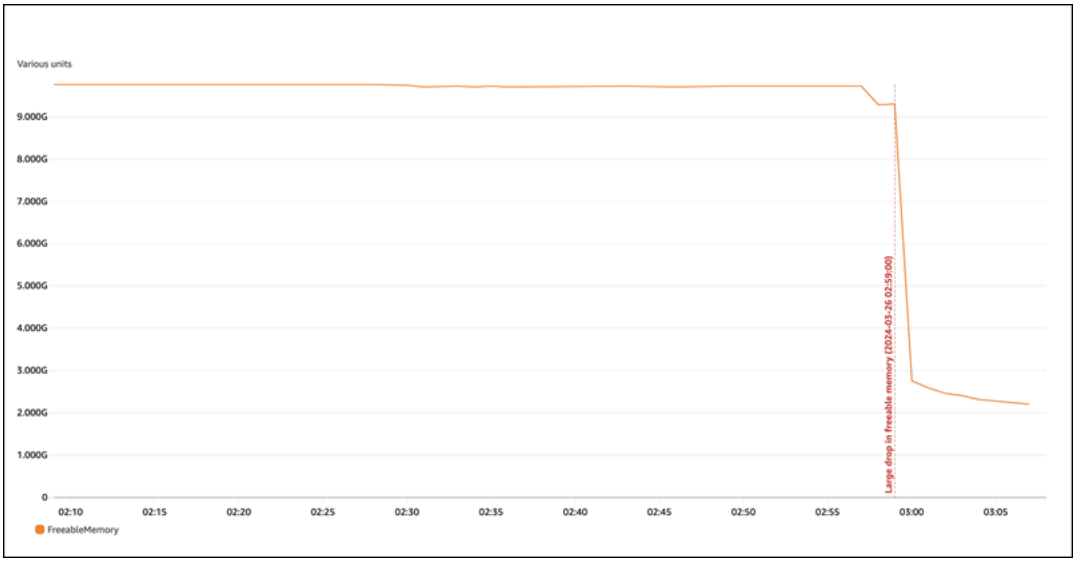
Dans Aurora MySQL 3.05 et versions ultérieures, vous pouvez également suivre l'utilisation maximale de la mémoire par résumé d'instruction dans les tableaux récapitulatifs des instructions du schéma de performance. Les tableaux récapitulatifs des instructions contiennent des résumés d'instructions normalisés et des statistiques agrégées sur leur exécution. La colonne `MAX_TOTAL_MEMORY` peut vous aider à identifier la mémoire maximale utilisée par le résumé des requêtes depuis la dernière réinitialisation des statistiques ou depuis le redémarrage de l'instance de base de données. Cela peut être utile pour identifier des requêtes spécifiques susceptibles de consommer beaucoup de mémoire.

> **Note**
>
> Le schéma de performance et le schéma `sys` indiquent l'utilisation actuelle de la mémoire sur le serveur, ainsi que le maximum de mémoire consommée par connexion et par composant du moteur. Le schéma de performance étant conservé en mémoire, les informations sont réinitialisées au redémarrage de l'instance de base de données. Pour en conserver un historique au fil du temps, nous vous recommandons de configurer la récupération et le stockage de ces données en dehors du schéma de performance.

Rubriques

Exemple 1 : utilisation élevée de la mémoire en permanence

En examinant `FreeableMemory` dans CloudWatch, nous constatons que l'utilisation de la mémoire a considérablement augmenté le 26/03/2024 à 02:59 UTC.



Nous ne disposons toutefois pas de toutes les informations requises. Pour déterminer quel composant utilise le plus de mémoire, vous pouvez vous connecter à la base de données et consulter `sys.memory_global_by_current_bytes`. Cette table contient une liste des événements de mémoire suivis par MySQL, ainsi que des informations sur l'allocation de mémoire par événement.

Chaque événement de suivi de la mémoire commence par `memory/%`, suivi d'autres informations sur le composant ou la fonctionnalité du moteur auquel l'événement est associé.

Par exemple, `memory/performance_schema/%` indique les événements de mémoire liés au schéma de performance, `memory/innodb/%` correspond à InnoDB, etc. Pour plus d'informations sur les conventions de dénomination utilisées dans les événements, consultez Performance Schema Instrument Naming Conventions dans la documentation MySQL.

Dans la requête suivante, nous pouvons identifier le responsable le plus probable en fonction de `current_alloc`, mais nous constatons également la présence de nombreux événements `memory/performance_schema/%`.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes LIMIT 10;

+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| event_name                                                                    |
 current_count | current_alloc | current_avg_alloc | high_count | high_alloc |
 high_avg_alloc |
+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| memory/sql/Prepared_statement::main_mem_root                                  |
 512817 | 4.91 GiB      | 10.04 KiB         |     512823 | 4.91 GiB   | 10.04 KiB
 |
| memory/performance_schema/prepared_statements_instances                       |
   252 | 488.25 MiB    | 1.94 MiB          |        252 | 488.25 MiB | 1.94 MiB       |
| memory/innodb/hash0hash                                                       |
     4 | 79.07 MiB     | 19.77 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/performance_schema/events_errors_summary_by_thread_by_error            |
  1028 | 52.27 MiB     | 52.06 KiB         |       1028 | 52.27 MiB  | 52.06 KiB      |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name |
     4 | 47.25 MiB     | 11.81 MiB         |          4 | 47.25 MiB  | 11.81 MiB      |
| memory/performance_schema/events_statements_summary_by_digest                 |
     1 | 40.28 MiB     | 40.28 MiB         |          1 | 40.28 MiB  | 40.28 MiB      |
| memory/performance_schema/memory_summary_by_thread_by_event_name              |
     4 | 31.64 MiB     | 7.91 MiB          |          4 | 31.64 MiB  | 7.91 MiB       |
| memory/innodb/memory                                                          |
 15227 | 27.44 MiB     | 1.85 KiB          |      20619 | 33.33 MiB  | 1.66 KiB       |
| memory/sql/String::value                                                      |
 74411 | 21.85 MiB     |  307 bytes        |      76867 | 25.54 MiB  |  348 bytes     |
```

```
| memory/sql/TABLE                                                                 |
  8381 | 21.03 MiB     | 2.57 KiB          |       8381 | 21.03 MiB  | 2.57 KiB       |
+-------------------------------------------------------------------------------
+--------------+--------------+------------------+------------+------------
+----------------+
10 rows in set (0.02 sec)
```

Nous avons mentionné précédemment que le schéma de performance est stocké en mémoire, ce qui signifie qu'il est également suivi dans l'instrumentation de la mémoire `performance_schema`.

> **Note**
>
> Si vous constatez que le schéma de performance utilise beaucoup de mémoire et que vous souhaitez limiter son utilisation, vous pouvez ajuster les paramètres de base de données en fonction de vos besoins. Pour plus d'informations, consultez The Performance Schema Memory-Allocation Model dans la documentation MySQL.

Pour une meilleure lisibilité, vous pouvez réexécuter la même requête, mais exclure les événements du schéma de performance. Voici les informations que nous donne le résultat :

- Le principal consommateur de mémoire est `memory/sql/Prepared_statement::main_mem_root`.
- La colonne `current_alloc` nous indique que MySQL dispose actuellement de 4,91 Gio alloués à cet événement.
- `high_alloc column` indique que 4,91 Gio est le seuil maximum de `current_alloc` depuis la dernière réinitialisation des statistiques ou depuis le redémarrage du serveur. Autrement dit, `memory/sql/Prepared_statement::main_mem_root` a atteint sa valeur la plus élevée.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name NOT LIKE
 'memory/performance_schema/%' LIMIT 10;

+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
```

```
| memory/sql/Prepared_statement::main_mem_root  |        512817 | 4.91 GiB      | 10.04
 KiB         |     512823 | 4.91 GiB   | 10.04 KiB      |
| memory/innodb/hash0hash                       |             4 | 79.07 MiB     | 19.77
 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/innodb/memory                          |         17096 | 31.68 MiB     | 1.90
 KiB         |      22498 | 37.60 MiB  | 1.71 KiB       |
| memory/sql/String::value                      |        122277 | 27.94 MiB     |  239
 bytes       |     124699 | 29.47 MiB  |  247 bytes     |
| memory/sql/TABLE                              |          9927 | 24.67 MiB     | 2.55
 KiB         |       9929 | 24.68 MiB  | 2.55 KiB       |
| memory/innodb/lock0lock                       |          8888 | 19.71 MiB     | 2.27
 KiB         |       8888 | 19.71 MiB  | 2.27 KiB       |
| memory/sql/Prepared_statement::infrastructure |        257623 | 16.24 MiB     |   66
 bytes       |     257631 | 16.24 MiB  |   66 bytes     |
| memory/mysys/KEY_CACHE                        |             3 | 16.00 MiB     | 5.33
 MiB         |          3 | 16.00 MiB  | 5.33 MiB       |
| memory/innodb/sync0arr                        |             3 | 7.03 MiB      | 2.34
 MiB         |          3 | 7.03 MiB   | 2.34 MiB       |
| memory/sql/THD::main_mem_root                 |           815 | 6.56 MiB      | 8.24
 KiB         |        849 | 7.19 MiB   | 8.67 KiB       |
+-----------------------------------------------+---------------+---------------
+------------------+-----------+-----------+----------------+
10 rows in set (0.06 sec)
```

Le nom de l'événement révèle que cette mémoire est utilisée pour des instructions préparées. Pour déterminer quelles connexions utilisent cette mémoire, vérifiez memory_by_thread_by_current_bytes.

Dans l'exemple suivant, environ 7 Mio sont alloués à chaque connexion, avec un seuil maximum d'environ 6,29 Mio (`current_max_alloc`). Cela est logique, car cet exemple utilise `sysbench` avec 80 tables et 800 connexions avec des instructions préparées. Si vous souhaitez réduire l'utilisation de la mémoire dans ce scénario, vous pouvez optimiser l'utilisation des instructions préparées par votre application afin de limiter la consommation de la mémoire.

```
mysql> SELECT * FROM sys.memory_by_thread_by_current_bytes;

+-----------+-----------------------------------------+--------------------
+------------------+------------------+------------------+-----------------+
| thread_id | user                                    | current_count_used |
 current_allocated | current_avg_alloc | current_max_alloc | total_allocated |
+-----------+-----------------------------------------+--------------------
+------------------+------------------+------------------+-----------------+
|        46 | rdsadmin@localhost                      |                405 | 8.47 MiB
          | 21.42 KiB         | 8.00 MiB          | 155.86 MiB      |
```

```
|        61 | reinvent@10.0.4.4                        |              1749 | 6.72 MiB
          | 3.93 KiB          | 6.29 MiB          | 14.24 MiB       |
|       101 | reinvent@10.0.4.4                        |              1845 | 6.71 MiB
          | 3.72 KiB          | 6.29 MiB          | 14.50 MiB       |
|        55 | reinvent@10.0.4.4                        |              1674 | 6.68 MiB
          | 4.09 KiB          | 6.29 MiB          | 14.13 MiB       |
|        57 | reinvent@10.0.4.4                        |              1416 | 6.66 MiB
          | 4.82 KiB          | 6.29 MiB          | 13.52 MiB       |
|       112 | reinvent@10.0.4.4                        |              1759 | 6.66 MiB
          | 3.88 KiB          | 6.29 MiB          | 14.17 MiB       |
|        66 | reinvent@10.0.4.4                        |              1428 | 6.64 MiB
          | 4.76 KiB          | 6.29 MiB          | 13.47 MiB       |
|        75 | reinvent@10.0.4.4                        |              1389 | 6.62 MiB
          | 4.88 KiB          | 6.29 MiB          | 13.40 MiB       |
|       116 | reinvent@10.0.4.4                        |              1333 | 6.61 MiB
          | 5.08 KiB          | 6.29 MiB          | 13.21 MiB       |
|        90 | reinvent@10.0.4.4                        |              1448 | 6.59 MiB
          | 4.66 KiB          | 6.29 MiB          | 13.58 MiB       |
|        98 | reinvent@10.0.4.4                        |              1440 | 6.57 MiB
          | 4.67 KiB          | 6.29 MiB          | 13.52 MiB       |
|        94 | reinvent@10.0.4.4                        |              1433 | 6.57 MiB
          | 4.69 KiB          | 6.29 MiB          | 13.49 MiB       |
|        62 | reinvent@10.0.4.4                        |              1323 | 6.55 MiB
          | 5.07 KiB          | 6.29 MiB          | 13.48 MiB       |
|        87 | reinvent@10.0.4.4                        |              1323 | 6.55 MiB
          | 5.07 KiB          | 6.29 MiB          | 13.25 MiB       |
|        99 | reinvent@10.0.4.4                        |              1346 | 6.54 MiB
          | 4.98 KiB          | 6.29 MiB          | 13.24 MiB       |
|       105 | reinvent@10.0.4.4                        |              1347 | 6.54 MiB
          | 4.97 KiB          | 6.29 MiB          | 13.34 MiB       |
|        73 | reinvent@10.0.4.4                        |              1335 | 6.54 MiB
          | 5.02 KiB          | 6.29 MiB          | 13.23 MiB       |
|        54 | reinvent@10.0.4.4                        |              1510 | 6.53 MiB
          | 4.43 KiB          | 6.29 MiB          | 13.49 MiB       |
.
                                                                    .
.
                                                                    .
.
                                                                    .
|       812 | reinvent@10.0.4.4                        |              1259 | 6.38 MiB
          | 5.19 KiB          | 6.29 MiB          | 13.05 MiB       |
|       214 | reinvent@10.0.4.4                        |              1279 | 6.38 MiB
          | 5.10 KiB          | 6.29 MiB          | 12.90 MiB       |
```

```
|       325 | reinvent@10.0.4.4                          |               1254 | 6.38 MiB
          | 5.21 KiB          | 6.29 MiB          | 12.99 MiB       |
|       705 | reinvent@10.0.4.4                          |               1273 | 6.37 MiB
          | 5.13 KiB          | 6.29 MiB          | 13.03 MiB       |
|       530 | reinvent@10.0.4.4                          |               1268 | 6.37 MiB
          | 5.15 KiB          | 6.29 MiB          | 12.92 MiB       |
|       307 | reinvent@10.0.4.4                          |               1263 | 6.37 MiB
          | 5.17 KiB          | 6.29 MiB          | 12.87 MiB       |
|       738 | reinvent@10.0.4.4                          |               1260 | 6.37 MiB
          | 5.18 KiB          | 6.29 MiB          | 13.00 MiB       |
|       819 | reinvent@10.0.4.4                          |               1252 | 6.37 MiB
          | 5.21 KiB          | 6.29 MiB          | 13.01 MiB       |
|        31 | innodb/srv_purge_thread                    |              17810 | 3.14 MiB
          |  184 bytes        | 2.40 MiB          | 205.69 MiB      |
|        38 | rdsadmin@localhost                         |                599 | 1.76 MiB
          | 3.01 KiB          | 1.00 MiB          | 25.58 MiB       |
|         1 | sql/main                                   |               3756 | 1.32 MiB
          |  367 bytes        | 355.78 KiB        | 6.19 MiB        |
|       854 | rdsadmin@localhost                         |                 46 | 1.08 MiB
          | 23.98 KiB         | 1.00 MiB          | 5.10 MiB        |
|        30 | innodb/clone_gtid_thread                   |               1596 | 573.14
 KiB        |  367 bytes        | 254.91 KiB        | 970.69 KiB      |
|        40 | rdsadmin@localhost                         |                235 | 245.19
 KiB        | 1.04 KiB          | 128.88 KiB        | 808.64 KiB      |
|       853 | rdsadmin@localhost                         |                 96 | 94.63
 KiB        | 1009 bytes        | 29.73 KiB         | 422.45 KiB      |
|        36 | rdsadmin@localhost                         |                 33 | 36.29
 KiB        | 1.10 KiB          | 16.08 KiB         | 74.15 MiB       |
|        33 | sql/event_scheduler                        |                  3 | 16.27
 KiB        | 5.42 KiB          | 16.04 KiB         | 16.27 KiB       |
|        35 | sql/compress_gtid_table                    |                  8 | 14.20
 KiB        | 1.77 KiB          | 8.05 KiB          | 18.62 KiB       |
|        25 | innodb/fts_optimize_thread                 |                 12 | 1.86 KiB
          |  158 bytes        |  648 bytes        | 1.98 KiB        |
|        23 | innodb/srv_master_thread                   |                 11 | 1.23 KiB
          |  114 bytes        |  361 bytes        | 24.40 KiB       |
|        24 | innodb/dict_stats_thread                   |                 11 | 1.23 KiB
          |  114 bytes        |  361 bytes        | 1.35 KiB        |
|         5 | innodb/io_read_thread                      |                  1 |  144
 bytes        |  144 bytes        |  144 bytes        |  144 bytes      |
|         6 | innodb/io_read_thread                      |                  1 |  144
 bytes        |  144 bytes        |  144 bytes        |  144 bytes      |
|         2 | sql/aws_oscar_log_level_monitor            |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
```

```
|         4 | innodb/io_ibuf_thread                       |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         7 | innodb/io_write_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         8 | innodb/io_write_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|         9 | innodb/io_write_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        10 | innodb/io_write_thread                      |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        11 | innodb/srv_lra_thread                       |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        12 | innodb/srv_akp_thread                       |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        18 | innodb/srv_lock_timeout_thread              |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |  248 bytes      |
|        19 | innodb/srv_error_monitor_thread             |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        20 | innodb/srv_monitor_thread                   |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        21 | innodb/buf_resize_thread                    |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        22 | innodb/btr_search_sys_toggle_thread         |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        32 | innodb/dict_persist_metadata_table_thread   |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
|        34 | sql/signal_handler                          |                  0 |    0
 bytes        |    0 bytes        |    0 bytes        |    0 bytes      |
+-----------+---------------------------------------------+--------------------
+-------------------+------------------+------------------+-----------------+
831 rows in set (2.48 sec)
```

Comme indiqué précédemment, la valeur de l'ID de thread (`thd_id`) ici peut faire référence aux threads d'arrière-plan du serveur ou aux connexions à la base de données. Si vous souhaitez associer les valeurs des ID de thread aux ID de connexion à la base de données, vous pouvez utiliser la table `performance_schema.threads` ou la vue `sys.processlist`, où `conn_id` correspond à l'ID de connexion.

```
mysql> SELECT thd_id,conn_id,user,db,command,state,time,last_wait FROM sys.processlist
 WHERE user='reinvent@10.0.4.4';

+--------+---------+------------------+----------+---------+---------------+------
+-----------------------------------------------+
```

```
| thd_id | conn_id | user               | db       | command | state          | time |
 last_wait                                    |
+--------+---------+--------------------+----------+---------+----------------+------
+----------------------------------------------+
|    590 |     562 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                       |
|    578 |     550 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                         |
|    579 |     551 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                       |
|    580 |     552 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                    |
|    581 |     553 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                    |
|    582 |     554 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                         |
|    583 |     555 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                         |
|    584 |     556 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                    |
|    585 |     557 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                       |
|    586 |     558 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                    |
|    587 |     559 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                       |
.
                                          .
.
                                          .
.
                                          .
|    323 |     295 | reinvent@10.0.4.4 | sysbench | Sleep   | NULL           |    0 |
 idle                                         |
|    324 |     296 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                    |
|    325 |     297 | reinvent@10.0.4.4 | sysbench | Execute | closing tables |    0 |
 wait/io/redo_log_flush                       |
|    326 |     298 | reinvent@10.0.4.4 | sysbench | Execute | updating       |    0 |
 wait/io/table/sql/handler                    |
|    438 |     410 | reinvent@10.0.4.4 | sysbench | Execute | System lock    |    0 |
 wait/lock/table/sql/handler                  |
|    280 |     252 | reinvent@10.0.4.4 | sysbench | Sleep   | starting       |    0 |
 wait/io/socket/sql/client_connection         |
```

```
|     98 |      70 | reinvent@10.0.4.4 | sysbench | Query   | freeing items  |    0 |
 NULL                                          |
+--------+---------+-------------------+----------+---------+----------------+------
+-----------------------------------------------+
804 rows in set (5.51 sec)
```

Nous arrêtons maintenant la charge de travail `sysbench`, qui ferme les connexions et libère de la mémoire. En vérifiant à nouveau les événements, nous pouvons confirmer que la mémoire est libérée, mais `high_alloc` nous indique tout de même quel est le seuil maximum. La colonne `high_alloc` peut être très utile pour identifier de courts pics d'utilisation de la mémoire, contrairement à `current_alloc`, qui indique uniquement la mémoire actuellement allouée.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name='memory/sql/
Prepared_statement::main_mem_root' LIMIT 10;

+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
| memory/sql/Prepared_statement::main_mem_root |            17 | 253.80 KiB    | 14.93
 KiB         |     512823 | 4.91 GiB   | 10.04 KiB      |
+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
1 row in set (0.00 sec)
```

Si vous souhaitez réinitialiser `high_alloc`, vous pouvez tronquer les tableaux récapitulatifs de la mémoire `performance_schema`, mais toute l'instrumentation de la mémoire sera réinitialisée. Pour plus d'informations, consultez Performance Schema General Table Characteristics dans la documentation MySQL.

Dans l'exemple suivant, nous pouvons voir que `high_alloc` est réinitialisé après la troncature.

```
mysql> TRUNCATE `performance_schema`.`memory_summary_global_by_event_name`;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM sys.memory_global_by_current_bytes WHERE event_name='memory/sql/
Prepared_statement::main_mem_root' LIMIT 10;

+-----------------------------------------------+--------------+--------------
+------------------+-----------+-----------+----------------+
```

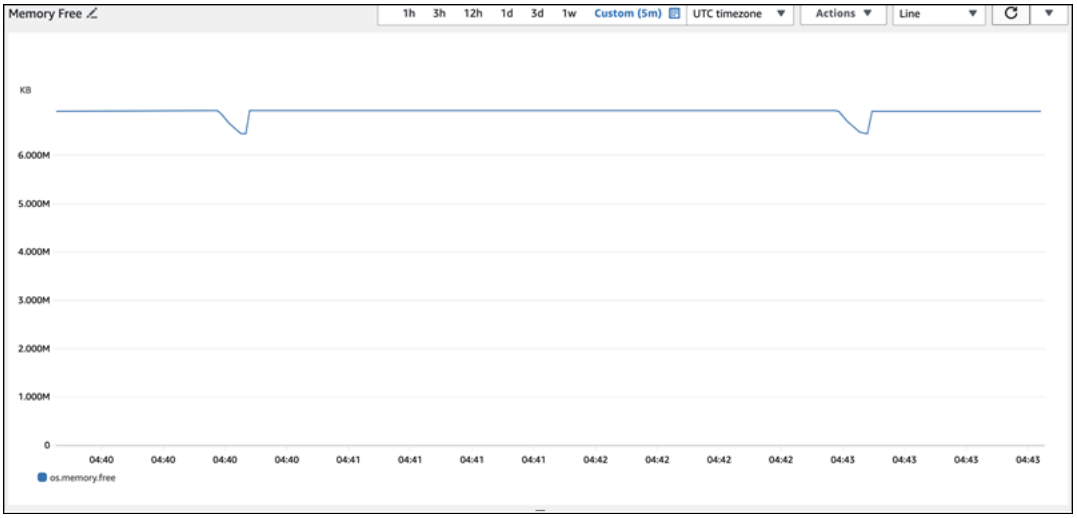
```
| event_name                                    | current_count | current_alloc |
 current_avg_alloc | high_count | high_alloc | high_avg_alloc |
+-----------------------------------------------+---------------+---------------
+------------------+------------+------------+----------------+
| memory/sql/Prepared_statement::main_mem_root |            17 | 253.80 KiB    | 14.93
 KiB         |         17 | 253.80 KiB | 14.93 KiB      |
+-----------------------------------------------+---------------+---------------
+------------------+------------+------------+----------------+
1 row in set (0.00 sec)
```

Exemple 2 : pics de mémoire transitoires

Les courts pics d'utilisation de la mémoire sur un serveur de base de données constituent un autre phénomène courant. Il peut s'agir de baisses périodiques de mémoire libérable difficiles à gérer avec `sys.memory_global_by_current_bytes` dans `current_alloc`, car la mémoire a déjà été libérée.

> **Note**
>
> Si les statistiques du schéma de performance ont été réinitialisées ou si l'instance de base de données a été redémarrée, ces informations ne seront pas disponibles dans `sys` ni `performance_schema`. Pour conserver ces informations, nous vous recommandons de configurer la collecte de métriques externes.

Le graphique suivant de la métrique `os.memory.free` dans Surveillance améliorée présente de courts pics d'utilisation de la mémoire de 7 secondes. La section Surveillance améliorée vous permet de surveiller l'utilisation à des intervalles pouvant atteindre une seconde, ce qui est parfait pour détecter ce type de pic transitoire.



Pour aider à diagnostiquer la cause de l'utilisation de la mémoire, nous pouvons utiliser à la fois `high_alloc` dans les vues récapitulatives de la mémoire `sys` et les tableaux récapitulatifs des instructions du schéma de performance afin d'essayer d'identifier les sessions et les connexions problématiques.

Comme prévu, étant donné que l'utilisation de la mémoire n'est pas élevée actuellement, nous ne voyons aucun responsable majeur dans la vue du schéma `sys`, sous`current_alloc`.

```
mysql> SELECT * FROM sys.memory_global_by_current_bytes LIMIT 10;

+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| event_name                                                                    |
 current_count | current_alloc | current_avg_alloc | high_count | high_alloc |
 high_avg_alloc |
+-------------------------------------------------------------------------------
+---------------+---------------+-------------------+------------+------------
+----------------+
| memory/innodb/hash0hash                                                       |
     4 | 79.07 MiB     | 19.77 MiB         |          4 | 79.07 MiB  | 19.77 MiB      |
| memory/innodb/os0event                                                        |
 439372 | 60.34 MiB     |  144 bytes        |     439372 | 60.34 MiB  |  144 bytes
 |
| memory/performance_schema/events_statements_summary_by_digest                |
     1 | 40.28 MiB     | 40.28 MiB         |          1 | 40.28 MiB  | 40.28 MiB      |
| memory/mysys/KEY_CACHE                                                        |
     3 | 16.00 MiB     | 5.33 MiB          |          3 | 16.00 MiB  | 5.33 MiB       |
```

```
| memory/performance_schema/events_statements_history_long                    |
     1 | 14.34 MiB     | 14.34 MiB         |          1 | 14.34 MiB  | 14.34 MiB      |
| memory/performance_schema/events_errors_summary_by_thread_by_error          |
   257 | 13.07 MiB     | 52.06 KiB         |        257 | 13.07 MiB  | 52.06 KiB      |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name |
     1 | 11.81 MiB     | 11.81 MiB         |          1 | 11.81 MiB  | 11.81 MiB      |
| memory/performance_schema/events_statements_summary_by_digest.digest_text   |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
| memory/performance_schema/events_statements_history_long.digest_text        |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
| memory/performance_schema/events_statements_history_long.sql_text           |
     1 | 9.77 MiB      | 9.77 MiB          |          1 | 9.77 MiB   | 9.77 MiB       |
+-------------------------------------------------------------------------------
+---------------+---------------+------------------+------------+------------
+----------------+
10 rows in set (0.01 sec)
```

En élargissant la vue pour effectuer un tri par `high_alloc`, nous constatons que le composant `memory/temptable/physical_ram` est très probablement responsable de l'utilisation élevée de la mémoire. À son maximum, il consommait 515,00 Mio.

Comme son nom l'indique, `memory/temptable/physical_ram` instrumente l'utilisation de la mémoire pour le moteur de stockage TEMP de MySQL, introduit dans MySQL 8.0. Pour plus d'informations sur la façon dont MySQL utilise les tables temporaires, consultez Internal temporary table use in MySQL dans la documentation MySQL.

> **Note**
>
> Nous utilisons la vue `sys.x$memory_global_by_current_bytes` dans cet exemple.

```
mysql> SELECT event_name, format_bytes(current_alloc) AS "currently allocated",
 sys.format_bytes(high_alloc) AS "high-water mark"
FROM sys.x$memory_global_by_current_bytes ORDER BY high_alloc DESC LIMIT 10;

+-------------------------------------------------------------------------------
+---------------------+-----------------+
| event_name                                                                   |
 currently allocated | high-water mark |
+-------------------------------------------------------------------------------
+---------------------+-----------------+
```

```
| memory/temptable/physical_ram                                                 | 4.00
 MiB            | 515.00 MiB      |
| memory/innodb/hash0hash                                                       | 79.07
 MiB           | 79.07 MiB       |
| memory/innodb/os0event                                                        | 63.95
 MiB           | 63.95 MiB       |
| memory/performance_schema/events_statements_summary_by_digest                | 40.28
 MiB           | 40.28 MiB       |
| memory/mysys/KEY_CACHE                                                        | 16.00
 MiB           | 16.00 MiB       |
| memory/performance_schema/events_statements_history_long                      | 14.34
 MiB           | 14.34 MiB       |
| memory/performance_schema/events_errors_summary_by_thread_by_error           | 13.07
 MiB           | 13.07 MiB       |
| memory/performance_schema/events_statements_summary_by_thread_by_event_name | 11.81
 MiB           | 11.81 MiB       |
| memory/performance_schema/events_statements_summary_by_digest.digest_text    | 9.77
 MiB            | 9.77 MiB        |
| memory/performance_schema/events_statements_history_long.sql_text            | 9.77
 MiB            | 9.77 MiB        |
+-------------------------------------------------------------------------------
+--------------------+-----------------+
10 rows in set (0.00 sec)
```

DansExemple 1 : utilisation élevée de la mémoire en permanence, nous avons vérifié l'utilisation actuelle de la mémoire pour chaque connexion afin de déterminer quelle connexion est responsable de l'utilisation de la mémoire en question. Dans cet exemple, la mémoire est déjà libérée. Il n'est donc pas utile de vérifier l'utilisation de la mémoire pour les connexions en cours.

Pour aller plus loin dans notre recherche et trouver les instructions, les utilisateurs et les hôtes problématiques, utilisons le schéma de performance. Le schéma de performance contient plusieurs tableaux récapitulatifs des instructions divisés en différentes dimensions, telles que le nom de l'événement, le résumé de l'instruction, l'hôte, le thread et l'utilisateur. Chaque vue vous permet de mieux comprendre où certaines instructions sont exécutées et ce qu'elles font. Cette section se concentre sur `MAX_TOTAL_MEMORY`, mais vous trouverez plus d'informations sur toutes les colonnes disponibles dans la documentation Tableaux récapitulatifs des instructions du schéma de performance.

```
mysql> SHOW TABLES IN performance_schema LIKE 'events_statements_summary_%';

+-----------------------------------------------------------+
| Tables_in_performance_schema (events_statements_summary_%) |
```

```
+-----------------------------------------------------------+
| events_statements_summary_by_account_by_event_name        |
| events_statements_summary_by_digest                       |
| events_statements_summary_by_host_by_event_name           |
| events_statements_summary_by_program                      |
| events_statements_summary_by_thread_by_event_name         |
| events_statements_summary_by_user_by_event_name           |
| events_statements_summary_global_by_event_name            |
+-----------------------------------------------------------+
7 rows in set (0.00 sec)
```

Vérifions d'abord `events_statements_summary_by_digest` pour voir `MAX_TOTAL_MEMORY`.

Nous en tirons les informations suivantes :

- La requête avec le résumé `20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a` semble être un responsable probable de cette utilisation de la mémoire. `MAX_TOTAL_MEMORY` indique 537450710, ce qui correspond au seuil maximum que nous avons observé pour l'événement `memory/temptable/physical_ram` dans `sys.x$memory_global_by_current_bytes`.
- Elle a été exécutée quatre fois (`COUNT_STAR`): la première fois le 26/03/2024 04:08:34.943256, et la dernière fois le 26/03/2024 04:43:06.998310.

```
mysql> SELECT SCHEMA_NAME,DIGEST,COUNT_STAR,MAX_TOTAL_MEMORY,FIRST_SEEN,LAST_SEEN
FROM performance_schema.events_statements_summary_by_digest ORDER BY MAX_TOTAL_MEMORY
 DESC LIMIT 5;

+-------------+------------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
| SCHEMA_NAME | DIGEST                                                           |
 COUNT_STAR | MAX_TOTAL_MEMORY | FIRST_SEEN                | LAST_SEEN
  |
+-------------+------------------------------------------------------------------
+------------+-----------------+---------------------------
+----------------------------+
| sysbench    | 20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a |
     4 |        537450710 | 2024-03-26 04:08:34.943256 | 2024-03-26 04:43:06.998310 |
| NULL        | f158282ea0313fefd0a4778f6e9b92fc7d1e839af59ebd8c5eea35e12732c45d |
     4 |          3636413 | 2024-03-26 04:29:32.712348 | 2024-03-26 04:36:26.269329 |
```

```
| NULL        | 0046bc5f642c586b8a9afd6ce1ab70612dc5b1fd2408fa8677f370c1b0ca3213 |
     2 |          3459965 | 2024-03-26 04:31:37.674008 | 2024-03-26 04:32:09.410718 |
| NULL        | 8924f01bba3c55324701716c7b50071a60b9ceaf17108c71fd064c20c4ab14db |
     1 |          3290981 | 2024-03-26 04:31:49.751506 | 2024-03-26 04:31:49.751506 |
| NULL        | 90142bbcb50a744fcec03a1aa336b2169761597ea06d85c7f6ab03b5a4e1d841 |
     1 |          3131729 | 2024-03-26 04:15:09.719557 | 2024-03-26 04:15:09.719557 |
+-------------+------------------------------------------------------------------
+------------+-----------------+--------------------------
+---------------------------+
5 rows in set (0.00 sec)
```

Maintenant que nous connaissons le résumé problématique, nous pouvons obtenir plus d'informations telles que le texte de la requête, l'utilisateur qui l'a exécutée et l'endroit où elle a été exécutée. Grâce au texte du résumé renvoyé, nous voyons qu'il s'agit d'une expression de table commune (CTE) qui crée quatre tables temporaires et effectue quatre analyses de tables, ce qui est très inefficace.

```
mysql> SELECT
 SCHEMA_NAME,DIGEST_TEXT,QUERY_SAMPLE_TEXT,MAX_TOTAL_MEMORY,SUM_ROWS_SENT,SUM_ROWS_EXAMINED,SUM
FROM performance_schema.events_statements_summary_by_digest
WHERE DIGEST='20676ce4a690592ff05debcffcbc26faeb76f22005e7628364d7a498769d0c4a'\G;

*************************** 1. row ***************************
           SCHEMA_NAME: sysbench
           DIGEST_TEXT: WITH RECURSIVE `cte` ( `n` ) AS ( SELECT ? FROM `sbtest1` UNION
 ALL SELECT `id` + ? FROM `sbtest1` ) SELECT * FROM `cte`
     QUERY_SAMPLE_TEXT: WITH RECURSIVE cte (n) AS (   SELECT 1  from sbtest1 UNION ALL
  SELECT id + 1 FROM sbtest1) SELECT * FROM cte
      MAX_TOTAL_MEMORY: 537450710
         SUM_ROWS_SENT: 80000000
     SUM_ROWS_EXAMINED: 80000000
SUM_CREATED_TMP_TABLES: 4
     SUM_NO_INDEX_USED: 4
1 row in set (0.01 sec)
```

Pour plus d'informations sur la table `events_statements_summary_by_digest` et les autres tableaux récapitulatifs des instructions du schéma de performance, consultez Statement summary tables dans la documentation MySQL.

Vous pouvez également exécuter une instruction EXPLAIN ou EXPLAIN ANALYZE pour obtenir plus de détails.

> **Note**
>
> L'instruction `EXPLAIN ANALYZE` peut fournir plus d'informations que `EXPLAIN`, mais elle exécute également la requête. Vous devez donc être prudent.

```
-- EXPLAIN
mysql> EXPLAIN WITH RECURSIVE cte (n) AS (SELECT 1  FROM sbtest1 UNION ALL SELECT id +
 1 FROM sbtest1) SELECT * FROM cte;

+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
| id | select_type | table      | partitions | type  | possible_keys | key  | key_len |
 ref  | rows     | filtered | Extra       |
+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
|  1 | PRIMARY     | <derived2> | NULL       | ALL   | NULL          | NULL | NULL    |
 NULL | 19221520 |   100.00 | NULL        |
|  2 | DERIVED     | sbtest1    | NULL       | index | NULL          | k_1  | 4       |
 NULL |  9610760 |   100.00 | Using index |
|  3 | UNION       | sbtest1    | NULL       | index | NULL          | k_1  | 4       |
 NULL |  9610760 |   100.00 | Using index |
+----+-------------+------------+------------+-------+---------------+------+---------
+------+----------+----------+-------------+
3 rows in set, 1 warning (0.00 sec)

-- EXPLAIN format=tree
mysql> EXPLAIN format=tree WITH RECURSIVE cte (n) AS (SELECT 1 FROM sbtest1 UNION ALL
 SELECT id + 1 FROM sbtest1) SELECT * FROM cte\G;

*************************** 1. row ***************************
EXPLAIN: -> Table scan on cte  (cost=4.11e+6..4.35e+6 rows=19.2e+6)
    -> Materialize union CTE cte  (cost=4.11e+6..4.11e+6 rows=19.2e+6)
        -> Index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
        -> Index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
1 row in set (0.00 sec)

-- EXPLAIN ANALYZE
mysql> EXPLAIN ANALYZE WITH RECURSIVE cte (n) AS (SELECT 1 from sbtest1 UNION ALL
 SELECT id + 1 FROM sbtest1) SELECT * FROM cte\G;

*************************** 1. row ***************************
```

```
EXPLAIN: -> Table scan on cte  (cost=4.11e+6..4.35e+6 rows=19.2e+6) (actual
 time=6666..9201 rows=20e+6 loops=1)
    -> Materialize union CTE cte  (cost=4.11e+6..4.11e+6 rows=19.2e+6) (actual
 time=6666..6666 rows=20e+6 loops=1)
       -> Covering index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
 (actual time=0.0365..2006 rows=10e+6 loops=1)
       -> Covering index scan on sbtest1 using k_1  (cost=1.09e+6 rows=9.61e+6)
 (actual time=0.0311..2494 rows=10e+6 loops=1)
1 row in set (10.53 sec)
```

Mais qui l'a exécutée ? Dans le schéma de performance, nous pouvons voir que l'utilisateur `destructive_operator` a atteint la valeur `MAX_TOTAL_MEMORY` 537450710, ce qui correspond aux résultats précédents.

> **Note**
>
> Comme le schéma de performance est stocké en mémoire, il ne doit pas être considéré comme la seule source d'audit. Si vous devez conserver un historique des instructions exécutées et des utilisateurs qui les ont exécutées, nous vous recommandons d'activer l'audit avancé avec Aurora. Si vous devez également conserver des informations sur l'utilisation de la mémoire, nous vous recommandons de configurer la surveillance afin d'exporter et de stocker ces valeurs.

```
mysql> SELECT USER,EVENT_NAME,COUNT_STAR,MAX_TOTAL_MEMORY FROM
 performance_schema.events_statements_summary_by_user_by_event_name
ORDER BY MAX_CONTROLLED_MEMORY DESC LIMIT 5;

+----------------------+---------------------------+------------+------------------+
| USER                 | EVENT_NAME                | COUNT_STAR | MAX_TOTAL_MEMORY |
+----------------------+---------------------------+------------+------------------+
| destructive_operator | statement/sql/select      |          4 |        537450710 |
| rdsadmin             | statement/sql/select      |       4172 |          3290981 |
| rdsadmin             | statement/sql/show_tables |          2 |          3615821 |
| rdsadmin             | statement/sql/show_fields |          2 |          3459965 |
| rdsadmin             | statement/sql/show_status |         75 |          1914976 |
+----------------------+---------------------------+------------+------------------+
5 rows in set (0.00 sec)

mysql> SELECT HOST,EVENT_NAME,COUNT_STAR,MAX_TOTAL_MEMORY FROM
 performance_schema.events_statements_summary_by_host_by_event_name
```

```
WHERE HOST != 'localhost' AND COUNT_STAR>0 ORDER BY MAX_CONTROLLED_MEMORY DESC LIMIT 5;

+------------+----------------------+------------+------------------+
| HOST       | EVENT_NAME           | COUNT_STAR | MAX_TOTAL_MEMORY |
+------------+----------------------+------------+------------------+
| 10.0.8.231 | statement/sql/select |          4 |        537450710 |
+------------+----------------------+------------+------------------+
1 row in set (0.00 sec)
```

Exemple 3 : la mémoire libérable diminue continuellement et n'est pas récupérée

Le moteur de base de données InnoDB utilise divers événements de suivi de mémoire spécialisés pour différents composants. Ces événements spécifiques permettent un suivi granulaire de l'utilisation de la mémoire dans les principaux sous-systèmes InnoDB, par exemple :

- `memory/innodb/buf0buf` — Dédié à la surveillance des allocations de mémoire pour le pool de tampons InnoDB.
- `memory/innodb/ibuf0ibuf` — Suit spécifiquement les modifications de mémoire liées au tampon de modification d'InnoDB.

Pour identifier les principaux consommateurs de mémoire, nous pouvons interroger `sys.memory_global_by_current_bytes` :

```
mysql> SELECT event_name,current_alloc FROM sys.memory_global_by_current_bytes LIMIT
 10;

+-----------------------------------------------------------------+---------------+
| event_name                                                      | current_alloc |
+-----------------------------------------------------------------+---------------+
| memory/innodb/memory                                            | 5.28 GiB      |
| memory/performance_schema/table_io_waits_summary_by_index_usage | 495.00 MiB    |
| memory/performance_schema/table_shares                          | 488.00 MiB    |
| memory/sql/TABLE_SHARE::mem_root                                | 388.95 MiB    |
| memory/innodb/std                                               | 226.88 MiB    |
| memory/innodb/fil0fil                                           | 198.49 MiB    |
| memory/sql/binlog_io_cache                                      | 128.00 MiB    |
| memory/innodb/mem0mem                                           | 96.82 MiB     |
| memory/innodb/dict0dict                                         | 96.76 MiB     |
| memory/performance_schema/rwlock_instances                      | 88.00 MiB     |
+-----------------------------------------------------------------+---------------+
10 rows in set (0.00 sec)
```

Les résultats révèlent que `memory/innodb/memory` est le principal consommateur, lequel utilise 5,28 Gio sur la mémoire actuellement allouée. Cet événement sert de catégorie pour les allocations de mémoire entre divers composants InnoDB non associés à des événements d'attente plus spécifiques, comme `memory/innodb/buf0buf` mentionné précédemment.

Après avoir établi que les composants InnoDB sont les principaux consommateurs de mémoire, nous pouvons approfondir notre recherche à l'aide de la commande MySQL suivante :

```
SHOW ENGINE INNODB STATUS \G;
```

La commande SHOW ENGINE INNODB STATUS fournit un rapport d'état complet pour le moteur de stockage InnoDB, y compris des statistiques détaillées d'utilisation de la mémoire pour les différents composants InnoDB. Il contribue à identifier les structures ou opérations InnoDB spécifiques qui consomment le plus de mémoire. Pour plus d'informations, consultez InnoDB In-Memory Structures dans la documentation MySQL.

En analysant la section `BUFFER POOL AND MEMORY` du rapport d'état d'InnoDB, nous constatons que 5 051 647 748 octets (4,7 Gio) sont alloués au cache d'objets du dictionnaire, ce qui représente 89 % de la mémoire suivie par `memory/innodb/memory`.

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 0
Dictionary memory allocated 5051647748
Buffer pool size 170512
Free buffers 142568
Database pages 27944
Old database pages 10354
Modified db pages 6
Pending reads 0
```

Le cache d'objets du dictionnaire est un cache global partagé qui stocke en mémoire les objets du dictionnaire de données qui ont déjà été consultés afin de permettre leur réutilisation et d'améliorer les performances. L'allocation de mémoire élevée pour le cache d'objets du dictionnaire suggère qu'un grand nombre d'objets de base de données se trouve dans le cache.

Maintenant que nous savons que le cache du dictionnaire de données est l'un des principaux consommateurs, nous allons l'inspecter pour détecter les tables ouvertes. Pour connaître le

nombre de tables ouvertes dans le cache de définition de table, interrogez la variable d'état globale open_table_definitions.

```
mysql> show global status like 'open_table_definitions';

+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| Open_table_definitions | 20000 |
+------------------------+-------+
1 row in set (0.00 sec)
```

Pour plus d'informations, consultez How MySQL Opens and Closes Tables dans la documentation MySQL.

Vous pouvez restreindre le nombre de définitions de tables dans le cache du dictionnaire de données en limitant le paramètre `table_definition_cache` dans le groupe de paramètres du cluster de bases de données ou de l'instance de base de données. Pour Aurora MySQL, cette valeur joue le rôle de limite flexible pour le nombre de tables dans le cache de définition de table. La valeur par défaut dépend de la classe d'instance et est définie comme suit :

```
LEAST({DBInstanceClassMemory/393040}, 20000)
```

Lorsque le nombre de tables dépasse la limite `table_definition_cache`, un mécanisme basé sur les éléments les moins récemment utilisés, ou LRU, évacue et supprime des tables du cache. Toutefois, les tables impliquées dans les relations de clé étrangère ne sont pas placées dans la liste LRU, ce qui empêche leur suppression.

Dans notre scénario actuel, nous exécutons FLUSH TABLES pour vider le cache de définition de table. Cette action entraîne une baisse significative de la variable d'état globale Open_Table_Definitions, qui passe de 20 000 à 12, comme indiqué ici :

```
mysql> show global status like 'open_table_definitions';

+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| Open_table_definitions | 12    |
+------------------------+-------+
1 row in set (0.00 sec)
```

Malgré cette réduction, nous observons que l'allocation de mémoire pour `memory/innodb/memory` de 5,18 Gio reste élevée, et que la mémoire allouée au dictionnaire reste également inchangée. Cela ressort clairement des résultats de requête suivants :

```
mysql> SELECT event_name,current_alloc FROM sys.memory_global_by_current_bytes LIMIT
 10;

+-----------------------------------------------------------------+---------------+
| event_name                                                      | current_alloc |
+-----------------------------------------------------------------+---------------+
| memory/innodb/memory                                            | 5.18 GiB      |
| memory/performance_schema/table_io_waits_summary_by_index_usage | 495.00 MiB    |
| memory/performance_schema/table_shares                          | 488.00 MiB    |
| memory/sql/TABLE_SHARE::mem_root                                | 388.95 MiB    |
| memory/innodb/std                                               | 226.88 MiB    |
| memory/innodb/fil0fil                                           | 198.49 MiB    |
| memory/sql/binlog_io_cache                                      | 128.00 MiB    |
| memory/innodb/mem0mem                                           | 96.82 MiB     |
| memory/innodb/dict0dict                                         | 96.76 MiB     |
| memory/performance_schema/rwlock_instances                      | 88.00 MiB     |
+-----------------------------------------------------------------+---------------+
10 rows in set (0.00 sec)
```

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total large memory allocated 0
Dictionary memory allocated 5001599639
Buffer pool size 170512
Free buffers 142568
Database pages 27944
Old database pages 10354
Modified db pages 6
Pending reads 0
```

Cette utilisation constamment élevée de la mémoire peut être attribuée aux tables impliquées dans des relations de clé étrangère. Ces tables ne sont pas placées dans la liste LRU de tables à supprimer, ce qui explique pourquoi l'allocation de mémoire reste élevée même après avoir vidé le cache de définition de table.

Pour résoudre ce problème :

1. Passez en revue et optimisez le schéma de votre base de données, en particulier les relations de clé étrangère.
2. Envisagez de passer à une classe d'instance de base de données plus vaste qui dispose de plus de mémoire pour héberger les objets de votre dictionnaire.

En suivant ces étapes et en comprenant les modèles d'allocation de mémoire, vous gérerez plus efficacement l'utilisation de la mémoire dans votre instance de base de données Aurora MySQL et éviterez les problèmes de performances potentiels dus à la pression de la mémoire.

Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL

Le paramètre de niveau instance Aurora MySQL `aurora_oom_response` peut autoriser l'instance de base de données à surveiller la mémoire système et à estimer la mémoire utilisée par différentes déclarations et connexions. Si le système manque de mémoire, il peut effectuer une série d'actions pour tenter de libérer cette mémoire. L'objectif est d'éviter le redémarrage de la base de données en raison de problèmes de mémoire insuffisante. Ce paramètre de niveau instance accepte une chaîne d'actions (séparées par des virgules) qu'une instance de base de données doit effectuer lorsque sa mémoire est faible. Le paramètre `aurora_oom_response` est pris en charge pour Aurora MySQL versions 2 et 3.

Les valeurs suivantes, ainsi que leurs combinaisons, peuvent être utilisées pour le paramètre `aurora_oom_response`. Une chaîne vide signifie qu'aucune action n'est effectuée et revient à désactiver la fonctionnalité. La base de données peut donc faire l'objet de redémarrages en raison d'une mémoire insuffisante.

- `decline` : refuse les nouvelles requêtes une fois que l'instance de base de données manque de mémoire.
- `kill_connect` : ferme les connexions de base de données qui consomment une grande quantité de mémoire et met fin aux transactions en cours et aux instructions DDL (Data Definition Language). Cette réponse n'est pas prise en charge pour Aurora MySQL version 2.

  Pour plus d'informations, consultez KILL Statement dans la documentation MySQL.
- `kill_query` : arrête les requêtes dans l'ordre décroissant de leur consommation de mémoire jusqu'à ce que la mémoire de l'instance passe au-dessus du seuil bas. Les instructions DDL ne sont pas arrêtées.

  Pour plus d'informations, consultez KILL Statement dans la documentation MySQL.

- `print` : imprime uniquement les requêtes qui consomment une grande quantité de mémoire.
- `tune` : affine les caches de table interne pour restituer de la mémoire au système. Aurora MySQL réduit la mémoire utilisée pour les caches tels que `table_open_cache` et `table_definition_cache` dans des conditions de faible mémoire. Finalement, Aurora MySQL rétablit l'utilisation de la mémoire à des conditions normales lorsque le système n'est plus à court de mémoire.

  Pour plus d'informations, consultez table_open_cache et table_definition_cache dans la documentation MySQL.
- `tune_buffer_pool` : diminue la taille du pool de tampons afin de libérer de la mémoire et de la rendre disponible pour que le serveur de base de données puisse traiter les connexions. Cette réponse est prise en charge pour Aurora MySQL versions 3.06 et ultérieures.

  Vous devez associer `tune_buffer_pool` avec `kill_query` ou `kill_connect` dans la valeur du paramètre `aurora_oom_response`. Dans le cas contraire, le redimensionnement du pool de tampons ne se produira pas, même si vous incluez `tune_buffer_pool` dans la valeur du paramètre.

Dans les versions d'Aurora MySQL antérieures à 3.06, pour les classes d'instance de base de données dont la mémoire est inférieure ou égale à 4 Gio, lorsque l'instance est soumise à une pression de mémoire, les actions par défaut incluent `print`, `tune`, `decline` et `kill_query`. Pour les classes d'instance de base de données dont la mémoire est supérieure à 4 Gio, la valeur du paramètre est vide par défaut (désactivé).

Dans Aurora MySQL 3.06 et versions ultérieures, pour les classes d'instance de base de données dont la mémoire est inférieure ou égale à 4 Gio, Aurora MySQL ferme également les connexions les plus gourmandes en mémoire (`kill_connect`). Pour les classes d'instance de base de données dont la mémoire est supérieure à 4 Gio, la valeur du paramètre par défaut est `print`.

Dans Aurora MySQL 3.09 et versions ultérieures, pour les classes d'instance de base de données dont la mémoire est supérieure à 4 Gio, la valeur du paramètre par défaut est `print,decline,kill_connect`.

Si vous rencontrez fréquemment des problèmes de mémoire insuffisante, l'utilisation de la mémoire peut être surveillée à l'aide de tableaux récapitulatifs de la mémoire lorsque `performance_schema` est activé.

Pour les métriques Amazon CloudWatch relatives à un problème de mémoire insuffisante, consultez Métriques de niveau instance pour Amazon Aurora. Pour les variables d'état globales liées à une mémoire insuffisante, consultez Variables d'état globales Aurora MySQL.

Journalisation pour les bases de données Aurora MySQL

Les journaux Aurora MySQL fournissent des informations essentielles sur l'activité et les erreurs de base de données. En les activant, vous pouvez identifier et résoudre les problèmes, comprendre les performances de la base de données et en auditer l'activité. Nous vous recommandons d'activer ces journaux pour toutes vos instances de base de données Aurora MySQL afin de garantir des performances et une disponibilité optimales des bases de données. Les types de journaux suivants peuvent être activés. Chaque journal contient des informations spécifiques qui peuvent contribuer à identifier des impacts sur le traitement de la base de données.

- Erreur : Aurora MySQL écrit dans le journal d'erreurs uniquement au moment du démarrage, de l'arrêt et lorsqu'une erreur survient. Une instance de base de données peut fonctionner pendant des heures ou des jours sans qu'aucune nouvelle entrée soit écrite dans le journal des erreurs. Si aucune entrée récente ne figure, cela signifie que le serveur n'a pas rencontré d'erreur justifiant une entrée de journal. La journalisation des erreurs est activée par défaut. Pour plus d'informations, consultez Journaux des erreurs Aurora MySQL.
- Général : le journal général fournit des informations détaillées sur l'activité de la base de données, y compris toutes les instructions SQL exécutées par le moteur de base de données. Pour plus d'informations sur l'activation de la journalisation générale et la définition des paramètres de journalisation, consultez Journal des requêtes lentes et journal général Aurora MySQL et Journal des requêtes générales dans la documentation MySQL.

  > **Note**
  >
  > Les journaux généraux peuvent devenir très volumineux et consommer de l'espace de stockage. Pour plus d'informations, consultez Renouvellement et rétention des journaux pour Aurora MySQL.

- Requêtes lentes : le journal des requêtes lentes se compose des instructions SQL qui ont pris plus de long_query_time secondes pour s'exécuter et qui nécessitent l'examen d'au moins min_examined_row_limit lignes. Vous pouvez utiliser le journal des requêtes lentes pour rechercher les requêtes dont l'exécution prend du temps et qui sont donc de bonnes candidates pour l'optimisation.

  La valeur par défaut de `long_query_time` est 10 secondes. Nous vous recommandons de commencer par une valeur élevée pour identifier les requêtes les plus lentes, puis de la réduire petit à petit pour optimiser le réglage.

Vous pouvez également utiliser des paramètres connexes, tels que `log_slow_admin_statements` et `log_queries_not_using_indexes`. Comparez `rows_examined` à `rows_returned`. Si la valeur de `rows_examined` est bien supérieure à `rows_returned`, ces requêtes peuvent potentiellement être bloquantes.

Dans Aurora MySQL version 3, vous pouvez activer `log_slow_extra` pour obtenir plus de détails. Pour plus d'informations, consultez Contenu du journal des requêtes lentes dans la documentation MySQL. Vous pouvez également modifier `long_query_time` au niveau de la session pour déboguer l'exécution des requêtes de manière interactive, ce qui est particulièrement utile si `log_slow_extra` est activé globalement.

Pour plus d'informations sur l'activation de la journalisation des requêtes lentes et la définition des paramètres de journalisation, consultez Journal des requêtes lentes et journal général Aurora MySQL et Journal des requêtes lentes dans la documentation MySQL.

- Audit ; le journal d'audit surveille et enregistre l'activité de la base de données. La journalisation d'audit pour Aurora MySQL se nomme « audit avancé ». Pour activer l'audit avancé, vous devez définie certains paramètres de cluster de bases de données. Pour plus d'informations, consultez Utilisation de l'Audit avancé avec un cluster de bases de données Amazon Aurora MySQL.
- Binaire : le journal binaire (binlog) contient les événements qui décrivent les modifications apportées à la base de données, telles que les opérations de création de tables ou de modification des données des tables. Il contient également les événements relatifs aux instructions susceptibles d'apporter des modifications (par exemple, une instruction DELETE ne correspondant à aucune ligne), à moins que la journalisation basée sur les lignes ne soit utilisée. Le journal binaire contient également des informations sur le temps nécessaire à chaque instruction pour mettre à jour les données.

  L'exécution d'un serveur lorsque la journalisation binaire est activée ralentit légèrement les performances. Toutefois, les avantages du journal binaire, qui vous permet de configurer la réplication et d'effectuer des opérations de restauration, compensent généralement cette baisse mineure des performances.

  > **Note**
  >
  > Aurora MySQL ne nécessite pas de journalisation binaire pour les opérations de restauration.

Pour plus d'informations sur l'activation de la journalisation binaire et la définition du format de journal binaire, consultez Configuration d'Aurora MySQL pour la journalisation des bases de données mono-AZ et Journal binaire dans la documentation MySQL.

Vous pouvez publier les journaux d'erreurs, les journaux généraux, les journaux de requêtes lentes et les journaux d'audit dans Amazon CloudWatch Logs. Pour plus d'informations, consultez Publication des journaux de base de données dans Amazon CloudWatch Logs.

pt-query-digest est un autre outil utile pour résumer les journaux de requêtes lentes, les journaux généraux et les journaux binaires.

Résolution des problèmes de connexion aux bases de données Aurora MySQL

Garantir une connectivité fiable entre vos applications et votre instance de base de données RDS est crucial pour le bon fonctionnement de vos charges de travail. Cependant, les problèmes de connectivité peuvent survenir en raison de divers facteurs, tels que les configurations réseau, les problèmes d'authentification ou les contraintes de ressources. Ce guide vise à fournir une approche complète pour résoudre les problèmes de connectivité rencontrés avec Aurora MySQL.

Table des matières

Identification des problèmes de connectivité des bases de données avec Aurora MySQL

L'identification de la catégorie spécifique du problème de connectivité permet de déterminer les causes potentielles et d'orienter le processus de résolution des problèmes. Chaque catégorie peut nécessiter des approches et des techniques différentes pour le diagnostic et la résolution. Les problèmes de connectivité des bases de données peuvent généralement être classés dans les catégories suivantes.

Erreurs et exceptions de connexion

> Des erreurs et exceptions de connexion peuvent se produire pour diverses raisons, telles qu'une chaîne de connexion incorrecte, un échec d'authentification, une interruption du réseau ou un problème sur le serveur de base de données. Parmi les causes figurent une configuration erronée

des paramètres de connexion, des identifiants non valides, une défaillance du réseau ou un crash/redémarrage du serveur de base de données. Les groupes de sécurité mal configurés, les paramètres du cloud privé virtuel (VPC), les listes de contrôle d'accès réseau ACLs () et les tables de routage associées aux sous-réseaux peuvent également entraîner des problèmes de connexion.

Limite de connexion atteinte

Ce problème survient lorsque le nombre de connexions simultanées au serveur de base de données dépasse la limite maximale autorisée. Les serveurs de bases de données sont généralement soumis à une limite maximale de connexions configurable, définie par le paramètre max_connections dans les groupes de paramètres de cluster et d'instance. En imposant une limite de connexion, le serveur de base de données s'assure qu'il dispose de suffisamment de ressources (par exemple, mémoire, CPU et descripteurs de fichiers) pour gérer efficacement les connexions existantes et fournir des performances acceptables. Parmi les causes citons des fuites de connexion dans l'application, un regroupement de connexions inefficace ou une augmentation inattendue du nombre de demandes de connexion.

Délai d'expiration de connexion

Les délais d'expiration de connexion se produisent lorsque l'application cliente ne parvient pas à établir une connexion avec le serveur de base de données dans le délai imparti. Parmi les causes les plus fréquentes figurent des dysfonctionnements réseau, une saturation du serveur, des règles de pare-feu ou une configuration incorrecte des paramètres de connexion.

Délai d'expiration de connexion inactive

Les connexions inactives qui restent inactives pendant une période prolongée peuvent être fermées automatiquement par le serveur de base de données pour économiser les ressources. Ce délai d'expiration est généralement configurable à l'aide de `wait_timeout` et `interactive_timeout parameters`. Il doit être ajusté en fonction des modèles d'utilisation des connexions de l'application. Les causes peuvent être liées à une logique d'application qui laisse des connexions inactives trop longtemps, ou à une gestion inadéquate des connexions.

Déconnexion intermittente des connexions existantes

Cette catégorie d'erreurs fait référence à un scénario dans lequel les connexions établies entre une application cliente et la base de données sont interrompues de façon inattendue ou déconnectées à intervalles irréguliers, alors qu'elles sont actives et en cours d'utilisation. Ces déconnexions se produisent de manière intermittente, c'est-à-dire à des intervalles irréguliers et de façon non prévisible. Voici les causes possibles :

- Problèmes liés au serveur de base de données, tels que les redémarrages ou les basculements
- Mauvaise gestion des connexions d'applications
- Problèmes d'équilibrage de charge et de proxy
- Instabilité réseau
- Problèmes liés aux composants tiers ou aux intergiciels impliqués dans le chemin de connexion
- Délais d'expiration d'exécution de requête
- Contraintes de ressources côté serveur ou côté client

Il est essentiel d'identifier la cause première à l'aide d'une surveillance, d'une journalisation et d'une analyse approfondies. La mise en œuvre de mécanismes adaptés de gestion des erreurs, de regroupement des connexions et de nouvelles tentatives peut contribuer à réduire l'impact de ces déconnexions intermittentes sur les fonctionnalités de l'application et sur l'expérience utilisateur.

Collecte de données sur les problèmes de connectivité pour Aurora MySQL

La collecte de données complètes relatives aux composants de l'application, de la base de données, du réseau et de l'infrastructure est essentielle pour résoudre efficacement les problèmes de connectivité entre une application et une base de données Aurora MySQL. En collectant les journaux, les configurations et les données de diagnostic pertinents, vous obtenez des informations précieuses qui peuvent vous aider à identifier la cause première des problèmes de connectivité et à vous guider vers une résolution appropriée.

Les journaux et les configurations réseau, tels que les règles des groupes de sécurité, les paramètres de VPC et les tables de routage, sont essentiels pour identifier d'éventuels goulots d'étranglement ou des erreurs de configuration réseau susceptibles d'empêcher la connexion entre l'application et la base de données. En analysant ces composants réseau, vous pouvez vous assurer que les ports nécessaires sont ouverts, que les adresses IP sont autorisées et que les configurations de routage sont correctement configurées.

Horodatages

Enregistrez les horodatages exacts lorsque les problèmes de connectivité surviennent. Cette approche permet d'identifier des modèles ou de corréler les problèmes avec d'autres événements ou activités.

Journaux du moteur de base de données

Outre les journaux généraux de la base de données, consultez les journaux du moteur de base de données (par exemple, le journal d'erreurs MySQL et le journal des requêtes lentes) pour obtenir toute information pertinente ou erreur susceptible d'être liée aux problèmes de connectivité intermittents. Pour plus d'informations, consultez Journalisation pour les bases de données Aurora MySQL.

Journaux d'applications clientes

Collectez des journaux détaillés à partir des applications clientes qui se connectent à la base de données. Les journaux d'application offrent une visibilité sur les tentatives de connexion, les erreurs et toute autre information pertinente du point de vue de l'application, susceptibles de révéler des problèmes liés aux chaînes de connexion, aux informations d'identification d'authentification ou à la gestion des connexions dans l'application.

Les journaux de base de données, quant à eux, fournissent des informations sur les erreurs liées à la base de données, la lenteur des requêtes ou les événements susceptibles de contribuer aux problèmes de connectivité. Pour plus d'informations, consultez Journalisation pour les bases de données Aurora MySQL.

Variables d'environnement du client

Vérifiez si des variables d'environnement ou des paramètres de configuration côté client peuvent affecter la connexion à la base de données, tels que les paramètres du proxy, SSL/TLS les paramètres ou toute autre variable pertinente.

Versions de la bibliothèque cliente

Assurez-vous que le client utilise les dernières versions de tous les pilotes de base de données, bibliothèques ou frameworks utilisés pour la connectivité aux bases de données. Les versions obsolètes peuvent présenter des problèmes connus ou des problèmes de compatibilité.

Capture réseau du client

Effectuez une capture réseau côté client à l'aide d'un outil tel que Wireshark ou `tcpdump` lorsque des problèmes de connectivité surviennent. Cette opération peut contribuer à identifier les problèmes ou irrégularités liés au réseau côté client.

Topologie réseau du client

Il est important de bien comprendre la topologie réseau du client, notamment la présence éventuelle de pare-feux, d'équilibreurs de charge ou d'autres composants (tels que RDS Proxy ou Proxy SQL) qui se connectent à la base de données à la place du client.

Paramètres du système d’exploitation client

Vérifiez les paramètres du système d’exploitation du client susceptibles d’affecter la connectivité réseau, tels que les règles de pare-feu, les paramètres de l’adaptateur réseau et tout autre paramètre pertinent.

Configuration du regroupement de connexions

Si vous utilisez un mécanisme de regroupement de connexions dans votre application, passez en revue les paramètres de configuration et surveillez les métriques du regroupement (par exemple, les connexions actives, les connexions inactives et les délais d’expiration des connexions) pour vous assurer que le groupe fonctionne correctement. Vérifiez également les paramètres du regroupement, tels que la taille maximale du groupe, la taille minimale du groupe et les paramètres de validation de connexion, pour vous assurer qu’ils sont correctement configurés.

Chaîne de connexion

La chaîne de connexion inclut généralement des paramètres tels que le nom d’hôte ou de point de terminaison, le numéro de port, le nom de la base de données et les informations d’identification d’authentification. L’analyse de la chaîne de connexion permet d’identifier les erreurs de configuration potentielles ou les paramètres incorrects susceptibles de provoquer des problèmes de connectivité. Par exemple, un nom d’hôte ou un numéro de port incorrect peut empêcher le client d’accéder à l’instance de base de données, tandis que des informations d’identification d’authentification non valides peuvent entraîner des échecs d’authentification et des rejets de connexion. En outre, la chaîne de connexion peut révéler des problèmes liés au regroupement des connexions, aux délais d’expiration ou à d’autres paramètres spécifiques à la connexion susceptibles de contribuer aux problèmes de connectivité. La fourniture de la chaîne de connexion complète utilisée par l’application cliente permet d’identifier les erreurs de configuration sur le client.

Métriques de base de données

Surveillez les indicateurs de base de données tels que l'utilisation du processeur, de la mémoire et du disque I/O lorsque des problèmes de connectivité surviennent. Elles permettent de déterminer si l’instance de base de données est confrontée à des problèmes de contention des ressources ou de performances.

Version du moteur de base de données

Notez la version du moteur de base de données Aurora MySQL. AWS publie régulièrement des mises à jour visant à résoudre des problèmes connus, corriger des failles de sécurité et améliorer les performances. Par conséquent, nous vous recommandons vivement de passer aux dernières

versions disponibles, car ces mises à jour incluent souvent des corrections de bogues et des améliorations spécifiquement liées à la connectivité, aux performances et à la stabilité. Fournir les informations de version de la base de données, ainsi que les autres informations collectées, peut aider Support à diagnostiquer et à résoudre efficacement les problèmes de connectivité.

Métriques de réseau

Collectez des métriques de réseau tels que la latence, la perte de paquets et le débit lorsque des problèmes de connectivité surviennent. Des outils tels que `ping`, `traceroute` et les outils de surveillance réseau permettent de recueillir ces données.

Détails sur la source et le client

Déterminez les adresses IP des serveurs d'applications, des équilibreurs de charge ou de tout autre composant qui initie les connexions à la base de données. Il peut s'agir d'une seule adresse IP ou d'une plage d'adresses IP (notation CIDR). Si la source est une EC2 instance Amazon, il est également utile de vérifier le type d'instance, la zone de disponibilité, l'ID de sous-réseau et les groupes de sécurité associés à l'instance, ainsi que les détails de l'interface réseau tels que l'adresse IP privée et l'adresse IP publique.

En analysant minutieusement les données collectées, vous pouvez identifier les erreurs de configuration, les contraintes de ressources, les perturbations du réseau ou les autres problèmes sous-jacents à l'origine des problèmes de connectivité intermittents ou persistants. Ces informations vous permettent de prendre des mesures ciblées, telles que l'ajustement des configurations, la résolution des problèmes de réseau ou la gestion des connexions au niveau de l'application.

Surveillance des connexions aux bases de données pour Aurora MySQL

Pour surveiller et résoudre les problèmes de connectivité, vous pouvez utiliser les métriques et fonctionnalités suivantes.

CloudWatch métriques

- `CPUUtilization` : l'utilisation élevée de l'UC sur l'instance de base de données peut ralentir l'exécution des requêtes, ce qui peut entraîner des délais d'expiration ou des rejets de connexion.
- `DatabaseConnections` : surveille le nombre de connexions actives à l'instance de base de données. Un nombre élevé de connexions proche de la valeur maximale configurée peut indiquer des problèmes de connectivité potentiels ou une saturation du regroupement de connexions.

- `FreeableMemory` : la faible quantité de mémoire disponible peut entraîner des problèmes de performances et de connectivité en raison de contraintes de ressources.
- `NetworkReceiveThroughput` et `NetworkTransmitThroughput` : des pics ou des baisses inhabituels du débit réseau peuvent indiquer des problèmes de connectivité ou des goulots d'étranglement du réseau.

Métriques de Performance Insights

Pour résoudre les problèmes de connectivité dans Aurora MySQL à l'aide de Performance Insights, analysez les métriques de base de données suivantes :

- Aborted_clients
- Aborted_connects
- Connexions
- max_connections
- Threads_connected
- Threads_created
- Threads_running

Ces métriques peuvent vous aider à identifier les goulots d'étranglement de connexion, détecter les problèmes de réseau ou d'authentification, optimiser le regroupement des connexions et garantir une gestion efficace des threads. Pour plus d'informations, consultez Compteurs Performance Insights pour Aurora MySQL.

Fonctionnalité de Performance Insights

- Charge de la base de données : visualisez la charge de la base de données au fil du temps et corrélez-la aux problèmes de connectivité ou à la dégradation des performances.
- Statistiques SQL : analysez les statistiques SQL pour identifier les requêtes ou les opérations de base de données inefficaces susceptibles d'entraîner des problèmes de connectivité.
- Requêtes principales : identifiez et analysez les requêtes les plus gourmandes en ressources, en vue d'identifier les problèmes de performance potentiels ou les requêtes de longue durée susceptibles de provoquer des problèmes de connectivité.

En surveillant ces métriques et en tirant parti de Performance Insights, vous pouvez acquérir une meilleure visibilité sur les performances de l'instance de base de données, l'utilisation des ressources et les goulots d'étranglement potentiels susceptibles de causer des problèmes de connectivité. Par exemple :

- Un niveau `DatabaseConnections` proche de la limite maximale peut indiquer une saturation du regroupement de connexions ou une mauvaise gestion des connexions, ce qui peut entraîner des problèmes de connectivité.
- Un niveau `CPUUtilization` élevé ou `FreeableMemory` faible peut indiquer des contraintes de ressources, susceptibles de ralentir l'exécution des requêtes et de provoquer des délais d'expiration ou des rejets de connexion.
- L'analyse des Requêtes principales et des Statistiques SQL permet d'identifier les requêtes inefficaces ou gourmandes en ressources susceptibles de contribuer aux problèmes de connectivité.

En outre, la surveillance CloudWatch des journaux et la configuration d'alarmes peuvent vous aider à identifier les problèmes de connectivité et à y répondre de manière proactive avant qu'ils ne s'aggravent.

Il est important de noter que si ces métriques et outils peuvent fournir des informations précieuses, ils doivent être utilisés conjointement avec d'autres processus de résolution des problèmes. En examinant également les configurations réseau, les règles des groupes de sécurité et la gestion des connexions au niveau de l'application, vous pouvez diagnostiquer et résoudre de manière exhaustive les problèmes de connectivité liés aux instances de base de données Aurora MySQL.

Surveillance supplémentaire pour Aurora MySQL

CloudWatch métriques

- `AbortedClients` : suit le nombre de connexions client qui n'ont pas été fermées correctement.
- `AuroraSlowConnectionHandleCount` : suit le nombre d'opérations de gestion des connexions lentes, en indiquant les problèmes de connectivité potentiels ou les goulots d'étranglement liés aux performances.
- `AuroraSlowHandshakeCount` : indique le nombre d'opérations d'établissement de liaison lentes, également susceptible de révéler des problèmes de connectivité.
- `ConnectionAttempts` : indique le nombre de tentatives de connexion effectuées à l'instance de base de données Aurora MySQL.

Variables d’état globales

`Aurora_external_connection_count` : indique le nombre de connexions de base de données à l’instance de base de données, à l’exclusion des connexions au service RDS utilisées pour la surveillance de l’état de la base de données.

En surveillant ces métriques et ces variables d’état globales, vous pouvez acquérir une meilleure visibilité sur les modèles de connexion, les erreurs et les goulots d’étranglement potentiels susceptibles de provoquer des problèmes de connectivité avec votre instance Amazon Aurora MySQL.

Par exemple, un nombre élevé de `AbortedClients` ou `AuroraSlowConnectionHandleCount` peut indiquer des problèmes de connectivité.

En outre, la configuration d' CloudWatch alarmes et de notifications peut vous aider à identifier les problèmes de connectivité et à y répondre de manière proactive avant qu'ils ne s'aggravent et n'affectent les performances de votre application.

Codes d’erreur de connectivité pour Aurora MySQL

Voici quelques erreurs de connectivité courantes concernant les bases de données Aurora MySQL, ainsi que leurs codes d’erreur et leurs explications.

Error Code 1040: Too many connections

Cette erreur se produit lorsque le client essaie d’établir un nombre de connexions supérieur au maximum autorisé par le serveur de base de données. Les causes possibles sont notamment les suivantes :

- Mauvaise configuration du regroupement de connexions : si vous utilisez un mécanisme de regroupement de connexions, assurez-vous que la taille maximale de ce dernier n’est pas trop élevée et que les connexions sont correctement rétablies dans le regroupement.
- Configuration de l’instance de base de données : vérifiez le paramètre de connexions maximales autorisées pour l’instance de base de données et ajustez-le si nécessaire en définissant le paramètre `max_connections`.
- Haute simultanéité : la connexion simultanée d’un grand nombre de clients ou d’applications à la base de données peut entraîner l’atteinte du seuil maximal de connexions autorisées.

Error Code 1045: Access denied for user '...'@'...' (using password: YES/NO)

Cette erreur indique un échec d'authentification lors de la tentative de connexion à la base de données. Les causes possibles sont notamment les suivantes :

- Compatibilité des plug-ins d'authentification : vérifiez si le plug-in d'authentification utilisé par le client est compatible avec le mécanisme d'authentification du serveur de base de données.
- Nom d'utilisateur ou mot de passe incorrect : vérifiez que le nom d'utilisateur et le mot de passe utilisés dans la chaîne de connexion ou le mécanisme d'authentification sont corrects.
- Autorisations utilisateur : assurez-vous que l'utilisateur dispose des autorisations nécessaires pour se connecter à l'instance de base de données depuis l'hôte ou le réseau spécifié.

Error Code 1049: Unknown database '...'

Cette erreur indique que le client tente de se connecter à une base de données qui n'existe pas sur le serveur. Les causes possibles sont notamment les suivantes :

- Base de données non créée : assurez-vous que la base de données spécifiée a été créée sur le serveur de base de données.
- Nom de base de données incorrect : vérifiez l'exactitude du nom de base de données utilisé dans la chaîne de connexion ou dans la requête.
- Autorisations utilisateur : vérifiez que l'utilisateur dispose des autorisations nécessaires pour accéder à la base de données spécifiée.

Error Code 1153: Got a packet bigger than 'max_allowed_packet' bytes

Cette erreur se produit lorsque le client tente d'envoyer ou de recevoir des données qui dépassent la taille de paquet maximale autorisée par le serveur de base de données. Les causes possibles sont notamment les suivantes :

- Ensembles de requêtes ou de résultats volumineux : si vous exécutez des requêtes impliquant de grandes quantités de données, la limite de taille des paquets peut être dépassée.
- Paramètres de taille de paquet mal configurés : vérifiez le paramètre `max_allowed_packet` sur le serveur de base de données et ajustez-le si nécessaire.
- Problèmes de configuration réseau : assurez-vous que la configuration réseau (par exemple, la taille MTU) autorise les tailles de paquets requises.

Error Code 1226: User '...' has exceeded the 'max_user_connections' resource (current value: ...)

Cette erreur indique que l'utilisateur a dépassé le nombre maximal de connexions simultanées autorisées par le serveur de base de données. Les causes possibles sont notamment les suivantes :

- Mauvaise configuration du regroupement de connexions : si vous utilisez un mécanisme de regroupement de connexions, assurez-vous que la taille maximale de ce dernier n'est pas configurée au-delà de la limite de connexions autorisée pour l'utilisateur.
- Configuration de l'instance de base de données : vérifiez le paramètre `max_user_connections` pour l'instance de base de données et ajustez-le si nécessaire.
- Haute simultanéité : si plusieurs clients ou applications se connectent simultanément à la base de données en utilisant le même utilisateur, la limite de connexions spécifique à cet utilisateur peut être atteinte.

Error Code 2003: Can't connect to MySQL server on '...' (10061)

Cette erreur se produit généralement lorsque le client ne parvient pas à établir une TCP/IP connexion avec le serveur de base de données. Elle peut être due à divers problèmes, notamment :

- État de l'instance de base de données : assurez-vous que l'instance de base de données présente l'état `available` et qu'elle ne fait l'objet d'aucune opération de maintenance ou de sauvegarde.
- Règles de pare-feu : vérifiez si des pare-feux (système d'exploitation, réseau ou groupe de sécurité) bloquent la connexion sur le port spécifié (généralement 3306 pour MySQL).
- Nom d'hôte ou point de terminaison incorrect : assurez-vous que le nom d'hôte ou le point de terminaison utilisé dans la chaîne de connexion est correct et correspond à l'instance de base de données.
- Problèmes de connectivité réseau : vérifiez que l'ordinateur client peut accéder à l'instance de base de données via le réseau. Vérifiez l'existence d'éventuelles pannes réseau, de problèmes de routage ou de mauvaises configurations de VPC ou de sous-réseau.

Error Code 2005: Unknown MySQL server host '...' (11001)

Cette erreur se produit lorsque le client ne parvient pas à convertir le nom d'hôte ou le point de terminaison du serveur de base de données en adresse IP. Les causes possibles sont notamment les suivantes :

- Problèmes de résolution DNS : vérifiez que l'ordinateur client peut résoudre correctement le nom d'hôte à l'aide du DNS. Vérifiez les paramètres DNS, le cache DNS et essayez d'utiliser l'adresse IP au lieu du nom d'hôte.
- Nom d'hôte ou point de terminaison incorrect : vérifiez l'exactitude du nom d'hôte ou du point de terminaison utilisé dans la chaîne de connexion.

- Problèmes de configuration réseau : assurez-vous que la configuration réseau du client (par exemple, VPC, sous-réseau et tables de routage) autorise la résolution DNS et la connectivité à l'instance de base de données.

Error Code 2026: SSL connection error

Cette erreur se produit en cas de problème de SSL/TLS configuration ou de validation du certificat lors de la tentative de connexion. Les causes possibles sont notamment les suivantes :

- Expiration du certificat — Vérifiez si le SSL/TLS certificat utilisé par le serveur a expiré et doit être renouvelé.
- Problèmes de validation des certificats : vérifiez que le client est en mesure de valider correctement le SSL/TLS certificat du serveur et que le certificat est fiable.
- Problèmes de configuration réseau — Assurez-vous que la configuration réseau autorise les SSL/TLS connexions et ne bloque pas ou n'interfère pas avec le processus de SSL/TLS prise de contact.
- SSL/TLS configuration mismatch – Make sure that the SSL/TLSles paramètres (par exemple, les suites de chiffrement et les versions de protocole) du client et du serveur sont compatibles.

La compréhension des explications détaillées et des causes possibles de chaque code d'erreur permet de mieux diagnostiquer et corriger les problèmes de connectivité rencontrés avec les bases de données Aurora MySQL.

Recommandations de réglage des paramètres pour Aurora MySQL

Nombre maximal de connexions

Le réglage de ces paramètres permet d'éviter les problèmes de connexion provoqués par l'atteinte de la limite maximale de connexions autorisées. Assurez-vous que ces valeurs sont définies de manière appropriée en fonction des exigences de simultanéité et des contraintes de ressources de votre application.

- `max_connections` : ce paramètre spécifie le nombre maximum de connexions simultanées autorisées à l'instance de base de données.
- `max_user_connections` : ce paramètre peut être spécifié lors de la création et de la modification d'un utilisateur. Il définit le nombre maximal de connexions simultanées autorisées pour un compte utilisateur spécifique.

Taille du tampon réseau

L'augmentation de ces valeurs peut améliorer les performances du réseau, en particulier pour les charges de travail impliquant des transferts de données ou des jeux de résultats importants. Faites néanmoins attention, car l'augmentation de la taille des tampons entraîne une consommation mémoire plus importante.

- `net_buffer_length` : ce paramètre définit la taille initiale de la connexion client et des tampons de résultats, en équilibrant l'utilisation de la mémoire avec les performances des requêtes.
- `max_allowed_packet` : ce paramètre spécifie la taille maximale d'un seul paquet réseau qui peut être envoyé ou reçu par l'instance de base de données.

Compression réseau (côté client)

L'activation de la compression réseau peut réduire l'utilisation de la bande passante du réseau, mais elle peut augmenter la charge de l'UC du côté client et du côté serveur.

- `compress`— Ce paramètre active ou désactive la compression réseau pour les client/server communications.
- `compress_protocol` : ce paramètre spécifie le protocole de compression à utiliser pour les communications réseau.

Réglage des performances réseau

Le réglage de ces délais permet de mieux gérer les connexions inactives et d'éviter une surcharge des ressources, mais attention : des valeurs trop faibles peuvent entraîner des déconnexions prématurées.

- `interactive_timeout` : ce paramètre indique le nombre de secondes pendant lesquelles le serveur attend une activité sur une connexion interactive avant de la fermer.
- `wait_timeout` : ce paramètre détermine le nombre de secondes pendant lesquelles le serveur attend une activité sur une connexion non interactive avant de la fermer.

Paramètres de délai d'expiration réseau

Le réglage de ces délais permet de résoudre les problèmes liés à la lenteur ou à l'absence de réponse des connexions. Soyez toutefois prudent : des valeurs trop faibles peuvent provoquer des échecs de connexion prématurés.

- `net_read_timeout` : ce paramètre indique le nombre de secondes à attendre pour recevoir davantage de données d'une connexion avant de mettre fin à l'opération de lecture.

- `net_write_timeout` : ce paramètre détermine le nombre de secondes à attendre avant qu'un bloc soit écrit sur une connexion avant de mettre fin à l'opération d'écriture.

Exemples de résolution des problèmes de connexion aux bases de données pour Aurora MySQL

Les exemples suivants montrent comment identifier et résoudre les problèmes de connexion aux bases de données Aurora MySQL.

Exemple 1 : résolution des problèmes liés aux tentatives de connexion infructueuses

Les tentatives de connexion peuvent échouer pour plusieurs raisons, notamment les échecs d'authentification, les échecs de SSL/TLS prise de contact, la `max_connections` limite atteinte et les contraintes de ressources sur l'instance de base de données.

Vous pouvez suivre le nombre de connexions ayant échoué à partir de Performance Insights ou à l'aide de la commande suivante.

```
mysql> show global status like 'aborted_connects';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| Aborted_connects | 7     |
+------------------+-------+
1 row in set (0.00 sec)
```

Si le nombre `Aborted_connects` augmente au fil du temps, il est possible que l'application rencontre des problèmes de connectivité intermittents.

Vous pouvez utiliser Aurora Advanced Auditing pour enregistrer les connexions et les déconnexions des clients. Vous pouvez le faire en définissant les paramètres suivants dans le groupe de paramètres de cluster de bases de données :

- `server_audit_logging` = `1`
- `server_audit_events` = `CONNECT`

Voici un extrait des journaux d'audit relatifs à un échec de connexion.

```
1728498527380921,auora-mysql-node1,user_1,172.31.49.222,147189,0,FAILED_CONNECT,,,1045
```

```
1728498527380940,auora-mysql-node1,user_1,172.31.49.222,147189,0,DISCONNECT,,,0
```

Où :

- `1728498527380921` : l'horodatage de l'époque à laquelle l'échec de connexion s'est produit
- `aurora-mysql-node1` : l'identifiant d'instance du nœud du cluster Aurora MySQL sur lequel la connexion a échoué
- `user_1` : le nom de l'utilisateur de base de données pour lequel la connexion a échoué
- `172.31.49.222` : l'adresse IP privée du client à partir duquel la connexion a été établie
- `147189` : l'identifiant de connexion de l'échec de connexion
- `FAILED_CONNECT` : indique que la connexion a échoué.
- `1045` : le code de retour. Une valeur différente de zéro indique une erreur. Dans ce cas, `1045` correspond à un accès refusé.

Pour plus d'informations, consultez Codes d'erreur serveur et Codes d'erreur client dans la documentation MySQL.

Vous pouvez également examiner les journaux d'erreurs d'Aurora MySQL pour rechercher d'éventuels messages d'erreur associés, par exemple :

```
2024-10-09T19:26:59.310443Z 220 [Note] [MY-010926] [Server] Access denied for user
 'user_1'@'172.31.49.222' (using password: YES) (sql_authentication.cc:1502)
```

Exemple 2 : résolution des problèmes de déconnexion anormale des clients

Vous pouvez suivre le nombre de déconnexions anormales des clients à partir de Performance Insights ou à l'aide de la commande suivante.

```
mysql> show global status like 'aborted_clients';
+-----------------+-------+
| Variable_name   | Value |
+-----------------+-------+
| Aborted_clients | 9     |
+-----------------+-------+
1 row in set (0.01 sec)
```

Si le nombre `Aborted_clients` augmente au fil du temps, cela signifie que l'application ne ferme pas correctement les connexions à la base de données. Si les connexions ne sont pas correctement

fermées, cela peut entraîner des fuites de ressources et des problèmes de performances potentiels. Le fait de laisser des connexions ouvertes inutilement peut entraîner une consommation excessive de ressources système (mémoire, descripteurs de fichiers), pouvant à terme bloquer l'application ou le serveur, voire provoquer un redémarrage.

Vous pouvez utiliser la requête suivante pour identifier les comptes qui ne ferment pas correctement les connexions. Elle récupère le nom du compte utilisateur, l'hôte à partir duquel l'utilisateur se connecte, le nombre de connexions non fermées et le pourcentage de connexions non fermées.

```
SELECT
    ess.user,
    ess.host,
    (a.total_connections - a.current_connections) - ess.count_star AS not_closed,
    (((a.total_connections - a.current_connections) - ess.count_star) * 100) /
 (a.total_connections - a.current_connections) AS pct_not_closed
FROM
    performance_schema.events_statements_summary_by_account_by_event_name AS ess
    JOIN performance_schema.accounts AS a ON (ess.user = a.user AND ess.host = a.host)
WHERE
    ess.event_name = 'statement/com/quit'
    AND (a.total_connections - a.current_connections) > ess.count_star;

+----------+---------------+------------+----------------+
| user     | host          | not_closed | pct_not_closed |
+----------+---------------+------------+----------------+
| user1    | 172.31.49.222 |          1 |        33.3333 |
| user1    | 172.31.93.250 |       1024 |        12.1021 |
| user2    | 172.31.93.250 |         10 |        12.8551 |
+----------+---------------+------------+----------------+
3 rows in set (0.00 sec)
```

Après avoir identifié les comptes utilisateurs et les hôtes à partir desquels les connexions ne sont pas fermées, vous pouvez vérifier le code qui ne ferme pas les connexions correctement.

Par exemple, avec le connecteur MySQL en Python, utilisez la méthode `close()` de l'objet de connexion pour fermer les connexions. Voici un exemple de fonction qui établit une connexion à une base de données, exécute une requête et ferme la connexion :

```
import mysql.connector

def execute_query(query):
    # Establish a connection to the database
```

```
    connection = mysql.connector.connect(
        host="your_host",
        user="your_username",
        password="your_password",
        database="your_database"
    )

    try:
        # Create a cursor object
        cursor = connection.cursor()

        # Execute the query
        cursor.execute(query)

        # Fetch and process the results
        results = cursor.fetchall()
        for row in results:
            print(row)

    finally:
        # Close the cursor and connection
        cursor.close()
        connection.close()
```

Dans cet exemple, la méthode `connection.close()` est appelée dans le bloc `finally` pour s'assurer que la connexion est fermée, qu'une exception se produise ou non.

Exemple 3 : résolution des problèmes liés à l'échec des tentatives de connexion IAM

La connectivité avec les utilisateurs d' AWS Identity and Access Management (IAM) peut échouer pour plusieurs raisons, notamment :

- Configuration de la politique IAM incorrecte
- Identifiants de sécurité expirés
- Problèmes liés à la connectivité réseau
- Incompatibilité des autorisations de base de données

Pour résoudre ces erreurs d'authentification, activez la fonctionnalité d'exportation des `iam-db-auth-error` journaux dans votre base de données Amazon Relational Database Service (RDS) ou Aurora. Cela vous permettra d'afficher les messages d'erreur d'authentification détaillés dans le groupe de CloudWatch journaux de votre cluster Amazon RDS ou Amazon Aurora.

Une fois activé, vous pouvez consulter ces journaux pour identifier et résoudre la cause spécifique de vos échecs d'authentification IAM.

Par exemple :

```
2025-09-22T12:02:30,806 [ERROR] Failed to authorize the connection request for
 user 'user_1' due to an internal IAM DB Auth error. (Status Code: 500, Error Code:
 InternalError)
```

and

```
2025-09-22T12:02:51,954 [ERROR] Failed to authenticate the connection request for user
 'user_2' because the provided token is malformed or otherwise invalid. (Status Code:
 400, Error Code: InvalidToken)
```

Pour obtenir des conseils de dépannage, reportez-vous au guide de dépannage Aurora pour l'authentification IAM DB.

Résolution des problèmes de performance des requêtes pour les bases de données Aurora MySQL

MySQL permet de contrôler l'optimiseur de requêtes par le biais de variables système qui influent sur le mode d'évaluation des plans de requêtes, les optimisations remplaçables, les indices d'optimiseur et d'index, ainsi que le modèle de coût de l'optimiseur. Ces points de données peuvent être utiles non seulement pour comparer différents environnements MySQL, mais également pour comparer les plans d'exécution de requêtes précédents avec les plans d'exécution actuels, et pour comprendre l'exécution globale d'une requête MySQL à tout moment.

Les performances des requêtes dépendent de nombreux facteurs, notamment le plan d'exécution, le schéma et la taille de la table, les statistiques, les ressources, les index et la configuration des paramètres. L'ajustement des requêtes nécessite d'identifier les goulots d'étranglement et d'optimiser le chemin d'exécution.

- Identifiez le plan d'exécution de la requête et vérifiez si cette dernière utilise les index appropriés. Vous pouvez optimiser votre requête en utilisant `EXPLAIN` et en examinant les détails de chaque plan.
- Aurora MySQL version 3 (compatible avec MySQL 8.0 Community Edition) utilise une instruction `EXPLAIN ANALYZE`. L'instruction `EXPLAIN ANALYZE` est un outil de profilage qui indique où MySQL consacre du temps à votre requête, et pourquoi. Avec `EXPLAIN ANALYZE`, Aurora MySQL planifie, prépare et exécute la requête tout en comptant les lignes et en mesurant le temps passé à différents moments du plan d'exécution. Lorsque la requête est terminée, `EXPLAIN ANALYZE` imprime le plan et ses mesures au lieu du résultat de la requête.
- Mettez à jour les statistiques de votre schéma à l'aide de l'instruction `ANALYZE`. L'optimiseur de requêtes peut parfois choisir des plans d'exécution inappropriés en raison de statistiques obsolètes. Cela peut nuire aux performances d'une requête en raison d'estimations de cardinalité inexactes à la fois pour les tables et les index. La colonne `last_update` de la table innodb_table_stats indique la dernière fois que les statistiques de votre schéma ont été mises à jour, ce qui est un bon indicateur d'« obsolescence ».
- D'autres problèmes peuvent survenir, tels que le biais de distribution des données, lesquels ne seront pas pris en compte pour la cardinalité des tables. Pour plus d'informations, consultez Estimation de la complexité d'ANALYZE TABLE pour les tables InnoDB et Statistiques d'histogramme dans MySQL dans la documentation MySQL.

Comprendre le temps passé par les requêtes

Les éléments suivants permettent de déterminer le temps passé par les requêtes :

- Profilage
- Schéma de performance
- Optimiseur de requête

Profilage

Par défaut, le profilage est désactivé. Activez le profilage, puis exécutez la requête lente et vérifiez son profil.

```
SET profiling = 1;
Run your query.
SHOW PROFILE;
```

1. Identifiez l'étape où la requête passe le plus de temps. Selon États généraux des threads dans la documentation MySQL, la lecture et le traitement des lignes d'une instruction `SELECT` est souvent l'état le plus long au cours de la durée de vie d'une requête donnée. Vous pouvez utiliser cette instruction `EXPLAIN` pour comprendre comment MySQL exécute cette requête.
2. Consultez le journal des requêtes lentes pour évaluer `rows_examined` et `rows_sent` afin de vous assurer que la charge de travail est similaire dans chaque environnement. Pour plus d'informations, consultez Journalisation pour les bases de données Aurora MySQL.
3. Exécutez la commande suivante pour les tables faisant partie de la requête identifiée :

   ```
   SHOW TABLE STATUS\G;
   ```

4. Capturez les sorties suivantes avant et après l'exécution de la requête sur chaque environnement :

   ```
   SHOW GLOBAL STATUS;
   ```

5. Exécutez les commandes suivantes sur chaque environnement pour voir si une autre requête/session influence les performances de cet exemple de requête.

   ```
   SHOW FULL PROCESSLIST;
   ```

```
SHOW ENGINE INNODB STATUS\G;
```

Parfois, lorsque les ressources du serveur sont occupées, cela a un impact sur toutes les autres opérations du serveur, y compris les requêtes. Vous pouvez également capturer des informations périodiquement lorsque des requêtes sont exécutées, ou configurer une tâche `cron` pour capturer des informations à des intervalles utiles.

Schéma de performance

Le schéma de performance fournit des informations utiles sur les performances d'exécution du serveur, tout en ayant un impact minimal sur ces performances. Il diffère de l'`information_schema`, qui fournit des informations de schéma sur l'instance de base de données. Pour plus d'informations, consultez Présentation du schéma de performance pour Performance Insights sur Aurora MySQL.

Traçabilité de l'optimiseur de requête

Pour comprendre pourquoi un plan de requête particulier a été choisi pour être exécuté, vous pouvez configurer `optimizer_trace` pour accéder à l'optimiseur de requêtes MySQL.

Exécutez une opération optimizer_trace pour afficher des informations détaillées sur tous les chemins disponibles pour l'optimiseur et sur son choix.

```
SET SESSION OPTIMIZER_TRACE="enabled=on";
SET optimizer_trace_offset=-5, optimizer_trace_limit=5;

-- Run your query.
SELECT * FROM table WHERE x = 1 AND y = 'A';

-- After the query completes:
SELECT * FROM information_schema.OPTIMIZER_TRACE;
SET SESSION OPTIMIZER_TRACE="enabled=off";
```

Examen des paramètres de l'optimiseur de requêtes

Aurora MySQL version 3 (compatible avec MySQL 8.0 Community Edition) inclut de nombreuses modifications liées à l'optimiseur par rapport à Aurora MySQL version 2 (compatible avec MySQL 5.7 Community Edition). Si vous avez des valeurs personnalisées pour `optimizer_switch`, nous vous recommandons de vérifier les différences entre les valeurs par défaut et de définir les valeurs

`optimizer_switch` les mieux adaptées à votre charge de travail. Nous vous recommandons également de tester les options disponibles pour Aurora MySQL version 3 afin d'examiner les performances de vos requêtes.

> **Note**
>
> Aurora MySQL version 3 utilise la valeur par défaut 20 pour le paramètre innodb_stats_persistent_sample_pages.

Vous pouvez utiliser la commande suivante pour afficher les valeurs `optimizer_switch` :

```
SELECT @@optimizer_switch\G;
```

Le tableau suivant affiche les valeurs `optimizer_switch` par défaut pour Aurora MySQL versions 2 et 3.

Paramètre	Aurora MySQL version 2	Aurora MySQL version 3
`batched_key_access`	off	off
`block_nested_loop`	on	on
`condition_fanout_filter`	on	on
`derived_condition_pushdown`	–	on
`derived_merge`	on	on
`duplicateweedout`	on	on
`engine_condition_pushdown`	on	on
`firstmatch`	on	on
`hash_join`	off	on
`hash_join_cost_based`	on	–
`hypergraph_optimizer`	–	off

Paramètre	Aurora MySQL version 2	Aurora MySQL version 3
index_condition_pushdown	on	on
index_merge	on	on
index_merge_intersection	on	on
index_merge_sort_union	on	on
index_merge_union	on	on
loosescan	on	on
materialization	on	on
mrr	on	on
mrr_cost_based	on	on
prefer_ordering_index	on	on
semijoin	on	on
skip_scan	–	on
subquery_materialization_cost_based	on	on
subquery_to_derived	–	off
use_index_extensions	on	on
use_invisible_indexes	–	off

Pour plus d'informations, consultez Optimisations remplaçables (MySQL 5.7) et Optimisations remplaçables (MySQL 8.0) dans la documentation MySQL.

Référence Amazon Aurora MySQL

Cette référence inclut des informations sur les paramètres Aurora MySQL, les variables d'état et les extensions SQL générales ou les différences avec le moteur de base de données MySQL de la communauté.

Rubriques

Paramètres de configuration d'Aurora MySQL

Vous gérez votre cluster de bases de données Amazon Aurora MySQL de la même façon que les autres instances de base de données Amazon RDS, en utilisant les paramètres d'un groupe de paramètres de base de données. Amazon Aurora diffère des autres moteurs de base de données en ce sens que vous avez un cluster de bases de données qui contient plusieurs instances de base de données. En conséquence, certains des paramètres que vous utilisez pour gérer votre cluster de bases de données Aurora MySQL s'appliquent à l'ensemble du cluster. D'autres paramètres s'appliquent uniquement à une instance de base de données spécifique du cluster de bases de données.

Pour gérer les paramètres de niveau cluster, utilisez des groupes de paramètres de cluster de bases de données. Pour gérer les paramètres de niveau instance, utilisez des groupes de paramètres de base de données. Chaque instance de base de données d'un cluster de bases de données Aurora MySQL est compatible avec le moteur de base de données MySQL. Toutefois, vous devez appliquer certains paramètres du moteur de base de données MySQL au niveau du cluster et gérer ces paramètres à l'aide de groupes de paramètres de cluster de bases de données. Pour une instance située dans un cluster de bases de données Aurora, vous ne pouvez pas trouver les paramètres de

niveau cluster dans le groupe de paramètres de base de données. Plus loin dans cette rubrique, vous trouverez une liste des paramètres de niveau cluster.

Vous pouvez gérer les paramètres de niveau cluster et de niveau instance à l'aide de l'AWS Management Console, de l'AWS CLI ou de l'API Amazon RDS. Vous pouvez utiliser des commandes distinctes pour gérer les paramètres de niveau cluster et les paramètres de niveau instance. Par exemple, vous pouvez utiliser la commande CLI modify-db-cluster-parameter-group pour gérer les paramètres de niveau cluster dans un groupe de paramètres de cluster de bases de données. Vous pouvez utiliser la commande modify-db-parameter-group de l'interface de ligne de commande (CLI) pour gérer les paramètres de niveau instance dans un groupe de paramètres de cluster de bases de données dans un cluster de bases de données.

Vous pouvez afficher les paramètres de niveau cluster et de niveau instance dans la console, à l'aide de l'interface de ligne de commande (CLI) ou de l'API RDS. Par exemple, vous pouvez utiliser la commande de l'AWS CLI describe-db-cluster-parameters pour afficher les paramètres de niveau cluster dans un groupe de paramètres de cluster de bases de données. Vous pouvez utiliser la commande describe-db-parameters de l'interface de ligne de commande (CLI) pour afficher les paramètres de niveau instance dans un groupe de paramètres de cluster de bases de données dans un cluster de bases de données.

> **Note**
>
> Chaque groupe de paramètres par défaut contient les valeurs par défaut de tous les paramètres du groupe de paramètres. Si le paramètre est « engine default » (moteur par défaut) pour cette valeur, consultez la documentation MySQL ou PostgreSQL spécifique à la version pour connaître la valeur par défaut réelle.
> Sauf indication contraire, les paramètres répertoriés dans les tables suivantes sont valides pour Aurora MySQL versions 2 et 3.

Pour plus d'informations sur les groupes de paramètres DB, consultez Groupes de paramètres pour Amazon Aurora. Pour obtenir les règles et les instructions pour les clusters Aurora Serverless v1, consultez Groupes de paramètres pour Aurora Serverless v1.

Rubriques

- Paramètres de niveau cluster
- Paramètres de niveau instance
- Paramètres MySQL ne s'appliquant pas à Aurora MySQL

Paramètres de niveau cluster

Le tableau suivant affiche tous les paramètres qui s'appliquent à la totalité du cluster de bases de données Aurora MySQL.

Nom du paramètre	Adaptabilité	Remarques
`aurora_binlog_read_buffer_size`	Oui	Affecte uniquement les clusters qui utilisent la réplication de journal binaire (binlog). Pour plus d'informations sur la réplication de journal binaire, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires). Supprimé d'Aurora MySQL version 3.
`aurora_binlog_replication_max_yield_seconds`	Oui	Affecte uniquement les clusters qui utilisent la réplication de journal binaire (binlog). Pour plus d'informations sur la réplication de journal binaire, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).
`aurora_binlog_replication_sec_index_parallel_workers`	Oui	Définit le nombre total de threads parallèles disponibles pour appliquer les modifications d'index secondaires lors de la réplication de transactions pour de grandes tables comportant plusieurs index secondaires. Par défaut, ce paramètre est défini sur `0` (désactivé). Ce paramètre est disponible dans Aurora MySQL version 306 et versions

Nom du paramètre	Adaptabilité	Remarques
		ultérieures. Pour plus d'informations, consultez Optimisation de la réplication des journaux binaires pour Aurora MySQL.
`aurora_binlog_use_large_read_buffer`	Oui	Affecte uniquement les clusters qui utilisent la réplication de journal binaire (binlog). Pour plus d'informations sur la réplication de journal binaire, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires). Supprimé d'Aurora MySQL version 3.
`aurora_disable_hash_join`	Oui	Définissez ce paramètre sur `ON` pour désactiver l'optimisation de jointure par hachage dans Aurora MySQL version 2.09 ou ultérieure. Il n'est pas pris en charge pour la version 3. Pour plus d'informations, consultez Requêtes parallèles pour Amazon Aurora MySQL.
`aurora_enable_replica_log_compression`	Oui	Pour plus d'informations, consultez Considérations sur les performances pour la réplication Amazon Aurora MySQL. Ne s'applique pas aux clusters qui font partie d'une base de données globale Aurora. Supprimé d'Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`aurora_enable_repl_bin_log_filtering`	Oui	Pour plus d'informations, consultez Considérations sur les performances pour la réplication Amazon Aurora MySQL. Ne s'applique pas aux clusters qui font partie d'une base de données globale Aurora. Supprimé d'Aurora MySQL version 3.
`aurora_enable_staggered_replica_restart`	Oui	Ce paramètre est disponible dans Aurora MySQL version 3, mais il n'est pas utilisé.
`aurora_enable_zdr`	Oui	Ce paramètre est activé par défaut dans Aurora MySQL versions 2.10 et ultérieures.
`aurora_in_memory_relaylog`	Oui	Définit le mode journal de relais en mémoire. Vous pouvez utiliser cette fonctionnalité sur les réplicas de journaux binaires pour améliorer les performances de réplication des journaux binaires. Pour désactiver cette fonctionnalité, définissez ce paramètre sur OFF. Pour activer cette fonctionnalité, définissez ce paramètre sur ON.
`aurora_enhanced_binlog`	Oui	Définissez la valeur de ce paramètre sur 1 pour activer le binlog amélioré dans Aurora MySQL versions 3.03.1 et ultérieures. Pour plus d'informations, consultez Configuration du binlog amélioré pour Aurora MySQL.

Nom du paramètre	Adaptabilité	Remarques
`aurora_full_double_precision_in_json`	Oui	Définissez la valeur de ce paramètre pour permettre l'analyse des nombres à virgule flottante dans les documents JSON avec une précision totale.
`aurora_jemalloc_background_thread`	Oui	Utilisez ce paramètre pour permettre à un thread d'arrière-plan d'effectuer des opérations de maintenance de la mémoire. Les valeurs autorisées sont `0` (désactivé) et 1 (activé). La valeur par défaut est `0`. Ce paramètre s'applique à Aurora MySQL versions 3.05 et ultérieures.
`aurora_jemalloc_dirty_decay_ms`	Oui	Utilisez ce paramètre pour conserver la mémoire libérée pendant un certain temps (en millisecondes). La conservation de la mémoire permet une réutilisation plus rapide. Les valeurs autorisées sont `0`–`18446744073709551615` . La valeur par défaut est `10000` (10 secondes). Vous pouvez utiliser un délai plus court pour éviter les problèmes de mémoire insuffisante, au détriment de performances plus lentes. Ce paramètre s'applique à Aurora MySQL versions 3.05 et ultérieures.

Nom du paramètre	Adaptabilité	Remarques
`aurora_jemalloc_tcache_enabled`	Oui	Utilisez ce paramètre pour traiter les faibles demandes de mémoire (jusqu'à 32 Kio) dans un cache local de thread, en contournant les arènes de mémoire. Les valeurs autorisées sont `0` (désactivé) et 1 (activé). La valeur par défaut est 1. Ce paramètre s'applique à Aurora MySQL versions 3.05 et ultérieures.
`aurora_load_from_s3_role`	Oui	Pour plus d'informations, consultez Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3. Actuellement non disponible dans Aurora MySQL version 3. Utilisez `aws_default_s3_role` .

Nom du paramètre	Adaptabilité	Remarques
`aurora_mask_password_hashes_type`	Oui	Ce paramètre est activé par défaut dans Aurora MySQL versions 2.11 et ultérieures. Utilisez ce paramètre pour masquer les hachages de mot de passe Aurora MySQL dans les journaux de requêtes lentes et d'audit. Les valeurs autorisées sont `0` et `1` (par défaut). Lorsque ce paramètre est défini sur `1`, les mots de passe sont enregistrés comme `<secret>`. Lorsque ce paramètre est défini sur `0`, les mots de passe sont enregistrés sous forme de valeurs de hachage (#).
`aurora_select_into_s3_role`	Oui	Pour plus d'informations, consultez Enregistrement de données d'un cluster de bases de données Amazon Aurora MySQL dans des fichiers texte stockés dans un compartiment Amazon S3. Actuellement non disponible dans Aurora MySQL version 3. Utilisez `aws_default_s3_role` .

Nom du paramètre	Adaptabilité	Remarques
`authentication_kerberos_caseins_cmp`	Oui	Contrôle la comparaison des noms d'utilisateur sans distinction de casse pour le plug-in `authentication_kerberos` . Définissez-le sur `true` pour une comparaison sans distinction de casse. Par défaut, la comparaison sensible à la casse est utilisée (`false`). Pour plus d'informations, consultez Utilisation de l'authentification Kerberos pour Aurora MySQL. Ce paramètre est disponible dans Aurora MySQL version 3.03 et versions ultérieures.
`auto_increment_increment`	Oui	Aucun
`auto_increment_offset`	Oui	Aucun
`aws_default_lambda_role`	Oui	Pour plus d'informations, consultez Appel d'une fonction Lambda à partir d'un cluster de bases de données Amazon Aurora MySQL.

Nom du paramètre	Adaptabilité	Remarques
`aws_default_s3_role`	Oui	Utilisé lors de l'appel de l'instruction `LOAD DATA FROM S3`, `LOAD XML FROM S3` ou `SELECT INTO OUTFILE S3` à partir de votre cluster de bases de données. Dans Aurora MySQL version 2, le rôle IAM spécifié dans ce paramètre est utilisé si un rôle IAM n'est pas spécifié pour `aurora_load_from_s3_role` ou `aurora_select_into_s3_role` pour l'instruction appropriée. Dans Aurora MySQL version 3, le rôle IAM spécifié pour ce paramètre est toujours utilisé. Pour plus d'informations, consultez Association d'un rôle IAM à un cluster de bases de données Amazon Aurora MySQL.
`binlog_backup`	Oui	Définissez la valeur de ce paramètre sur 0 pour activer le binlog amélioré dans Aurora MySQL versions 3.03.1 et ultérieures. Vous ne pouvez désactiver ce paramètre que lorsque vous utilisez le binlog amélioré. Pour plus d'informations, consultez Configuration du binlog amélioré pour Aurora MySQL.

Nom du paramètre	Adaptabilité	Remarques
`binlog_checksum`	Oui	La AWS CLI et l'API RDS indiquent la valeur `None` si ce paramètre n'est pas défini. Dans ce cas, Aurora MySQL utilise la valeur par défaut du moteur, qui est `CRC32`. Ceci est différent du paramétrage explicite de `NONE`, qui désactive le total de contrôle.
`binlog-do-db`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`binlog_format`	Oui	Pour plus d'informations, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).
`binlog_group_commit_sync_delay`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`binlog_group_commit_sync_no_delay_count`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`binlog-ignore-db`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`binlog_replication_globaldb`	Oui	Définissez la valeur de ce paramètre sur 0 pour activer le binlog amélioré dans Aurora MySQL versions 3.03.1 et ultérieures. Vous ne pouvez désactiver ce paramètre que lorsque vous utilisez le binlog amélioré. Pour plus d'informations, consultez Configuration du binlog amélioré pour Aurora MySQL.

Nom du paramètre	Adaptabilité	Remarques
binlog_row_image	Non	Aucun
binlog_row_metadata	Oui	Ce paramètre s’applique à Aurora MySQL version 3.
binlog_row_value_options	Oui	Ce paramètre s’applique à Aurora MySQL version 3.
binlog_rows_query_log_events	Oui	Aucun
binlog_transaction_compression	Oui	Ce paramètre s’applique à Aurora MySQL version 3.
binlog_transaction_compression_level_zstd	Oui	Ce paramètre s’applique à Aurora MySQL version 3.
binlog_transaction_dependency_history_size	Oui	Ce paramètre définit une limite supérieure du nombre de hachages de ligne conservés en mémoire et utilisés pour rechercher la transaction qui a modifié pour la dernière fois une ligne donnée. Une fois ce nombre de hachages atteint, l’historique est purgé. Ce paramètre s’applique à Aurora MySQL versions 2.12 et ultérieures, et version 3.
binlog_transaction_dependency_tracking	Oui	Ce paramètre s’applique à Aurora MySQL version 3.
character-set-client-handshake	Oui	Aucun
character_set_client	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
character_set_connection	Oui	Aucun
character_set_database	Oui	Jeu de caractères utilisé par la base de données par défaut. La valeur par défaut est utf8mb4.
character_set_filesystem	Oui	Aucun
character_set_results	Oui	Aucun
character_set_server	Oui	Aucun
collation_connection	Oui	Aucun
collation_server	Oui	Aucun
completion_type	Oui	Aucun
default_storage_engine	Non	Les clusters Aurora MySQL utilisent le moteur de stockage InnoDB pour toutes vos données.
enforce_gtid_consistency	Parfois	Modifiable dans Aurora MySQL version 2 et versions ultérieures.
event_scheduler	Oui	Indique l'état du planificateur d'événements. Modifiable uniquement au niveau du cluster dans Aurora MySQL version 3.
gtid-mode	Parfois	Modifiable dans Aurora MySQL version 2 et versions ultérieures.

Nom du paramètre	Adaptabilité	Remarques
`information_schema_stats_expiry`	Oui	Nombre de secondes après lesquelles le serveur de base de données MySQL récupère les données du moteur de stockage et remplace les données du cache. Les valeurs autorisées sont `0`–`31536000`. Ce paramètre s'applique à Aurora MySQL version 3.
`init_connect`	Oui	La commande à exécuter par le serveur pour chaque client qui se connecte. Utilisez des guillemets (") pour les paramètres afin d'éviter les échecs de connexion, par exemple : `SET optimizer_switch="hash_join =off"` Dans Aurora MySQL version 3, ce paramètre ne s'applique pas aux utilisateurs disposant du privilège `CONNECTION_ADMIN` . Cela inclut l'utilisateur principal d'Aurora. Pour plus d'informations, consultez Modèle de privilège basé sur les rôles.
`innodb_adaptive_hash_index`	Oui	Vous pouvez modifier ce paramètre au niveau du cluster de bases de données dans Aurora MySQL versions 2 et 3. L'index de hachage adaptatif n'est pas pris en charge par les instances de base de données du lecteur.

Nom du paramètre	Adaptabilité	Remarques
innodb_aurora_instant_alter_column_allowed	Oui	Contrôle si l'algorithme INSTANT peut être utilisé pour des opérations ALTER COLUMN au niveau global. Les valeurs autorisées sont les suivantes : • 0 : l'algorithme INSTANT n'est pas autorisé pour les opérations ALTER COLUMN (OFF). Revient à d'autres algorithmes. • 1 : l'algorithme INSTANT est autorisé pour les opérations ALTER COLUMN (ON). C'est la valeur par défaut. Pour plus d'informations, consultez Column Operations dans la documentation MySQL. Ce paramètre s'applique à Aurora MySQL versions 3.05 et ultérieures.
innodb_autoinc_lock_mode	Oui	Aucun
innodb_checksums	Non	Supprimé d'Aurora MySQL version 3.
innodb_cmp_per_index_enabled	Oui	Aucun
innodb_commit_concurrency	Oui	Aucun
innodb_data_home_dir	Non	Aurora MySQL utilise des instances gérées dans lesquelles vous n'accédez pas directement au système de fichiers.

Nom du paramètre	Adaptabilité	Remarques
`innodb_deadlock_detect`	Oui	Cette option permet de désactiver la détection de blocage dans Aurora MySQL version 2.11 ou ultérieure, ou version 3. Sur les systèmes à concurrence élevée, la détection de blocage peut entraîner un ralentissement lorsque de nombreux threads attendent le même verrouillage. Pour plus d'informations sur ce paramètre, consultez la documentation MySQL.
`innodb_default_row_format`	Oui	Ce paramètre définit le format de ligne par défaut pour les tables InnoDB (y compris les tables temporaires InnoDB créées par l'utilisateur). Il s'applique aux versions 2 et 3 d'Aurora MySQL. Ses valeurs peuvent être `DYNAMIC`, `COMPACT` ou `REDUNDANT`.
`innodb_file_per_table`	Oui	Ce paramètre affecte la façon dont le stockage de table est organisé. Pour plus d'informations, consultez Dimensionnement du stockage.

Nom du paramètre	Adaptabilité	Remarques
innodb_flush_log_at_trx_commit	Oui	Nous vous recommandons de conserver la valeur par défaut 1. Dans Aurora MySQL version 3, avant de pouvoir attribuer à ce paramètre une valeur autre que 1, vous devez définir la valeur de innodb_trx_commit_allow_data_loss sur 1. Pour plus d'informations, consultez Configuration de la fréquence à laquelle le tampon du journal est vidé.
innodb_ft_max_token_size	Oui	Aucun
innodb_ft_min_token_size	Oui	Aucun
innodb_ft_num_word_optimize	Oui	Aucun
innodb_ft_sort_pll_degree	Oui	Aucun
innodb_online_alter_log_max_size	Oui	Aucun
innodb_optimize_fulltext_only	Oui	Aucun
innodb_page_size	Non	Aucun

Nom du paramètre	Adaptabilité	Remarques
innodb_print_all_deadlocks	Oui	Lorsque ce paramètre est activé, les informations relatives à tous les blocages InnoDB sont enregistrées dans le journal des erreurs Aurora MySQL. Pour plus d'informations, consultez Minimisation et résolution des blocages d'Aurora MySQL.
innodb_purge_batch_size	Oui	Aucun
innodb_purge_threads	Oui	Aucun
innodb_rollback_on_timeout	Oui	Aucun
innodb_rollback_segments	Oui	Aucun
innodb_spin_wait_delay	Oui	Aucun
innodb_strict_mode	Oui	Aucun
innodb_support_xa	Oui	Supprimé d'Aurora MySQL version 3.
innodb_sync_array_size	Oui	Aucun
innodb_sync_spin_loops	Oui	Aucun
innodb_stats_include_delete_marked	Oui	Lorsque ce paramètre est activé, InnoDB inclut les enregistrements marqués pour suppression lors du calcul des statistiques persistantes de l'optimiseur. Ce paramètre s'applique à Aurora MySQL versions 2.12 et ultérieures, et version 3.
innodb_table_locks	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
innodb_trx_commit_allow_data_loss	Oui	Dans Aurora MySQL version 3, définissez la valeur de ce paramètre sur 1 afin de pouvoir modifier la valeur de innodb_flush_log_at_trx_commit . La valeur par défaut de innodb_trx_commit_allow_data_loss est 0. Pour plus d'informations, consultez Configuration de la fréquence à laquelle le tampon du journal est vidé.
innodb_undo_directory	Non	Aurora MySQL utilise des instances gérées dans lesquelles vous n'accédez pas directement au système de fichiers.
internal_tmp_disk_storage_engine	Oui	Contrôle quel moteur de stockage en mémoire est utilisé pour les tables temporaires internes. Les valeurs autorisées sont INNODB et MYISAM. Ce paramètre s'applique à Aurora MySQL version 2.
internal_tmp_mem_storage_engine	Oui	Contrôle quel moteur de stockage en mémoire est utilisé pour les tables temporaires internes. Les valeurs autorisées sont MEMORY et TempTable . Ce paramètre s'applique à Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`key_buffer_size`	Oui	Cache de clé pour les tables MyISAM. Pour plus d'informations, consultez keycache->cache_lock mutex.
`lc_time_names`	Oui	Aucun
`log_error_suppression_list`	Oui	Spécifie une liste de codes d'erreur qui ne sont pas consignés dans le journal d'erreurs MySQL. Cela vous permet d'ignorer certaines conditions d'erreur non critiques afin de préserver l'intégrité de vos journaux d'erreurs. Pour plus d'informations, consultez log_error_suppression_list dans la documentation MySQL. Ce paramètre s'applique à Aurora MySQL versions 3.03 et ultérieures.
`low_priority_updates`	Oui	Les opérations `INSERT`, `UPDATE`, `DELETE` et `LOCK TABLE WRITE` attendent qu'il ne reste aucune opération `SELECT` en attente. Ce paramètre affecte uniquement les moteurs de stockage qui utilisent uniquement le verrouillage au niveau de la table (MyISAM, MEMORY, MERGE). Ce paramètre s'applique à Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`lower_case_table_names`	Oui (Aurora MySQL version 2) Uniquement au moment de la création du cluster (Aurora MySQL version 3)	Dans Aurora MySQL version 2.10 et versions 2.x ultérieures, veillez à redémarrer toutes les instances de lecteur après avoir modifié ce paramètre et redémarré l'instance d'enregistreur. Pour en savoir plus, consultez Redémarrage d'un cluster Aurora avec disponibilité en lecture. Dans Aurora MySQL version 3, la valeur de ce paramètre est définie de façon permanente au moment de la création du cluster. Si vous utilisez une valeur autre que par défaut pour cette option, configurez votre groupe de paramètres personnalisés Aurora MySQL version 3 avant la mise à niveau, puis spécifiez le groupe de paramètres pendant l'opération de restauration des instantanés qui crée le cluster version 3. Avec une base de données globale Aurora basée sur Aurora MySQL, vous ne pouvez pas effectuer une mise à niveau sur place d'Aurora MySQL version 2 vers la version 3 si le paramètre `lower_case_table_names` est activé. Pour plus d'informations sur les méthodes que vous pouvez utiliser, consultez Mises à niveau de version majeure..
`master-info-repository`	Oui	Supprimé d'Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`master_verify_checksum`	Oui	Aurora MySQL version 2. Utilisez `source_verify_checksum` dans Aurora MySQL version 3.
`max_delayed_threads`	Oui	Définit le nombre maximal de threads pour gérer les instructions `INSERT DELAYED`. Ce paramètre s'applique à Aurora MySQL version 3.
`max_error_count`	Oui	Nombre maximal de messages d'erreur, d'avertissement et de note à stocker pour affichage. Ce paramètre s'applique à Aurora MySQL version 3.
`max_execution_time`	Oui	Délai d'attente pour l'exécution des instructions `SELECT`, en millisecondes. Cette valeur peut être comprise entre `0` et `18446744073709551615`. Lorsqu'elle est définie sur `0`, il n'y a aucun délai d'attente. Pour plus d'informations, consultez max_execution_time dans la documentation MySQL.
`min_examined_row_limit`	Oui	Utilisez ce paramètre pour empêcher la journalisation des requêtes examinant un nombre de lignes inférieur au nombre spécifié.
`partial_revokes`	Non	Ce paramètre s'applique à Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`preload_buffer_size`	Oui	Taille de la mémoire tampon allouée lors du préchargement des index. Ce paramètre s’applique à Aurora MySQL version 3.
`query_cache_type`	Oui	Supprimé d’Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`read_only`	Oui	Lorsque ce paramètre est activé, le serveur n'autorise aucune mise à jour, à l'exception de celles effectuées par les threads de réplica. Pour Aurora MySQL version 2, les valeurs valides sont les suivantes : • `0` – `OFF` • `1` – `ON` • `{TrueIfReplica}` — `ON` pour les réplicas en lecture. C'est la valeur par défaut. • `{TrueIfClusterReplica}` — `ON` pour les clusters de réplica tels que les réplicas en lecture entre régions, les clusters secondaires dans une base de données globale Aurora et les déploiements bleu/vert. Pour Aurora MySQL version 3, les valeurs valides sont les suivantes : • `0` – `OFF`. C'est la valeur par défaut. • `1` – `ON` • `{TrueIfClusterReplica}` — `ON` pour les clusters de réplica tels que les réplicas en lecture entre régions, les clusters secondaires dans une base de données globale Aurora et les déploiements bleu/vert.

Nom du paramètre	Adaptabilité	Remarques
		Dans Aurora MySQL version 3, ce paramètre ne s'applique pas aux utilisateurs disposant du privilège `CONNECTION_ADMIN` . Cela inclut l'utilisateur principal d'Aurora. Pour plus d'informations, consultez Modèle de privilège basé sur les rôles.
`relay-log-space-limit`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`replica_parallel_type`	Oui	Ce paramètre permet une exécution parallèle sur le réplica de tous les threads non validés déjà en phase de préparation, sans porter atteinte à la cohérence. Il s'applique à Aurora MySQL version 3. Dans les versions 3.03.* et antérieures d'Aurora MySQL, la valeur par défaut est DATABASE. Dans les versions 3.04 et ultérieures d'Aurora MySQL, la valeur par défaut est LOGICAL_CLOCK.
`replica_preserve_commit_order`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`replica_transaction_retries`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`replica_type_conversions`	Oui	Ce paramètre détermine les conversions de type utilisées sur les réplicas. Les valeurs autorisées sont `ALL_LOSSY` , `ALL_NON_LOSSY` , `ALL_SIGNED` et `ALL_UNSIGNED` . Pour plus d'informations, consultez Réplication avec des définitions de tables différentes sur la source et le réplica (langue française non garantie) dans la documentation MySQL. Ce paramètre s'applique à Aurora MySQL version 3.
`replicate-do-db`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`replicate-do-table`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`replicate-ignore-db`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`replicate-ignore-table`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`replicate-wild-do-table`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`replicate-wild-ignore-table`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`require_secure_transport`	Oui	Ce paramètre s'applique à Aurora MySQL versions 2 et 3. Pour plus d'informations, consultez Connexions TLS aux clusters de bases de données Aurora MySQL.
`rpl_read_size`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`server_audit_cw_upload`	Non	Ce paramètre est devenu obsolète dans Aurora MySQL. Utilisez `server_audit_logs_upload` . Pour plus d'informations, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs.
`server_audit_events`	Oui	Pour plus d'informations, consultez Utilisation de l'Audit avancé avec un cluster de bases de données Amazon Aurora MySQL.
`server_audit_excl_users`	Oui	Pour plus d'informations, consultez Utilisation de l'Audit avancé avec un cluster de bases de données Amazon Aurora MySQL.
`server_audit_incl_users`	Oui	Pour plus d'informations, consultez Utilisation de l'Audit avancé avec un cluster de bases de données Amazon Aurora MySQL.

Nom du paramètre	Adaptabilité	Remarques
`server_audit_logging`	Oui	Pour accéder aux instructions concernant le chargement de journaux dans Amazon CloudWatch Logs, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs.
`server_audit_logs_upload`	Oui	Vous pouvez publier des journaux d'audit sur CloudWatch Logs en activant Audit avancé et en définissant ce paramètre sur 1. La valeur par défaut du paramètre `server_audit_logs_upload` est `0`. Pour plus d'informations, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs.
`server_id`	Non	Aucun
`skip-character-set-client-handshake`	Oui	Aucun
`skip_name_resolve`	Non	Aucun
`slave-skip-errors`	Oui	S'applique uniquement aux clusters de la version 2 d'Aurora MySQL, avec compatibilité MySQL 5.7.
`source_verify_checksum`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`sync_frm`	Oui	Supprimé d'Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`thread_cache_size`	Oui	Le nombre de threads à mettre en cache. Ce paramètre s'applique à Aurora MySQL versions 2 et 3.
`time_zone`	Oui	Par défaut, le fuseau horaire d'un cluster de bases de données Aurora est le fuseau UTC (temps universel). Vous pouvez à la place définir le fuseau horaire des instances de votre cluster de bases de données sur le fuseau horaire local de votre application. Pour plus d'informations, consultez Fuseau horaire local pour les clusters de bases de données Amazon Aurora.
`tls_version`	Oui	Pour plus d'informations, consultez Versions TLS pour Aurora MySQL.

Paramètres de niveau instance

Le tableau suivant affiche tous les paramètres qui s'appliquent à une instance de base de données spécifique d'un cluster de bases de données Aurora MySQL.

Nom du paramètre	Adaptabilité	Remarques
`activate_all_roles_on_login`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.
`allow-suspicious-udfs`	Non	Aucun
`aurora_disable_hash_join`	Oui	Définissez ce paramètre sur `0N` pour désactiver l'optimisation de jointure par hachage dans Aurora MySQL

Nom du paramètre	Adaptabilité	Remarques
		version 2.09 ou ultérieure. Il n'est pas pris en charge pour la version 3. Pour plus d'informations, consultez Requêtes parallèles pour Amazon Aurora MySQL.
`aurora_lab_mode`	Oui	Pour plus d'informations, consultez Mode Lab Amazon Aurora MySQL. Supprimé d'Aurora MySQL version 3.
`aurora_oom_response`	Oui	Ce paramètre est pris en charge pour Aurora MySQL versions 2 et 3. Pour plus d'informations, consultez Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL.
`aurora_parallel_query`	Oui	Définissez sur `ON` pour activer la requête parallèle dans Aurora MySQL version 2.09 ou ultérieure. L'ancien paramètre `aurora_pq` n'est pas utilisé dans ces versions. Pour plus d'informations, consultez Requêtes parallèles pour Amazon Aurora MySQL.
`aurora_pq`	Oui	Définissez sur `OFF` pour désactiver la requête parallèle pour des instances de base de données spécifiques dans les versions d'Aurora MySQL antérieures à 2.09. Dans la version 2.09 ou une version ultérieure, activez et désactivez la requête parallèle avec `aurora_parallel_query` à la place. Pour plus d'informations, consultez Requêtes parallèles pour Amazon Aurora MySQL.

Nom du paramètre	Adaptabilité	Remarques
`aurora_read_replica_read_committed`	Oui	Active le niveau d'isolement `READ COMMITTED` pour les réplicas Aurora et modifie le comportement d'isolement pour réduire le retard de purge pendant les requêtes de longue durée. N'activez ce paramètre que si vous maîtrisez les modifications de comportement et la façon dont ils affectent les résultats de vos requêtes. Par exemple, ce paramètre utilise un isolement moins strict que l'isolement MySQL par défaut. Lorsque le paramètre est activé, les requêtes de longue durée peuvent voir plusieurs exemplaires d'une même ligne, car Aurora réorganise les données des tables au cours de l'exécution de la requête. Pour plus d'informations, consultez Niveaux d'isolement Aurora MySQL.
`aurora_tmptable_enable_per_table_limit`	Oui	Détermine si le paramètre `tmp_table_size` contrôle la taille maximale des tables temporaires en mémoire créées par le moteur de stockage `TempTable` dans Aurora MySQL version 3.04 ou ultérieure. Pour plus d'informations, consultez Limitation de la taille des tables temporaires internes en mémoire.

Nom du paramètre	Adaptabilité	Remarques
aurora_use_vector_instructions	Oui	Lorsque ce paramètre est activé, Aurora MySQL utilise des instructions de traitement vectoriel optimisé fournies par les processeurs modernes pour améliorer les performances des charges de travail intensives en E/S. Ce paramètre est activé par défaut dans Aurora MySQL 3.05 et versions ultérieures.
autocommit	Oui	Aucun
automatic_sp_privileges	Oui	Aucun
back_log	Oui	Aucun
basedir	Non	Aurora MySQL utilise des instances gérées dans lesquelles vous n’accédez pas directement au système de fichiers.
binlog_cache_size	Oui	Aucun
binlog_max_flush_queue_time	Oui	Aucun
binlog_order_commits	Oui	Aucun
binlog_stmt_cache_size	Oui	Aucun
binlog_transaction_compression	Oui	Ce paramètre s’applique à Aurora MySQL version 3.
binlog_transaction_compression_level_zstd	Oui	Ce paramètre s’applique à Aurora MySQL version 3.
bulk_insert_buffer_size	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
concurrent_insert	Oui	Aucun
connect_timeout	Oui	Aucun
core-file	Non	Aurora MySQL utilise des instances gérées dans lesquelles vous n'accédez pas directement au système de fichiers.
datadir	Non	Aurora MySQL utilise des instances gérées dans lesquelles vous n'accédez pas directement au système de fichiers.
default_authentication_plugin	Non	Ce paramètre s'applique à Aurora MySQL version 3.
default_time_zone	Non	Aucun
default_tmp_storage_engine	Oui	Le moteur de stockage par défaut pour les tables temporaires.
default_week_format	Oui	Aucun
delay_key_write	Oui	Aucun
delayed_insert_limit	Oui	Aucun
delayed_insert_timeout	Oui	Aucun
delayed_queue_size	Oui	Aucun
div_precision_increment	Oui	Aucun
end_markers_in_json	Oui	Aucun
eq_range_index_dive_limit	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
`event_scheduler`	Parfois	Indique l'état du planificateur d'événements. Modifiable uniquement au niveau du cluster dans Aurora MySQL version 3.
`explicit_defaults_for_timestamp`	Oui	Aucun
`flush`	Non	Aucun
`flush_time`	Oui	Aucun
`ft_boolean_syntax`	Non	Aucun
`ft_max_word_len`	Oui	Aucun
`ft_min_word_len`	Oui	Aucun
`ft_query_expansion_limit`	Oui	Aucun
`ft_stopword_file`	Oui	Aucun
`general_log`	Oui	Pour accéder aux instructions concernant le chargement de journaux dans CloudWatch Logs, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs.
`general_log_file`	Non	Aurora MySQL utilise des instances gérées dans lesquelles vous n'accédez pas directement au système de fichiers.
`group_concat_max_len`	Oui	Aucun
`host_cache_size`	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
`init_connect`	Oui	La commande à exécuter par le serveur pour chaque client qui se connecte. Utilisez des guillemets (") pour les paramètres afin d'éviter les échecs de connexion, par exemple : `SET optimizer_switch="hash_join =off"` Dans Aurora MySQL version 3, ce paramètre ne s'applique pas aux utilisateurs disposant du privilège `CONNECTION_ADMIN` , y compris l'utilisateur principal Aurora. Pour plus d'informations, consultez Modèle de privilège basé sur les rôles.
`innodb_adaptive_hash_index`	Oui	Vous pouvez modifier ce paramètre au niveau de l'instance de base de données dans Aurora MySQL version 2. Il peut être modifié uniquement au niveau du cluster de bases de données dans Aurora MySQL version 3. L'index de hachage adaptatif n'est pas pris en charge par les instances de base de données du lecteur.
`innodb_adaptive_max_sleep_delay`	Oui	La modification de ce paramètre n'a aucun effet, car la valeur de `innodb_thread_concurrency` est toujours 0 pour Aurora.

Nom du paramètre	Adaptabilité	Remarques
innodb_aurora_max_partitions_for_range	Oui	Si les statistiques persistantes ne sont pas disponibles, vous pouvez utiliser ce paramètre pour améliorer les performances des estimations du nombre de lignes dans les tables partitionnées. Vous pouvez le définir sur une valeur comprise entre 0 et 8192, cette valeur déterminant le nombre de partitions à vérifier lors de l'estimation du nombre de lignes. La valeur par défaut est 0, qui estime l'utilisation de toutes les partitions, conformément au comportement par défaut de MySQL. Ce paramètre est disponible pour Aurora MySQL versions 3.03.1 et ultérieures.
innodb_autoextend_increment	Oui	Aucun
innodb_buffer_pool_dump_at_shutdown	Non	Aucun
innodb_buffer_pool_dump_now	Non	Aucun
innodb_buffer_pool_filename	Non	Aucun
innodb_buffer_pool_load_abort	Non	Aucun
innodb_buffer_pool_load_at_startup	Non	Aucun

Nom du paramètre	Adaptabilité	Remarques
innodb_buffer_pool_load_now	Non	Aucun
innodb_buffer_pool_size	Oui	La valeur par défaut est représentée par une formule. Pour plus de détails sur le calcul de la valeur DBInstanceClassMemory dans la formule, consultez Variables de formule de paramètre de bases de données.
innodb_change_buffer_max_size	Non	Aurora MySQL n'utilise pas du tout le tampon de modification InnoDB.
innodb_compression_failure_threshold_pct	Oui	Aucun
innodb_compression_level	Oui	Aucun
innodb_compression_pad_pct_max	Oui	Aucun
innodb_concurrency_tickets	Oui	La modification de ce paramètre n'a aucun effet, car la valeur de innodb_thread_concurrency est toujours 0 pour Aurora.

Nom du paramètre	Adaptabilité	Remarques
innodb_deadlock_detect	Oui	Cette option permet de désactiver la détection de blocage dans Aurora MySQL version 2.11 ou ultérieure, ou version 3. Sur les systèmes à concurrence élevée, la détection de blocage peut entraîner un ralentissement lorsque de nombreux threads attendent le même verrouillage. Pour plus d'informations sur ce paramètre, consultez la documentation MySQL.
innodb_file_format	Oui	Supprimé d'Aurora MySQL version 3.
innodb_flushing_avg_loops	Non	Aucun
innodb_force_load_corrupted	Non	Aucun
innodb_ft_aux_table	Oui	Aucun
innodb_ft_cache_size	Oui	Aucun
innodb_ft_enable_stopword	Oui	Aucun
innodb_ft_server_stopword_table	Oui	Aucun
innodb_ft_user_stopword_table	Oui	Aucun
innodb_large_prefix	Oui	Supprimé d'Aurora MySQL version 3.
innodb_lock_wait_timeout	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
innodb_log_compressed_pages	Non	Aucun
innodb_lru_scan_depth	Oui	Aucun
innodb_max_purge_lag	Oui	Aucun
innodb_max_purge_lag_delay	Oui	Aucun
innodb_monitor_disable	Oui	Aucun
innodb_monitor_enable	Oui	Aucun
innodb_monitor_reset	Oui	Aucun
innodb_monitor_reset_all	Oui	Aucun
innodb_old_blocks_pct	Oui	Aucun
innodb_old_blocks_time	Oui	Aucun
innodb_open_files	Oui	Aucun
innodb_print_all_deadlocks	Oui	Lorsque ce paramètre est activé, les informations relatives à tous les blocages InnoDB sont enregistrées dans le journal des erreurs Aurora MySQL. Pour plus d'informations, consultez Minimisation et résolution des blocages d'Aurora MySQL.
innodb_random_read_ahead	Oui	Aucun
innodb_read_ahead_threshold	Oui	Aucun
innodb_read_io_threads	Non	Aucun

Nom du paramètre	Adaptabilité	Remarques
innodb_read_only	Non	Aurora MySQL gère l'état en lecture seule et en lecture/écriture des instances de base de données en fonction du type de cluster. Par exemple, un cluster alloué a une instance de base de données en lecture/écriture (l'instance principale) et toutes les autres instances contenues dans le cluster sont en lecture/écriture (les réplicas Aurora).
innodb_replication_delay	Oui	Aucun
innodb_sort_buffer_size	Oui	Aucun
innodb_stats_auto_recalc	Oui	Aucun
innodb_stats_method	Oui	Aucun
innodb_stats_on_metadata	Oui	Aucun
innodb_stats_persistent	Oui	Aucun
innodb_stats_persistent_sample_pages	Oui	Aucun
innodb_stats_transient_sample_pages	Oui	Aucun
innodb_thread_concurrency	Non	Aucun
innodb_thread_sleep_delay	Oui	La modification de ce paramètre n'a aucun effet, car la valeur de innodb_thread_concurrency est toujours 0 pour Aurora.

Nom du paramètre	Adaptabilité	Remarques
`interactive_timeout`	Oui	Aurora évalue la valeur minimale de `interactive_timeout` et `wait_timeout` . Il utilise ensuite ce minimum comme délai pour mettre fin à toutes les sessions inactives, à la fois interactives et non interactives.
`internal_tmp_disk_storage_engine`	Oui	Contrôle quel moteur de stockage en mémoire est utilisé pour les tables temporaires internes. Les valeurs autorisées sont `INNODB` et `MYISAM`. Ce paramètre s'applique à Aurora MySQL version 2.
`internal_tmp_mem_storage_engine`	Parfois	Contrôle quel moteur de stockage en mémoire est utilisé pour les tables temporaires internes. Les valeurs autorisées pour les instances de base de données d'enregistreur sont `MEMORY` et `TempTable` . Pour les instances de base de données de lecteur, ce paramètre est défini sur `TempTable` et ne peut pas être modifié. Ce paramètre s'applique à Aurora MySQL version 3.
`join_buffer_size`	Oui	Aucun
`keep_files_on_create`	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
key_buffer_size	Oui	Cache de clé pour les tables MyISAM. Pour plus d'informations, consultez keycache->cache_lock mutex.
key_cache_age_threshold	Oui	Aucun
key_cache_block_size	Oui	Aucun
key_cache_division_limit	Oui	Aucun
local_infile	Oui	Aucun
lock_wait_timeout	Oui	Aucun
log-bin	Non	La définition de binlog_format sur STATEMENT MIXED, ou ROW définit automatiquement log-bin sur ON. La définition de binlog_format sur OFF définit automatiquement log-bin sur OFF. Pour plus d'informations, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).
log_bin_trust_function_creators	Oui	Aucun
log_bin_use_v1_row_events	Oui	Supprimé d'Aurora MySQL version 3.
log_error	Non	Aucun

Nom du paramètre	Adaptabilité	Remarques
log_error_suppression_list	Oui	Spécifie une liste de codes d'erreur qui ne sont pas consignés dans le journal d'erreurs MySQL. Cela vous permet d'ignorer certaines conditions d'erreur non critiques afin de préserver l'intégrité de vos journaux d'erreurs. Pour plus d'informations, consultez log_error_suppression_list dans la documentation MySQL. Ce paramètre s'applique à Aurora MySQL versions 3.03 et ultérieures.
log_output	Oui	Aucun
log_queries_not_using_indexes	Oui	Aucun
log_slave_updates	Non	Aurora MySQL version 2. Utilisez log_replica_updates dans Aurora MySQL version 3.
log_replica_updates	Non	Aurora MySQL version 3
log_throttle_queries_not_using_indexes	Oui	Aucun
log_warnings	Oui	Supprimé d'Aurora MySQL version 3.
long_query_time	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
low_priority_updates	Oui	Les opérations INSERT, UPDATE, DELETE et LOCK TABLE WRITE attendent qu'il ne reste aucune opération SELECT en attente. Ce paramètre affecte uniquement les moteurs de stockage qui utilisent uniquement le verrouillage au niveau de la table (MyISAM, MEMORY, MERGE). Ce paramètre s'applique à Aurora MySQL version 3.
max_allowed_packet	Oui	Aucun
max_binlog_cache_size	Oui	Aucun
max_binlog_size	Non	Aucun
max_binlog_stmt_cache_size	Oui	Aucun
max_connect_errors	Oui	Aucun
max_connections	Oui	La valeur par défaut est représentée par une formule. Pour plus de détails sur le calcul de la valeur DBInstanceClassMemory dans la formule, consultez Variables de formule de paramètre de bases de données. Pour connaître les valeurs par défaut en fonction de la classe d'instance, consultez Nombre maximal de connexions à une instance de base de données Aurora MySQL.

Nom du paramètre	Adaptabilité	Remarques
`max_delayed_threads`	Oui	Définit le nombre maximal de threads pour gérer les instructions `INSERT DELAYED`. Ce paramètre s'applique à Aurora MySQL version 3.
`max_error_count`	Oui	Nombre maximal de messages d'erreur, d'avertissement et de note à stocker pour affichage. Ce paramètre s'applique à Aurora MySQL version 3.
`max_execution_time`	Oui	Délai d'attente pour l'exécution des instructions `SELECT`, en millisecondes. Cette valeur peut être comprise entre `0` et `18446744073709551615` . Lorsqu'elle est définie sur `0`, il n'y a aucun délai d'attente. Pour plus d'informations, consultez max_execution_time dans la documentation MySQL.
`max_heap_table_size`	Oui	Aucun
`max_insert_delayed_threads`	Oui	Aucun
`max_join_size`	Oui	Aucun
`max_length_for_sort_data`	Oui	Supprimé d'Aurora MySQL version 3.
`max_prepared_stmt_count`	Oui	Aucun
`max_seeks_for_key`	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
max_sort_length	Oui	Aucun
max_sp_recursion_depth	Oui	Aucun
max_tmp_tables	Oui	Supprimé d'Aurora MySQL version 3.
max_user_connections	Oui	Aucun
max_write_lock_count	Oui	Aucun
metadata_locks_cache_size	Oui	Supprimé d'Aurora MySQL version 3.
min_examined_row_limit	Oui	Utilisez ce paramètre pour empêcher la journalisation des requêtes examinant un nombre de lignes inférieur au nombre spécifié. Ce paramètre s'applique à Aurora MySQL version 3.
myisam_data_pointer_size	Oui	Aucun
myisam_max_sort_file_size	Oui	Aucun
myisam_mmap_size	Oui	Aucun
myisam_sort_buffer_size	Oui	Aucun
myisam_stats_method	Oui	Aucun
myisam_use_mmap	Oui	Aucun
net_buffer_length	Oui	Aucun
net_read_timeout	Oui	Aucun
net_retry_count	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
net_write_timeout	Oui	Aucun
old-style-user-limits	Oui	Aucun
old_passwords	Oui	Supprimé d'Aurora MySQL version 3.
optimizer_prune_level	Oui	Aucun
optimizer_search_depth	Oui	Aucun
optimizer_switch	Oui	Pour plus d'informations sur les fonctions Aurora MySQL qui utilisent ce basculement, consultez Bonnes pratiques avec Amazon Aurora MySQL.
optimizer_trace	Oui	Aucun
optimizer_trace_features	Oui	Aucun
optimizer_trace_limit	Oui	Aucun
optimizer_trace_max_mem_size	Oui	Aucun
optimizer_trace_offset	Oui	Aucun
performance-schema-consumer-events-waits-current	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance-schema-consumer-events-waits-current. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
`performance-schema-instrument`	Oui	Si la colonne Source du paramètre `performance_schema` est définie sur `Modified`, le schéma de performance utilise le paramètre `performance-schema-instrument`. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
`performance_schema`	Oui	Si la colonne Source est définie sur `Modified`, Performance Insights gère automatiquement le schéma de performance. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
`performance_schema_accounts_size`	Oui	Si la colonne Source du paramètre `performance_schema` est définie sur `Modified`, le schéma de performance utilise le paramètre `performance_schema_accounts_size`. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_consumer_global_instrumentation	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_global_instrumentation. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_consumer_thread_instrumentation	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_thread_instrumentation. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_consumer_events_stages_current	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_events_stages_current. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_consumer_events_stages_history	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_events_stages_history. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_consumer_events_stages_history_long	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_events_stages_history_long. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_consumer_events_statements_current	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_events_statements_current. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_consumer_events_statements_history	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_events_statements_history. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_consumer_events_statements_history_long	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_events_statements_history_long. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_consumer_events_waits_history	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_events_waits_history. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_consumer_events_waits_history_long	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_events_waits_history_long. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_consumer_statements_digest	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_consumer_statements_digest. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_digests_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_digests_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_events_stages_history_long_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_events_stages_history_long_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_events_stages_history_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_events_stages_history_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_events_statements_history_long_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_events_statements_history_long_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_events_statements_history_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_events_statements_history_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_events_transactions_history_long_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_events_transactions_history_long_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_events_transactions_history_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_events_transactions_history_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_events_waits_history_long_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_events_waits_history_long_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_events_waits_history_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_events_waits_history_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_hosts_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_hosts_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_max_cond_classes	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_cond_classes. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_cond_instances	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_cond_instances. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_digest_length	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_digest_length. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_max_file_classes	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_file_classes. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_file_handles	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_file_handles. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_file_instances	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_file_instances. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_max_index_stat	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_index_stat. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_memory_classes	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_memory_classes. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_metadata_locks	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_metadata_locks. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_max_mutex_classes	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_mutex_classes. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_mutex_instances	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_mutex_instances. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_prepared_statements_instances	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_prepared_statements_instances. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_max_program_instances	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_program_instances. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_rwlock_classes	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_rwlock_classes. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_rwlock_instances	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_rwlock_instances. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_max_socket_classes	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_socket_classes. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_socket_instances	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_socket_instances. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_sql_text_length	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_sql_text_length. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_max_stage_classes	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_stage_classes. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_statement_classes	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_statement_classes. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_statement_stack	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_statement_stack. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_max_table_handles	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_table_handles. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_table_instances	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_table_instances. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_table_lock_stat	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_table_lock_stat. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_max_thread_classes	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_thread_classes. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_max_thread_instances	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_max_thread_instances. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
performance_schema_session_connect_attrs_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_session_connect_attrs_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
`performance_schema_setup_actors_size`	Oui	Si la colonne Source du paramètre `performance_schema` est définie sur `Modified`, le schéma de performance utilise le paramètre `performance_schema_setup_actors_size`. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
`performance_schema_setup_objects_size`	Oui	Si la colonne Source du paramètre `performance_schema` est définie sur `Modified`, le schéma de performance utilise le paramètre `performance_schema_setup_objects_size`. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.

Nom du paramètre	Adaptabilité	Remarques
performance_schema_show_processlist	Oui	Ce paramètre détermine quelle implémentation SHOW PROCESSLIST utiliser : • L'implémentation par défaut effectue des itérations sur les threads actifs depuis le gestionnaire de threads tout en conservant un mutex global. Cela peut ralentir les performances, en particulier sur des systèmes très encombrés. • L'implémentation SHOW PROCESSLIST alternative est basée sur la table processlist de schéma de performance. Cette implémentation interroge les données des threads actifs à partir du schéma de performance plutôt que du gestionnaire de threads et ne nécessite pas de mutex. Ce paramètre s'applique à Aurora MySQL versions 2.12 et ultérieures, et version 3. Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_show_processlist. Pour plus d'informations sur l'activation du schéma de performance, consultez

Nom du paramètre	Adaptabilité	Remarques
		Déterminer si Performance Insights gère le schéma de performance.
performance_schema_users_size	Oui	Si la colonne Source du paramètre performance_schema est définie sur Modified, le schéma de performance utilise le paramètre performance_schema_users_size. Pour plus d'informations sur l'activation du schéma de performance, consultez Déterminer si Performance Insights gère le schéma de performance.
pid_file	Non	Aucun
plugin_dir	Non	Aurora MySQL utilise des instances gérées dans lesquelles vous n'accédez pas directement au système de fichiers.
port	Non	Aurora MySQL gère les propriétés de connexion et applique des paramètres cohérents pour toutes les instances de base de données contenues dans un cluster.
preload_buffer_size	Oui	Taille de la mémoire tampon allouée lors du préchargement des index. Ce paramètre s'applique à Aurora MySQL version 3.
profiling_history_size	Oui	Aucun
query_alloc_block_size	Oui	Aucun
query_cache_limit	Oui	Supprimé d'Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`query_cache_min_res_unit`	Oui	Supprimé d'Aurora MySQL version 3.
`query_cache_size`	Oui	La valeur par défaut est représentée par une formule. Pour plus de détails sur le calcul de la valeur `DBInstanceClassMemory` dans la formule, consultez Variables de formule de paramètre de bases de données. Supprimé d'Aurora MySQL version 3.
`query_cache_type`	Oui	Supprimé d'Aurora MySQL version 3.
`query_cache_wlock_invalidate`	Oui	Supprimé d'Aurora MySQL version 3.
`query_prealloc_size`	Oui	Aucun
`range_alloc_block_size`	Oui	Aucun
`read_buffer_size`	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
`read_only`	Oui	Lorsque ce paramètre est activé, le serveur n'autorise aucune mise à jour, à l'exception de celles effectuées par les threads de réplica. Pour Aurora MySQL version 2, les valeurs valides sont les suivantes : • `0` – `OFF` • `1` – `ON` • `{TrueIfReplica}` — `ON` pour les réplicas en lecture. C'est la valeur par défaut. • `{TrueIfClusterReplica}` — `ON` pour les instances dans les clusters de réplica tels que les réplicas en lecture entre régions, les clusters secondaires dans une base de données globale Aurora et les déploiements bleu/vert. Nous vous recommandons d'utiliser le groupe de paramètres du cluster de bases de données dans Aurora MySQL version 2 pour vous assurer que le paramètre `read_only` est appliqué aux nouvelles instances d'enregistreur lors du basculement. **Note** Les instances de lecteur sont toujours en lecture seule,

Nom du paramètre	Adaptabilité	Remarques
		car Aurora MySQL définit `innodb_read_only` sur 1 pour tous les lecteurs. Par conséquent, `read_only` est redondant sur les instances de lecteur. Supprimé au niveau de l'instance d'Aurora MySQL version 3.
`read_rnd_buffer_size`	Oui	Aucun
`relay-log`	Non	Aucun
`relay_log_info_repository`	Oui	Supprimé d'Aurora MySQL version 3.
`relay_log_recovery`	Non	Aucun
`replica_checkpoint_group`	Oui	Aurora MySQL version 3
`replica_checkpoint_period`	Oui	Aurora MySQL version 3
`replica_parallel_workers`	Oui	Aurora MySQL version 3
`replica_pending_jobs_size_max`	Oui	Aurora MySQL version 3
`replica_skip_errors`	Oui	Aurora MySQL version 3
`replica_sql_verify_checksum`	Oui	Aurora MySQL version 3
`safe-user-create`	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
`secure_auth`	Oui	Ce paramètre est toujours activé dans Aurora MySQL version 2. Essayer de le désactiver génère une erreur. Supprimé d'Aurora MySQL version 3.
`secure_file_priv`	Non	Aurora MySQL utilise des instances gérées dans lesquelles vous n'accédez pas directement au système de fichiers.
`show_create_table_verbosity`	Oui	En raison de l'activation de cette variable, SHOW_CREATE_TABLE affiche `ROW_FORMAT` , qu'il s'agisse du format par défaut ou non. Ce paramètre s'applique à Aurora MySQL versions 2.12 et ultérieures, et version 3.
`skip-slave-start`	Non	Aucun
`skip_external_locking`	Non	Aucun
`skip_show_database`	Oui	Aucun
`slave_checkpoint_group`	Oui	Aurora MySQL version 2. Utilisez `replica_checkpoint_group` dans Aurora MySQL version 3.
`slave_checkpoint_period`	Oui	Aurora MySQL version 2. Utilisez `replica_checkpoint_period` dans Aurora MySQL version 3.
`slave_parallel_workers`	Oui	Aurora MySQL version 2. Utilisez `replica_parallel_workers` dans Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`slave_pending_jobs_size_max`	Oui	Aurora MySQL version 2. Utilisez `replica_pending_jobs_size_max` dans Aurora MySQL version 3.
`slave_sql_verify_checksum`	Oui	Aurora MySQL version 2. Utilisez `replica_sql_verify_checksum` dans Aurora MySQL version 3.
`slow_launch_time`	Oui	Aucun
`slow_query_log`	Oui	Pour accéder aux instructions concernant le chargement de journaux dans CloudWatch Logs, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs.
`slow_query_log_file`	Non	Aurora MySQL utilise des instances gérées dans lesquelles vous n'accédez pas directement au système de fichiers.
`socket`	Non	Aucun
`sort_buffer_size`	Oui	Aucun
`sql_mode`	Oui	Aucun
`sql_select_limit`	Oui	Aucun
`stored_program_cache`	Oui	Aucun
`sync_binlog`	Non	Aucun
`sync_master_info`	Oui	Aucun
`sync_source_info`	Oui	Ce paramètre s'applique à Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`sync_relay_log`	Oui	Supprimé d'Aurora MySQL version 3.
`sync_relay_log_info`	Oui	Aucun
`sysdate-is-now`	Oui	Aucun
`table_cache_element_entry_ttl`	Non	Aucun
`table_definition_cache`	Oui	La valeur par défaut est représentée par une formule. Pour plus de détails sur le calcul de la valeur `DBInstanceClassMemory` dans la formule, consultez Variables de formule de paramètre de bases de données.
`table_open_cache`	Oui	La valeur par défaut est représentée par une formule. Pour plus de détails sur le calcul de la valeur `DBInstanceClassMemory` dans la formule, consultez Variables de formule de paramètre de bases de données.
`table_open_cache_instances`	Oui	Aucun
`temp-pool`	Oui	Supprimé d'Aurora MySQL version 3.
`temptable_max_mmap`	Oui	Ce paramètre s'applique à Aurora MySQL version 3. Pour en savoir plus, consultez Nouveau comportement de table temporaire dans Aurora MySQL version 3.

Nom du paramètre	Adaptabilité	Remarques
`temptable_max_ram`	Oui	Ce paramètre s'applique à Aurora MySQL version 3. Pour en savoir plus, consultez Nouveau comportement de table temporaire dans Aurora MySQL version 3.
`temptable_use_mmap`	Oui	Ce paramètre s'applique à Aurora MySQL version 3. Pour en savoir plus, consultez Nouveau comportement de table temporaire dans Aurora MySQL version 3.
`thread_cache_size`	Oui	Le nombre de threads à mettre en cache. Ce paramètre s'applique à Aurora MySQL versions 2 et 3.
`thread_handling`	Non	Aucun
`thread_stack`	Oui	Aucun
`timed_mutexes`	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
`tmp_table_size`	Oui	Définit la taille maximale des tables temporaires en mémoire internes créées par le moteur de stockage `MEMORY` dans Aurora MySQL version 3. Dans Aurora MySQL version 3.04, définit la taille maximale des tables temporaires en mémoire internes créées par le moteur de stockage `TempTable` quand `aurora_tmptable_enable_per_table_limit` a pour valeur `ON`. Pour plus d'informations, consultez Limitation de la taille des tables temporaires internes en mémoire.
`tmpdir`	Non	Aurora MySQL utilise des instances gérées dans lesquelles vous n'accédez pas directement au système de fichiers.
`transaction_alloc_block_size`	Oui	Aucun
`transaction_isolation`	Oui	Ce paramètre s'applique à Aurora MySQL version 3. Remplace `tx_isolation` .
`transaction_prealloc_size`	Oui	Aucun
`tx_isolation`	Oui	Supprimé d'Aurora MySQL version 3. Remplacé par `transaction_isolation` .
`updatable_views_with_limit`	Oui	Aucun

Nom du paramètre	Adaptabilité	Remarques
validate-password	Non	Aucun
validate_password_dictionary_file	Non	Aucun
validate_password_length	Non	Aucun
validate_password_mixed_case_count	Non	Aucun
validate_password_number_count	Non	Aucun
validate_password_policy	Non	Aucun
validate_password_special_char_count	Non	Aucun
wait_timeout	Oui	Aurora évalue la valeur minimale de interactive_timeout et wait_timeout . Il utilise ensuite ce minimum comme délai pour mettre fin à toutes les sessions inactives, à la fois interactives et non interactives.

Paramètres MySQL ne s'appliquant pas à Aurora MySQL

En raison des différences d'architecture entre Aurora MySQL et MySQL, certains paramètres MySQL ne s'appliquent pas à Aurora MySQL.

Les paramètres MySQL suivants ne s'appliquent pas à Aurora MySQL. Cette liste n'est pas exhaustive.

- activate_all_roles_on_login – Ce paramètre ne s'applique pas à Aurora MySQL version 2. Il est disponible dans Aurora MySQL version 3.
- big_tables

- `bind_address`
- `character_sets_dir`
- `innodb_adaptive_flushing`
- `innodb_adaptive_flushing_lwm`
- `innodb_buffer_pool_chunk_size`
- `innodb_buffer_pool_instances`
- `innodb_change_buffering`
- `innodb_checksum_algorithm`
- `innodb_data_file_path`
- `innodb_dedicated_server`
- `innodb_doublewrite`
- `innodb_flush_log_at_timeout` – Ce paramètre ne s'applique pas à Aurora MySQL. Pour plus d'informations, consultez Configuration de la fréquence à laquelle le tampon du journal est vidé.
- `innodb_flush_method`
- `innodb_flush_neighbors`
- `innodb_io_capacity`
- `innodb_io_capacity_max`
- `innodb_log_buffer_size`
- `innodb_log_file_size`
- `innodb_log_files_in_group`
- `innodb_log_spin_cpu_abs_lwm`
- `innodb_log_spin_cpu_pct_hwm`
- `innodb_log_writer_threads`
- `innodb_max_dirty_pages_pct`
- `innodb_numa_interleave`
- `innodb_page_size`
- `innodb_redo_log_capacity`
- `innodb_redo_log_encrypt`
- `innodb_undo_log_encrypt`
- `innodb_undo_log_truncate`

- `innodb_undo_logs`
- `innodb_undo_tablespaces`
- `innodb_use_native_aio`
- `innodb_write_io_threads`

Variables d'état globales Aurora MySQL

Aurora MySQL inclut des variables d'état issues de la communauté MySQL et des variables propres à Aurora. Vous pouvez examiner ces variables pour savoir ce qui se passe dans le moteur de base de données. Pour plus d'informations sur les variables d'état dans la communauté MySQL, consultez Variables d'état du serveur dans la documentation MySQL 8.0 de la communauté.

Vous pouvez trouver les valeurs actuelles des variables d'état globales Aurora MySQL à l'aide d'une instruction telle que la suivante :

```
show global status like '%aurora%';
```

> **Note**
>
> Les variables d'état globales sont effacées lorsque le moteur de base de données redémarre.

Le tableau suivant décrit les variables d'état globales utilisées par Aurora MySQL.

Nom	Description
`AuroraDb_commits`	Nombre total de validations depuis le dernier redémarrage.
`AuroraDb_commit_latency`	Latence de validation agrégée depuis le dernier redémarrage.
`AuroraDb_ddl_stmt_duration`	Latence DDL agrégée depuis le dernier redémarrage.
`AuroraDb_select_stmt_duration`	Latence d'instruction `SELECT` agrégée depuis le dernier redémarrage.

Nom	Description
AuroraDb_insert_stmt_duration	Latence d'instruction INSERT agrégée depuis le dernier redémarrage.
AuroraDb_update_stmt_duration	Latence d'instruction UPDATE agrégée depuis le dernier redémarrage.
AuroraDb_delete_stmt_duration	Latence d'instruction DELETE agrégée depuis le dernier redémarrage.
Aurora_binlog_io_cache_allocated	Nombre d'octets alloués au cache d'E/S des journaux binaires.
Aurora_binlog_io_cache_read_requests	Nombre de demandes de lecture adressées au cache d'E/S des journaux binaires.
Aurora_binlog_io_cache_reads	Nombre de demandes de lecture qui ont été traitées à partir du cache d'E/S des journaux binaires.
Aurora_enhanced_binlog	Indique si le journal binaire amélioré est activé ou désactivé pour cette instance de base de données. Pour plus d'informations, consultez Configuration du binlog amélioré pour Aurora MySQL.
Aurora_external_connection_count	Nombre de connexions de base de données à l'instance de base de données, à l'exclusion des connexions au service RDS utilisées pour les vérifications d'état de la base de données.
Aurora_fast_insert_cache_hits	Compteur incrémenté quand le curseur mis en cache est récupéré et vérifié avec succès. Pour plus d'informations sur le cache d'insertion rapide, consultez Améliorations des performances Amazon Aurora MySQL.

Nom	Description
`Aurora_fast_insert_cache_misses`	Compteur incrémenté quand le curseur mis en cache n'est plus valide et qu'Aurora exécute une traversée d'index normale. Pour plus d'informations sur le cache d'insertion rapide, consultez Améliorations des performances Amazon Aurora MySQL.
`Aurora_fts_cache_memory_used`	Quantité de mémoire, en octets, qu'utilise le système de recherche de texte intégral InnoDB. Cette variable s'applique à Aurora MySQL 3.07 ou version ultérieure.
`Aurora_fwd_master_dml_stmt_count`	Nombre total d'instructions DML transférées à cette instance de base de données d'enregis treur. Cette variable s'applique à Aurora MySQL version 2.
`Aurora_fwd_master_dml_stmt_ duration`	Durée totale des instructions DML transférées à cette instance de base de données d'enregis treur. Cette variable s'applique à Aurora MySQL version 2.
`Aurora_fwd_master_errors_rp c_timeout`	Nombre de fois où une connexion transférée n'a pas pu être établie sur l'enregistreur.
`Aurora_fwd_master_errors_se ssion_limit`	Nombre de requêtes transférées qui sont rejetées en raison de `session full` sur l'enregistreur.
`Aurora_fwd_master_errors_se ssion_timeout`	Nombre de fois qu'une session de transfert est interrompue en raison d'un dépassement de délai d'attente sur l'enregistreur.
`Aurora_fwd_master_open_sessions`	Nombre de sessions transférées sur l'instanc e de base de données d'enregistreur. Cette variable s'applique à Aurora MySQL version 2.

Nom	Description
`Aurora_fwd_master_select_stmt_count`	Nombre total d'instructions `SELECT` transférées à cette instance de base de données d'enregistreur. Cette variable s'applique à Aurora MySQL version 2.
`Aurora_fwd_master_select_stmt_duration`	Durée totale des instructions `SELECT` transférées à cette instance de base de données d'enregistreur. Cette variable s'applique à Aurora MySQL version 2.
`Aurora_fwd_writer_dml_stmt_count`	Nombre total d'instructions DML transférées à cette instance de base de données d'enregistreur. Cette variable s'applique à Aurora MySQL version 3.
`Aurora_fwd_writer_dml_stmt_duration`	Durée totale des instructions DML transférées à cette instance de base de données d'enregistreur. Cette variable s'applique à Aurora MySQL version 3.
`Aurora_fwd_writer_errors_rpc_timeout`	Nombre de fois où une connexion transférée n'a pas pu être établie sur l'enregistreur.
`Aurora_fwd_writer_errors_session_limit`	Nombre de requêtes transférées qui sont rejetées en raison de `session full` sur l'enregistreur.
`Aurora_fwd_writer_errors_session_timeout`	Nombre de fois qu'une session de transfert est interrompue en raison d'un dépassement de délai d'attente sur l'enregistreur.
`Aurora_fwd_writer_open_sessions`	Nombre de sessions transférées sur l'instance de base de données d'enregistreur. Cette variable s'applique à Aurora MySQL version 3.

Nom	Description
`Aurora_fwd_writer_select_stmt_count`	Nombre total d'instructions `SELECT` transférées à cette instance de base de données d'enregistreur. Cette variable s'applique à Aurora MySQL version 3.
`Aurora_fwd_writer_select_stmt_duration`	Durée totale des instructions `SELECT` transférées à cette instance de base de données d'enregistreur. Cette variable s'applique à Aurora MySQL version 3.
`Aurora_lockmgr_buffer_pool_memory_used`	Quantité de mémoire de pool de tampons, en octets, qu'utilise le gestionnaire de verrouillage Aurora MySQL.
`Aurora_lockmgr_memory_used`	Quantité de mémoire en octets qu'utilise le gestionnaire de verrouillage Aurora MySQL.
`Aurora_ml_actual_request_cnt`	Nombre total de demandes effectuées par Aurora MySQL auprès des services de machine learning Aurora sur l'ensemble des requêtes exécutées par des utilisateurs de l'instance de base de données. Pour plus d'informations, consultez Utilisation du machine learning Amazon Aurora avec Aurora MySQL.
`Aurora_ml_actual_response_cnt`	Le nombre total de réponses reçues par Aurora MySQL de la part des services de machine learning Aurora dans toutes les requêtes exécutées par des utilisateurs de l'instance de base de données. Pour plus d'informations, consultez Utilisation du machine learning Amazon Aurora avec Aurora MySQL.

Nom	Description
`Aurora_ml_cache_hit_cnt`	Le nombre total d'accès au cache interne reçus par Aurora MySQL de la part des services de machine learning Aurora dans toutes les requêtes exécutées par des utilisateurs de l'instance de base de données. Pour plus d'informations, consultez Utilisation du machine learning Amazon Aurora avec Aurora MySQL.
`Aurora_ml_logical_request_cnt`	Nombre de demandes logiques que l'instance de base de données a évaluées pour qu'elles soient envoyées aux services de machine learning Aurora depuis la dernière réinitialisation de l'état. Si un traitement par lots a été utilisé, cette valeur peut être supérieure à `Aurora_ml_actual_request_cnt` . Pour plus d'informations, consultez Utilisation du machine learning Amazon Aurora avec Aurora MySQL.
`Aurora_ml_logical_response_cnt`	Le nombre total de réponses reçues par Aurora MySQL de la part des services de machine learning Aurora dans toutes les requêtes exécutées par des utilisateurs de l'instance de base de données. Pour plus d'informations, consultez Utilisation du machine learning Amazon Aurora avec Aurora MySQL.
`Aurora_ml_retry_request_cnt`	Nombre de nouvelles tentatives de demande que l'instance de base de données a envoyées aux services de machine learning Aurora depuis la dernière réinitialisation de l'état. Pour plus d'informations, consultez Utilisation du machine learning Amazon Aurora avec Aurora MySQL.

Nom	Description
`Aurora_ml_single_request_cnt`	Le nombre total de fonctions de machine learning Aurora évaluées par un mode autre que par lots dans toutes les requêtes exécutées par des utilisateurs de l'instanc e de base de données. Pour plus d'informa tions, consultez Utilisation du machine learning Amazon Aurora avec Aurora MySQL.
`aurora_oom_avoidance_recove ry_state`	Indique si la restauration pour éviter une mémoire insuffisante est à l'état `ACTIVE` ou `INACTIVE` pour cette instance de base de données. Cette variable s'applique à Aurora MySQL 3.06.0 ou version ultérieure.
`aurora_oom_reserved_mem_enter_kb`	Représente le seuil à atteindre pour passer à l'état `RESERVED` dans le mécanisme de gestion des insuffisances de mémoire d'Aurora. Lorsque la mémoire disponible sur le serveur tombe en dessous de ce seuil, `aurora_oo m_status` indique `RESERVED`, ce qui signifie que le serveur se rapproche d'un niveau critique d'utilisation de la mémoire. Cette variable s'applique à Aurora MySQL 3.06.0 ou version ultérieure.

Nom	Description
`aurora_oom_reserved_mem_exit_kb`	Représente le seuil à atteindre pour sortir de l'état `RESERVED` dans le mécanisme de gestion des insuffisances de mémoire d'Aurora. Lorsque la mémoire disponible sur le serveur dépasse ce seuil, `aurora_oom_status` indique `NORMAL`, ce qui signifie que le serveur est revenu à un état plus stable avec des ressources de mémoire suffisantes. Cette variable s'applique à Aurora MySQL 3.06.0 ou version ultérieure.
`aurora_oom_status`	Représente l'état actuel de cette instance de base de données en matière d'insuffisance de mémoire. Lorsque cette valeur est `NORMAL`, cela indique que les ressources de mémoire sont suffisantes. Si cette valeur passe à `RESERVED`, cela indique que la mémoire disponible du serveur est faible. Les actions sont effectuées en fonction de la configuration du paramètre `aurora_oom_response` . Pour plus d'informations, consultez Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL. Cette variable s'applique à Aurora MySQL 3.06.0 ou version ultérieure.

Nom	Description
`Aurora_pq_bytes_returned`	Nombre d'octets des structures de données à tuple transmises au nœud principal lors des requêtes parallèles. Divisez cette valeur par 16 384 pour la comparer à `Aurora_pq_pages_pushed_down` .
`Aurora_pq_max_concurrent_requests`	Nombre maximal de sessions de requêtes parallèles pouvant être exécutées simultanément sur cette instance de base de données Aurora. Il s'agit d'un nombre fixe qui dépend de la classe d'instance de base de données AWS.
`Aurora_pq_pages_pushed_down`	Nombre de pages de données (chacune avec une taille fixe de 16 Kio) pour lesquelles une requête parallèle a évité une transmission réseau au nœud principal.
`Aurora_pq_request_attempted`	Nombre de sessions de requêtes parallèles demandées. Cette valeur peut représenter plus d'une session par requête, en fonction des constructions SQL telles que les sous-requêtes et les jointures.
`Aurora_pq_request_executed`	Nombre de sessions de requêtes parallèles ayant réussi.

Nom	Description
`Aurora_pq_request_failed`	Nombre de sessions de requêtes parallèles ayant renvoyé une erreur au client. Dans certains cas, les demandes de requêtes parallèles peuvent échouer (en cas de problème au niveau de la couche de stockage, par exemple). Dans ce cas, la partie de la requête ayant échoué fait l'objet d'une autre tentative avec un mécanisme de requête non parallèle. Si cette nouvelle tentative échoue également, une erreur est renvoyée au client, et ce compteur est incrémenté.
`Aurora_pq_request_in_progress`	Nombre de sessions de requêtes parallèles en cours. Ce nombre s'applique à l'instance de base de données Aurora spécifique à laquelle vous êtes connecté, et non à l'ensemble du cluster de bases de données Aurora. Pour déterminer si une instance de base de données se rapproche de sa limite de simultanéité, comparez cette valeur à `Aurora_pq_max_concurrent_requests` .
`Aurora_pq_request_not_chosen`	Nombre de fois qu'une requête parallèle n'a pas été choisie pour accomplir une requête. Cette valeur correspond à la somme de plusieurs autres compteurs plus précis. Une instruction `EXPLAIN` peut incrémenter ce compteur même si la requête n'est pas réellement effectuée.
`Aurora_pq_request_not_chosen_below_min_rows`	Nombre de fois qu'une requête parallèle n'a pas été choisie en raison du nombre de lignes dans la table. Une instruction `EXPLAIN` peut incrémenter ce compteur même si la requête n'est pas réellement effectuée.

Nom	Description
`Aurora_pq_request_not_chosen_column_bit`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle en raison d'un type de données non pris en charge dans la liste des colonnes projetées.
`Aurora_pq_request_not_chosen_column_geometry`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table comporte des colonnes avec le type de données `GEOMETRY`. Pour plus d'informations sur les versions Aurora MySQL qui suppriment cette limitation, consultez Mise à niveau des clusters de requêtes parallèles vers Aurora MySQL version 3.
`Aurora_pq_request_not_chosen_column_lob`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table comporte des colonnes avec un type de données `LOB` ou des colonnes `VARCHAR` stockées en externe en raison de la longueur déclarée. Pour plus d'informations sur les versions Aurora MySQL qui suppriment cette limitation, consultez Mise à niveau des clusters de requêtes parallèles vers Aurora MySQL version 3.
`Aurora_pq_request_not_chosen_column_virtual`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table contient une colonne virtuelle.
`Aurora_pq_request_not_chosen_custom_charset`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table comporte des colonnes avec un jeu de caractères personnalisé.

Nom	Description
`Aurora_pq_request_not_chosen_fast_ddl`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle car la table est actuellement modifiée par une instruction DDL rapide `ALTER`.
`Aurora_pq_request_not_chosen_few_pages_outside_buffer_pool`	Nombre de fois qu'une requête parallèle n'a pas été choisie, même si moins de 95 % des données de la table se trouvait dans le pool de mémoires tampons, car il n'y avait pas suffisamment de données hors mémoire tampon pour que l'application de la fonction de requête parallèle soit justifiée.
`Aurora_pq_request_not_chosen_full_text_index`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table comporte des index de texte intégral.
`Aurora_pq_request_not_chosen_high_buffer_pool_pct`	Nombre de fois qu'une requête parallèle n'a pas été choisie, car un pourcentage élevé de données de la table (pourcentage actuellement supérieur à 95 %) se trouvait déjà dans un pool de mémoires tampons. Dans ce cas, l'optimiseur détermine que la lecture des données à partir du pool de mémoires tampons est plus efficace. Une instruction `EXPLAIN` peut incrémenter ce compteur même si la requête n'est pas réellement effectuée.
`Aurora_pq_request_not_chosen_index_hint`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la requête inclut un indicateur d'index.

Nom	Description
`Aurora_pq_request_not_chosen_innodb_table_format`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la table utilise un format de ligne InnoDB non pris en charge. Une requête parallèle Aurora s'applique uniquement aux formats de ligne `COMPACT`, `REDUNDANT` et `DYNAMIC`.
`Aurora_pq_request_not_chosen_long_trx`	Nombre de demandes de requêtes parallèles ayant utilisé le chemin de traitement de requête non parallèle en raison du lancement de la requête dans une transaction de longue durée. Une instruction `EXPLAIN` peut incrémenter ce compteur même si la requête n'est pas réellement effectuée.
`Aurora_pq_request_not_chosen_no_where_clause`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la requête n'inclut aucune clause `WHERE`.
`Aurora_pq_request_not_chosen_range_scan`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la requête utilise une analyse de plage sur un index.
`Aurora_pq_request_not_chosen_row_length_too_long`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la longueur totale combinée de toutes les colonnes est trop élevée.

Nom	Description
`Aurora_pq_request_not_chosen_small_table`	Nombre de fois qu'une requête parallèle n'a pas été choisie en raison de la taille globale de la table, telle que déterminée par le nombre de lignes dans la table et leur longueur moyenne. Une instruction `EXPLAIN` peut incrémenter ce compteur même si la requête n'est pas réellement effectuée.
`Aurora_pq_request_not_chosen_temporary_table`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la requête fait référence à des tables temporaires qui utilisent les types de table `MyISAM` ou `memory` non pris en charge.
`Aurora_pq_request_not_chosen_tx_isolation`	Nombre de demandes de requête parallèle qui utilisent le chemin de traitement de requête non parallèle, car la requête utilise un niveau d'isolement de transaction non pris en charge. Sur les instances de base de données de lecteur, la requête parallèle s'applique uniquement aux niveaux d'isolement `REPEATABLE READ` et `READ COMMITTED` .
`Aurora_pq_request_not_chosen_update_delete_stmts`	Nombre de demandes de requête parallèle utilisant le chemin de traitement de requête non parallèle, car la requête fait partie d'une instruction `UPDATE` ou `DELETE`.

Nom	Description
`Aurora_pq_request_not_chosen_unsupported_access`	Nombre de demandes de requêtes parallèles qui utilisent le chemin de traitement de requête non parallèle, car la clause `WHERE` ne remplit pas les critères des requêtes parallèles. Ce résultat peut se produire si la requête ne nécessite aucune analyse à usage intensif de données ou si la requête est une instruction `DELETE` ou `UPDATE`.
`Aurora_pq_request_not_chosen_unsupported_storage_type`	Nombre de demandes de requêtes parallèles qui utilisent le chemin de traitement des requêtes non parallèles parce que le cluster de bases de données Aurora MySQL n'utilise pas de configuration de stockage de cluster Aurora prise en charge. Pour plus d'informations, consultez Limitations. Ce paramètre s'applique à Aurora MySQL versions 3.04 et ultérieures.
`Aurora_pq_request_throttled`	Nombre de fois qu'une requête parallèle n'a pas été choisie en raison du nombre maximal de requêtes parallèles simultanées déjà exécutées sur une instance de base de données Aurora spécifique.
`Aurora_repl_bytes_received`	Nombre d'octets répliqués vers une instance de base de données de lecteur Aurora MySQL depuis le dernier redémarrage. Pour plus d'informations, consultez Réplication avec Amazon Aurora MySQL.

Nom	Description
`Aurora_reserved_mem_exceeded_incidents`	Nombre de fois depuis le dernier redémarrage où le moteur a dépassé les limites de mémoire réservées. Si `aurora_oom_response` est configuré, ce seuil définit à quel moment les activités d'évitement de mémoire insuffisante (OOM) sont déclenchées. Pour plus d'informations sur la réponse OOM d'Aurora MySQL, consultez Résolution des problèmes de mémoire insuffisante pour les bases de données Aurora MySQL.
`aurora_temptable_max_ram_allocation`	Quantité de mémoire, en octets, utilisée à tout moment par les tables temporaires internes depuis le dernier redémarrage.
`aurora_temptable_ram_allocation`	Quantité de mémoire actuelle, en octets, utilisée par les tables temporaires internes.
`Aurora_in_memory_relaylog_status`	État actuel de la fonctionnalité de journal de relais en mémoire, qui peut être ACTIVÉE ou DÉSACTIVÉE.
`Aurora_in_memory_relaylog_disabled_reason`	Indique la cause de l'état actuel de la fonctionnalité de journal de relais en mémoire. Si cette fonctionnalité est désactivée, un message explique pourquoi elle est désactivée.
`Aurora_in_memory_relaylog_fallback_count`	Affiche le nombre total de remplacements de la fonctionnalité de journal de relais en mémoire par le mode journal de relais persistant (ancien). Le remplacement peut être dû à un événement ayant une taille supérieure à celle du cache (128 Mo actuellement) ou à un dépassement de la limite de nouvelles tentatives de transaction du réplica (replica_transaction_retries).

Nom	Description
`Aurora_in_memory_relaylog_recovery_count`	Indique le nombre total de restaurations automatiques du journal de relais en mémoire. Ce décompte inclut le nombre total de remplacements et le nombre de rétablissements automatiques du mode journal de relais en mémoire après les remplacements temporaires.
`Aurora_thread_pool_thread_count`	Nombre actuel de threads dans le pool de threads Aurora. Pour plus d'informations sur le pool de threads dans Aurora MySQL, consultez Groupe de threads.
`Aurora_tmz_version`	Désigne la version actuelle des informations de fuseau horaire utilisées par le cluster de bases de données. Les valeurs suivent le format de l'IANA (Internet Assigned Numbers Authority) : `YYYYsuffix` , par exemple `2022a` et `2023c`. Ce paramètre s'applique à Aurora MySQL versions 2.12 et ultérieures, et versions 3.04 et ultérieures.
`Aurora_zdr_oom_threshold`	Représente le seuil de mémoire, en kilo-octets (Ko), à atteindre pour qu'une instance de base de données Aurora lance un redémarrage sans durée d'indisponibilité (ZDR) afin de remédier à d'éventuels problèmes liés à la mémoire.
`server_aurora_das_running`	Indique si les flux d'activité de base de données (DAS) sont activés ou désactivés sur cette instance de base de données. Pour plus d'informations, consultez Surveillance d'Amazon Aurora à l'aide des flux d'activité de base de données.

Variables d’état MySQL ne s’appliquant pas à Aurora MySQL

En raison des différences d’architecture entre Aurora MySQL et MySQL, certaines variables d’état MySQL ne s’appliquent pas à Aurora MySQL.

Les variables d’état MySQL suivantes ne s’appliquent pas à Aurora MySQL. Cette liste n’est pas exhaustive.

- `innodb_buffer_pool_bytes_dirty`
- `innodb_buffer_pool_pages_dirty`
- `innodb_buffer_pool_pages_flushed`

Aurora MySQL version 3 supprime les variables d’état suivantes précédemment disponibles dans Aurora MySQL version 2 :

- `AuroraDb_lockmgr_bitmaps0_in_use`
- `AuroraDb_lockmgr_bitmaps1_in_use`
- `AuroraDb_lockmgr_bitmaps_mem_used`
- `AuroraDb_thread_deadlocks`
- `available_alter_table_log_entries`
- `Aurora_lockmgr_memory_used`
- `Aurora_missing_history_on_replica_incidents`
- `Aurora_new_lock_manager_lock_release_cnt`
- `Aurora_new_lock_manager_lock_release_total_duration_micro`
- `Aurora_new_lock_manager_lock_timeout_cnt`
- `Aurora_total_op_memory`
- `Aurora_total_op_temp_space`
- `Aurora_used_alter_table_log_entries`
- `Aurora_using_new_lock_manager`
- `Aurora_volume_bytes_allocated`
- `Aurora_volume_bytes_left_extent`
- `Aurora_volume_bytes_left_total`
- `Com_alter_db_upgrade`

- `Compression`
- `External_threads_connected`
- `Innodb_available_undo_logs`
- `Last_query_cost`
- `Last_query_partial_plans`
- `Slave_heartbeat_period`
- `Slave_last_heartbeat`
- `Slave_received_heartbeats`
- `Slave_retried_transactions`
- `Slave_running`
- `Time_since_zero_connections`

Ces variables d'état MySQL sont disponibles dans Aurora MySQL version 2, mais pas disponibles dans Aurora MySQL version 3 :

- `Innodb_redo_log_enabled`
- `Innodb_undo_tablespaces_total`
- `Innodb_undo_tablespaces_implicit`
- `Innodb_undo_tablespaces_explicit`
- `Innodb_undo_tablespaces_active`

Événements d'attente Aurora MySQL

Voici quelques événements d'attente courants pour Aurora MySQL.

> **Note**
>
> Pour plus d'informations sur le réglage des performances d'Aurora MySQL à l'aide d'événements d'attente, consultez Réglage d'Aurora MySQL avec des événements d'attente. Pour plus d'informations sur les conventions de dénomination utilisées dans les événements d'attente MySQL, consultez Performance Schema Instrument Naming Conventions dans la documentation MySQL.

cpu

Le nombre de connexions actives prêtes à fonctionner est constamment supérieur au nombre de CPUs v. Pour plus d'informations, consultezcpu.

io/aurora_redo_log_flush

Une session conserve les données dans le stockage Aurora. Cet événement d'attente correspond généralement à une opération I/O d'écriture dans Aurora MySQL. Pour plus d'informations, consultez io/aurora_redo_log_flush.

io/aurora_respond_to_client

Le traitement des requêtes est terminé et les résultats sont renvoyés au client d'application pour les versions suivantes d'Aurora MySQL : 2.10.2 et versions 2.10 ultérieures, 2.09.3 et versions 2.09 ultérieures, 2.07.7 et versions 2.07 ultérieures. Comparez la bande passante réseau de la classe d'instance de base de données avec la taille de l'ensemble de résultats renvoyé. Vérifiez également les temps de réponse côté client. Si le client ne répond pas et n'est pas en mesure de traiter les paquets TCP, des pertes de paquets et des retransmissions TCP peuvent se produire. Cette situation affecte négativement la bande passante réseau. Dans les versions antérieures aux versions 2.10.2, 2.09.3 et 2.07.7, l'événement d'attente inclut par erreur le temps d'inactivité. Pour savoir comment régler votre base de données lorsque cette attente est importante, consultez io/aurora_respond_to_client.

io/file/csv/data

Les threads écrivent dans les tables au format CSV. Vérifiez votre utilisation de la table CSV. Cet événement est souvent déclenché par la définition de `log_output` sur une table.

io/file/sql/binlog

Un thread attend un fichier journal binaire (binlog) en cours d'écriture sur le disque.

io/redo_log_flush

Une session conserve les données dans le stockage Aurora. Généralement, cet événement d'attente concerne une I/O opération d'écriture dans Aurora MySQL. Pour de plus amples informations, veuillez consulter io/redo_log_flush.

io/socket/sql/client_connexion

Le programme `mysqld` est occupé à créer des threads pour gérer les nouvelles connexions client entrantes. Pour de plus amples informations, veuillez consulter io/socket/sql/client_connection.

io/table/sql/handler

Le moteur attend d'accéder à une table. Cet événement se produit indépendamment du fait que les données soient mises en cache dans le groupe de mémoires tampons ou accessibles sur disque. Pour de plus amples informations, veuillez consulter io/table/sql/handler.

lock/table/sql/handler

Cet événement d'attente est un gestionnaire d'événements d'attente de verrouillage de table. Pour plus d'informations sur les événements « atom » et « molecule » du schéma de performance, consultez Performance Schema atom and molecule events dans la documentation MySQL.

synch/cond/innodb/row_bloquer_attendre

Plusieurs instructions en langage de manipulation de données (DML) accèdent simultanément aux mêmes lignes de base de données. Pour de plus amples informations, veuillez consulter synch/cond/innodb/row_lock_wait.

synch/cond/innodb/row_lock_wait_cond

Plusieurs instructions DML accèdent simultanément aux mêmes lignes de base de données. Pour de plus amples informations, veuillez consulter synch/cond/innodb/row_lock_wait_cond.

synch/cond/sql/MDL_context : :Cond_Wait_Status

Les threads attendent sur un verrou de métadonnées de table. Le moteur utilise ce type de verrou pour gérer l'accès simultané à un schéma de base de données et assurer la cohérence des données. Pour plus d'informations, consultez Optimizing Locking Operations dans la documentation MySQL. Pour savoir comment régler votre base de données lorsque cet événement est important, consultez synch/cond/sql/MDL_context::COND_wait_status.

synch/cond/sql/MYSQL_BIN_LOG : :Cond_Done

Vous avez activé la journalisation binaire. Il peut y avoir un débit de validation élevé, un grand nombre de transactions en cours de validation ou des réplicas lisant des journaux binaires. Envisagez d'utiliser des instructions à plusieurs lignes ou de regrouper des instructions dans une seule transaction. Dans Aurora, privilégiez les bases de données globales à la réplication des journaux binaires, ou utilisez le paramètres `aurora_binlog_*`.

synch/mutex/innodb/aurora_lock_thread_slot_futex

Plusieurs instructions DML accèdent simultanément aux mêmes lignes de base de données. Pour de plus amples informations, veuillez consulter synch/mutex/innodb/aurora_lock_thread_slot_futex.

synch/mutex/innodb/buf_pool_mutex

Le groupe de mémoires de tampons n'est pas suffisamment grand pour contenir l'ensemble de données de travail. Ou bien, la charge de travail accède aux pages d'une table spécifique, ce qui entraîne une contention dans le groupe de mémoires tampons. Pour de plus amples informations, veuillez consulter synch/mutex/innodb/buf_pool_mutex.

synch/mutex/innodb/fil_system_mutex

Le processus est en attente d'accès au cache mémoire de l'espace disque logique. Pour de plus amples informations, veuillez consulter synch/mutex/innodb/fil_system_mutex.

synch/mutex/innodb/trx_sys_mutex

Les opérations vérifient, mettent à jour, suppriment ou ajoutent des transactions IDs dans InnoDB de manière cohérente ou contrôlée. Ces opérations nécessitent un appel de mutex `trx_sys`, suivi par l'instrumentation Performance Schema. Parmi les opérations figurent les suivantes : gestion du système de transactions lorsque la base de données démarre ou s'arrête, restaurations, nettoyages après annulation, accès en lecture de ligne et charges de groupe de mémoires tampons. Une charge de base de données élevée avec un grand nombre de transactions entraîne l'apparition fréquente de cet événement d'attente. Pour de plus amples informations, veuillez consulter synch/mutex/innodb/trx_sys_mutex.

synch/mutex/mysys/KEY_CACHE : :cache_lock

Le mutex `keycache->cache_lock` contrôle l'accès au cache de clé pour les tables MyISAM. Bien qu'Aurora MySQL n'autorise pas l'utilisation des tables MyISAM pour stocker des données persistantes, elles sont utilisées pour stocker des tables temporaires internes de stockage. Envisagez de vérifier les compteurs d'état `created_tmp_tables` et `created_tmp_disk_tables`, car dans certaines situations, les tables temporaires sont écrites sur le disque lorsqu'elles ne tiennent plus dans la mémoire.

synch/mutex/sql/FILE_AS_TABLE : :Lock_Offsets

Le moteur acquiert ce mutex lors de l'ouverture ou de la création d'un fichier de métadonnées de table. Lorsque cet événement d'attente se produit à une fréquence excessive, le nombre de tables créées ou ouvertes a connu un pic.

synch/mutex/sql/FILE_AS_TABLE : :Lock_Shim_Lists

Le moteur acquiert ce mutex tout en effectuant des opérations telles que `reset_size`, `detach_contents` ou `add_contents` sur la structure interne qui permet de suivre les tables ouvertes. Le mutex synchronise l'accès au contenu de la liste. Lorsque cet événement d'attente

se produit très fréquemment, cela indique un changement soudain de l'ensemble de tables précédemment consultées. Le moteur doit accéder à de nouvelles tables ou renoncer au contexte lié aux tables précédemment consultées.

synch/mutex/sql/LOCK_ouvert

Le nombre de tables que vos sessions ouvrent dépasse la taille du cache de définition de table ou du cache ouvert de table. Augmentez la taille de ces caches. Pour plus d'informations, consultez How MySQL opens and closes tables.

synch/mutex/sql/LOCK_cache_table

Le nombre de tables que vos sessions ouvrent dépasse la taille du cache de définition de table ou du cache ouvert de table. Augmentez la taille de ces caches. Pour plus d'informations, consultez How MySQL opens and closes tables.

synch/mutex/sql/LOG

Dans cet événement d'attente, certains threads attendent un verrouillage des journaux. Par exemple,un thread peut attendre qu'un verrouillage écrive dans le fichier journal de requêtes lentes.

synch/mutex/sql/MYSQL_BIN_LOG : :LOCK_COMMIT

Dans cet événement d'attente, un thread attend d'acquérir un verrouillage avec l'intention de valider dans le journal binaire. Des conflits de journalisation binaire peuvent se produire sur des bases de données présentant un rythme de changement très élevé. Selon votre version de MySQL, certains verrouillages sont utilisés pour protéger la cohérence et la longévité du journal binaire. Dans RDS pour MySQL, les journaux binaires sont utilisés pour la réplication et le processus de sauvegarde automatisée. Dans Aurora MySQL, les journaux binaires ne sont pas nécessaires pour la réplication native ou les sauvegardes. Ils sont désactivés par défaut, mais ils peuvent être activés et utilisés pour la réplication externe ou la capture des données modifiées. Pour plus d'informations, consultez The Binary Log dans la documentation MySQL.

sync/mutex/sql/MYSQL_BIN_LOG : :Lock_Dump_Thread_Metrics_Collection

Si la journalisation binaire est activée, le moteur acquiert ce mutex lorsqu'il imprime des métriques de threads de vidage actifs dans le journal des erreurs du moteur et sur le mappage des opérations internes.

sync/mutex/sql/MYSQL_BIN_LOG : :LOCK_INACTIVE_BINLOGS_MAP

Si la journalisation binaire est activée, le moteur acquiert ce mutex lorsqu'il ajoute, supprime ou effectue une recherche dans la liste des fichiers binlog antérieurs au plus récent.

sync/mutex/sql/MYSQL_BIN_LOG : :LOCK_IO_CACHE

Si la journalisation binaire est activée, le moteur acquiert ce mutex lors des opérations de cache d'I/O du journal binaire Aurora : allouer, redimensionner, libérer, écrire, lire, purger et accéder aux informations du cache. Si cet événement se produit fréquemment, le moteur accède au cache où les événements de journal binaire sont stockés. Pour réduire les temps d'attente, réduisez les validations. Essayez de regrouper plusieurs instructions dans une seule transaction.

synch/mutex/sql/MYSQL_BIN_LOG : :LOCK_LOG

Vous avez activé la journalisation binaire. Il peut y avoir un débit de validation élevé, de nombreuses transactions en cours de validation ou des réplicas lisant des journaux binaires. Envisagez d'utiliser des instructions à plusieurs lignes ou de regrouper des instructions dans une seule transaction. Dans Aurora, privilégiez les bases de données globales à la réplication des journaux binaires, ou utilisez le paramètres `aurora_binlog_*`.

synch/mutex/sql/SERVER_THREAD : :Lock_Sync

Le mutex `SERVER_THREAD::LOCK_sync` est acquis lors de la planification, du traitement ou du lancement de threads pour les écritures de fichier. Si cet événement d'attente se produit de manière excessive, cela indique une activité d'écriture accrue dans la base de données.

synch/mutex/sql/TABLESPACES:verrou

Le moteur acquiert le mutex `TABLESPACES:lock` lors des opérations d'espace disque logique suivantes : créer, supprimer, tronquer et étendre. Si cet événement d'attente se produit de manière excessive, cela indique une fréquence élevée d'opérations d'espace disque logique. Le chargement d'une grande quantité de données dans la base de données en est un exemple.

synch/rwlock/innodb/dict

Dans cet événement d'attente, certains threads attendent un rwlock maintenu sur le dictionnaire de données InnoDB.

synch/rwlock/innodb/dict_operation_lock

Dans cet événement d'attente, certains threads maintiennent des verrouillages sur les opérations de dictionnaire de données InnoDB.

synch/rwlock/innodb/dictsystème RW lock

Un grand nombre d'instructions simultanées du langage de contrôle des données (DCLs) dans le code du langage de définition des données (DDLs) sont déclenchées simultanément. Réduisez la dépendance de l'application par rapport à DDLs l'activité normale de l'application.

synch/rwlock/innodb/index_tree_rw_lock

Un grand nombre d’instructions en langage de manipulation de données (DML) accèdent simultanément au même objet de base de données. Essayez d’utiliser des instructions à plusieurs lignes. Répartissez également la charge de travail sur différents objets de base de données. Par exemple, implémentez le partitionnement.

synch/sxlock/innodb/dict_operation_lock

Un grand nombre d'instructions simultanées du langage de contrôle des données (DCLs) dans le code du langage de définition des données (DDLs) sont déclenchées simultanément. Réduisez la dépendance de l'application par rapport à DDLs l'activité normale de l'application.

synch/sxlock/innodb/dict_sys_lock

Un grand nombre d'instructions simultanées du langage de contrôle des données (DCLs) dans le code du langage de définition des données (DDLs) sont déclenchées simultanément. Réduisez la dépendance de l'application par rapport à DDLs l'activité normale de l'application.

synch/sxlock/innodb/hash_verrous de table

La session n’a pas pu trouver de pages dans le groupe de mémoires tampons. Le moteur doit lire un fichier ou modifier la liste utilisée le moins récemment (LRU) pour le groupe de mémoires tampons. Envisagez d’augmenter la taille du cache de mémoire tampon et d’améliorer les chemins d’accès pour les requêtes pertinentes.

synch/sxlock/innodb/index_tree_rw_lock

Un grand nombre d’instructions similaires en langage de manipulation de données (DML) accèdent simultanément au même objet de base de données. Essayez d’utiliser des instructions à plusieurs lignes. Répartissez également la charge de travail sur différents objets de base de données. Par exemple, implémentez le partitionnement.

synch/mutex/innodb/temp_pool_manager_mutex

Cet événement d'attente se produit lorsqu'une session attend d'acquérir un mutex pour gérer le pool de tablespaces temporaires de session.

Pour plus d’informations sur le dépannage des événements d’attente SYNCH, consultez Pourquoi mon instance DB MySQL affiche-t-il un nombre élevé de séances actives en attente sur les événements d’attente SYNCH dans Performance Insights ?

États de thread Aurora MySQL

Voici quelques états de thread courants pour Aurora MySQL.

checking permissions

Le thread vérifie si le serveur dispose des privilèges requis pour exécuter l'instruction.

checking query cache for query

Le serveur vérifie si la requête actuelle est présente dans le cache de requête.

cleaned up

Il s'agit de l'état final d'une connexion dont la tâche est terminée, mais qui n'a pas été fermée par le client. La meilleure solution consiste à fermer explicitement la connexion dans le code. Vous pouvez également définir une valeur inférieure pour `wait_timeout` dans votre groupe de paramètres.

closing tables

Le thread vide les données de table modifiées sur le disque et ferme les tables utilisées. Si l'opération se prolonge, vérifiez les métriques de consommation de bande passante réseau par rapport à la bande passante réseau de classe d'instance. Vérifiez également que les valeurs des paramètres pour `table_open_cache`et `table_definition_cache` permettent d'ouvrir simultanément un nombre suffisant de tables pour éviter au moteur de devoir ouvrir et fermer fréquemment les tables. Ces paramètres ont une incidence sur la consommation de mémoire sur l'instance.

converting HEAP to MyISAM

La requête convertit une table temporaire de table en mémoire à table sur disque. Cette conversion est nécessaire car les tables temporaires créées par MySQL au cours des étapes intermédiaires du traitement des requêtes sont devenues trop volumineuses pour la mémoire. Vérifiez les valeurs de `tmp_table_size` et `max_heap_table_size`. Dans les versions ultérieures, ce nom d'état de thread est`converting HEAP to ondisk`.

converting HEAP to ondisk

Le thread convertit une table temporaire interne de table en mémoire à table sur disque.

copy to tmp table

Le thread traite une instruction `ALTER TABLE`. Cet état intervient après la création de la table avec la nouvelle structure, mais avant que des lignes n'y soient copiées. En présence d'un thread

dans cet état, vous pouvez utiliser le schéma de performance pour obtenir des informations relatives à la progression de l'opération de copie.

creating sort index

Aurora MySQL effectue un tri, car il n'est pas en mesure d'utiliser un index existant pour satisfaire la clause `ORDER BY` ou `GROUP BY` d'une requête. Pour plus d'informations, consultez creating sort index.

creating table

Le thread crée une table permanente ou temporaire.

delayed commit ok done

Dans Aurora MySQL, une validation asynchrone a reçu un accusé de réception et est terminée.

delayed commit ok initiated

Le thread Aurora MySQL a démarré le processus de validation asynchrone, mais attend l'accusé de réception. Il s'agit généralement du moment auquel une transaction est validée.

delayed send ok done

Un thread de travail Aurora MySQL lié à une connexion peut être libéré lorsqu'une réponse est envoyée au client. Le thread peut entamer d'autres tâches. L'état `delayed send ok` signifie que l'accusé de réception asynchrone adressé au client est terminé.

delayed send ok initiated

Un thread de travail Aurora MySQL a envoyé une réponse de manière asynchrone à un client et est désormais libre pour d'autres connexions. La transaction a lancé un processus de validation asynchrone qui n'a pas encore été confirmé.

executing

Le thread a commencé à exécuter une instruction.

freeing items

Le thread a exécuté une commande. La libération de certains éléments durant cet état implique le cache de requête. Cet état est généralement suivi d'un nettoyage.

init

Cet état intervient avant l'initialisation des instructions `ALTER TABLE`, `DELETE`, `INSERT`, `SELECT` ou `UPDATE`. Les actions correspondant à cet état incluent le vidage du journal binaire ou du journal InnoDB, et le nettoyage du cache de requête.

La source a envoyé tous les journaux binaires au réplica ; en attente de nouvelles mises à jour

Le nœud primaire a terminé sa part de la réplication. Le thread attend l'exécution d'autres requêtes pour pouvoir écrire dans le journal binaire (binlog).

opening tables

Le thread essaie d'ouvrir une table. Cette opération est rapide, sauf si une instruction `ALTER TABLE` ou `LOCK TABLE`doit se terminer, ou si elle dépasse la valeur de `table_open_cache`.

optimizing

Le serveur effectue des optimisations initiales pour une requête.

preparing

Cet état intervient lors de l'optimisation des requêtes.

query end

Cet état intervient après le traitement d'une requête, mais avant l'état de libération des éléments.

removing duplicates

Aurora MySQL n'a pas pu optimiser une opération `DISTINCT` au début d'une requête. Aurora MySQL doit supprimer toutes les lignes dupliquées avant d'envoyer le résultat au client.

searching rows for update

Le thread recherche toutes les lignes correspondantes avant de les mettre à jour. Cette étape est nécessaire si la `UPDATE` modifie l'index utilisé par le moteur pour trouver les lignes.

sending binlog event to slave

Le thread lit un événement à partir du journal binaire et l'envoie au réplica.

sending cached result to client

Le serveur extrait le résultat d'une requête du cache de requête et l'envoie au client.

sending data

Le thread lit et traite les lignes d'une instruction `SELECT`, mais n'a pas encore commencé à envoyer des données au client. Le processus consiste à identifier les pages qui contiennent les résultats nécessaires pour satisfaire la requête. Pour plus d'informations, consultez envoi de données.

sending to client

Le serveur écrit un paquet sur le client. Dans les versions antérieures de MySQL, cet événement d'attente était labélisé `writing to net`.

démarrage

Il s'agit de la première étape intervenant au début de l'exécution de l'instruction.

statistics

Le serveur calcule des statistiques pour développer un plan d'exécution des requêtes. Si un thread perdure dans cet état, le serveur est probablement lié au disque pendant qu'il effectue d'autres tâches.

storing result in query cache

Le serveur stocke le résultat d'une requête dans le cache de requête.

system lock

Le thread a appelé `mysql_lock_tables`, mais l'état du thread n'a pas été mis à jour depuis l'appel. Cet état général intervient pour de nombreuses raisons.

mise à jour

Le thread se prépare à commencer à mettre à jour la table.

updating

Le thread recherche des lignes et les met à jour.

user lock

Le thread a émis un appel `GET_LOCK`. Le thread a demandé un verrou consultatif et l'attend, ou envisage de le demander.

waiting for more updates

Le nœud primaire a terminé sa part de la réplication. Le thread attend l'exécution d'autres requêtes pour pouvoir écrire dans le journal binaire (binlog).

waiting for schema metadata lock

Il s'agit de l'attente d'un verrou de métadonnées.

waiting for stored function metadata lock

Il s’agit de l’attente d’un verrou de métadonnées.

waiting for stored procedure metadata lock

Il s’agit de l’attente d’un verrou de métadonnées.

waiting for table flush

Le thread exécute `FLUSH TABLES` et attend que tous les threads de ferment leurs tables. Ou bien, le thread a reçu une notification indiquant que la structure sous-jacente d’une table a changé et qu’il lui faut rouvrir cette table pour obtenir la nouvelle structure. Pour rouvrir cette table, le thread doit attendre que tous les autres threads l’aient fermée. Cette notification intervient si un autre thread a utilisé une des instructions suivantes sur la table : `FLUSH TABLES`, `ALTER TABLE`, `RENAME TABLE`, `REPAIR TABLE`, `ANALYZE TABLE` ou `OPTIMIZE TABLE`.

waiting for table level lock

Une session maintient un verrou sur une table tandis qu’une autre tente d’acquérir le même verrou sur la même table.

waiting for table metadata lock

Aurora MySQL utilise le verrouillage des métadonnées pour gérer l’accès simultané aux objets de base de données et assurer la cohérence des données. Dans cet événement d’attente, une session maintient un verrou de métadonnées sur une table tandis qu’une autre tente d’acquérir le même verrou sur la même table. Lorsque le schéma de performance est activé, cet état de thread est signalé en tant qu’événement d’attente `synch/cond/sql/MDL_context::COND_wait_status`.

writing to net

Le serveur écrit un paquet sur le réseau. Dans les versions ultérieures de MySQL, cet événement d’attente est labélisé `Sending to client`.

Niveaux d’isolement Aurora MySQL

Découvrez comment les instances de base de données d’un cluster Aurora MySQL implémentent la propriété d’isolement d’une base de données. Cette rubrique explique comment le comportement par défaut d’Aurora MySQL cherche l’équilibre entre une cohérence stricte et des performances élevées. Vous pouvez utiliser ces informations pour décider quand modifier les paramètres par défaut en fonction des caractéristiques de votre charge de travail.

Niveaux d’isolement disponibles pour les instances d’enregistreur

Vous pouvez utiliser les niveaux d’isolement `REPEATABLE READ`, `READ COMMITTED`, `READ UNCOMMITTED` et `SERIALIZABLE` sur l’instance principale d’un cluster de bases de données Aurora MySQL. Ces niveaux d’isolement fonctionnent de la même façon dans Aurora MySQL que dans RDS for MySQL.

Niveaux d’isolement REPEATABLE READ pour les instances de lecteur

Par défaut, les instances de base de données Aurora MySQL configurées comme réplicas Aurora en lecture seule utilisent toujours le niveau d’isolement `REPEATABLE READ`. Ces instances de bases de données ignorent toutes les instructions `SET TRANSACTION ISOLATION LEVEL` et continuent à utiliser le niveau d’isolement `REPEATABLE READ`.

Niveaux d’isolement READ COMMITTED pour les instances en lecture

Si votre application inclut une charge de travail gourmande en écriture sur l’instance principale et des requêtes de longue durée sur les réplicas Aurora, il se peut que vous soyez confronté à un retard de purge conséquent. Le retard de purge se produit quand le nettoyage interne de la mémoire est bloqué par les requêtes de longue durée. Le symptôme que vous voyez est une valeur élevée pour `history list length` dans la sortie de la commande `SHOW ENGINE INNODB STATUS`. Vous pouvez superviser cette valeur dans CloudWatch à l’aide de la métrique `RollbackSegmentHistoryListLength`. Un retard de purge substantiel peut réduire l’efficacité des index secondaires, réduire les performances globales des requêtes et conduire à un gaspillage de l’espace de stockage.

Si vous rencontrez de tels problèmes, vous pouvez définir un paramètre de configuration de niveau session Aurora MySQL, `aurora_read_replica_read_committed`, pour utiliser le niveau d’isolement `READ COMMITTED` sur les réplicas Aurora. Quand vous appliquez ce paramètre, vous pouvez aider à diminuer les ralentissements et l’espace gaspillé qui peuvent résulter de l’exécution de requêtes de longue durée simultanément aux transactions qui modifient vos tables.

Nous vous recommandons de bien comprendre le comportement spécifique d’Aurora MySQL de l’isolement `READ COMMITTED` avant d’utiliser ce paramètre. Le comportement `READ COMMITTED` du réplica Aurora est conforme à la norme ANSI SQL. Cependant, le niveau d’isolement est moins strict que le comportement `READ COMMITTED` MySQL classique que vous connaissez sans doute. Par conséquent, vous pouvez voir des résultats de requête sous `READ COMMITTED` sur un réplica en lecture Aurora MySQL différents de ceux de la même requête sous `READ COMMITTED` sur l’instance principale Aurora MySQL ou sur RDS for MySQL. Vous pouvez envisager d’utiliser le paramètre

`aurora_read_replica_read_committed` pour ces cas d'utilisation en tant que rapport exhaustif qui couvre une base de données très volumineuse. En revanche, vous pouvez l'éviter dans le cas de courtes requêtes avec des ensembles de résultats réduits, où la précision et la reproductibilité sont importantes.

Le niveau d'isolement `READ COMMITTED` n'est pas disponible pour les sessions dans un cluster secondaire dans une base de données Aurora globale qui utilisent la fonction de transfert d'écriture. Pour plus d'informations sur le transfert d'écriture, consultez Utilisation du transfert d'écriture dans une base de données globale Amazon Aurora.

Utilisation de READ COMMITTED pour les lecteurs

Pour utiliser le niveau d'isolement `READ COMMITTED` pour les réplicas Aurora, définissez le paramètre de configuration `aurora_read_replica_read_committed` sur `ON`. Utilisez ce paramètre au niveau de la session tout en étant connecté à un réplica Aurora spécifique. Pour ce faire, exécutez les commandes SQL suivantes :

```
set session aurora_read_replica_read_committed = ON;
set session transaction isolation level read committed;
```

Vous pouvez utiliser ce paramètre de configuration temporairement pour exécuter des requêtes uniques interactives. Il se peut aussi que vous souhaitiez exécuter un rapport ou une application d'analyse des données qui tire profit du niveau d'isolement `READ COMMITTED`, tout en conservant le paramètre par défaut inchangé pour les autres applications.

Quand le paramètre `aurora_read_replica_read_committed` est activé, utilisez la commande `SET TRANSACTION ISOLATION LEVEL` pour spécifier le niveau d'isolement pour les transactions appropriées.

```
set transaction isolation level read committed;
```

Différences de comportement READ COMMITTED sur les réplicas Aurora

Le paramètre `aurora_read_replica_read_committed` garantit la disponibilité du niveau d'isolement `READ COMMITTED` pour un réplica Aurora, avec une cohérence optimisée pour les transactions de longue durée. Le niveau d'isolement `READ COMMITTED` sur les réplicas Aurora offre un isolement moins strict que sur les instances principales Aurora. Pour cette raison, l'activation de ce paramètre uniquement sur les réplicas Aurora où vous savez que vos requêtes peuvent accepter la possibilité de certains types de résultats incohérents.

Vos requêtes peuvent expérimenter certains types d'anomalies en lecture quand le paramètre `aurora_read_replica_read_committed` est activé. Deux types d'anomalies sont particulièrement importants à comprendre et à gérer dans le code de votre application. Une lecture non reproductible se produit quand une autre transaction est validée tandis que votre requête est en cours d'exécution. Une requête de longue durée peut voir des données au démarrage de la requête différentes de celles qu'il voit à la fin. Une lecture fantôme se produit quand d'autres transactions entraînent une réorganisation des lignes existantes tandis que votre requête est en cours d'exécution, et qu'une ou plusieurs lignes sont lues deux fois par votre requête.

Vos requêtes peuvent expérimenter des nombres de lignes incohérents comme résultat des lectures fantômes. Vos requêtes peuvent aussi retourner des résultats incomplets ou incohérents suite à des lectures non reproductibles. Par exemple, supposons qu'une opération de jointure fasse référence à des tables modifiées simultanément par des instructions SQL, telles que `INSERT` ou `DELETE`. Dans ce cas, la requête de jointure peut lire une ligne d'une autre table, mais pas la ligne correspondante d'une autre table.

La norme ANSI SQL permet les deux comportements pour le niveau d'isolement `READ COMMITTED`. Cependant, ces comportements diffèrent de l'implémentation MySQL typique de `READ COMMITTED`. Ainsi, avant d'activer le paramètre `aurora_read_replica_read_committed`, vérifiez le code SQL existant pour vous assurer qu'il fonctionne comme attendu selon le modèle de cohérence le moins strict.

Les nombres de lignes et autres résultats peuvent ne pas offrir une cohérence forte sous le niveau d'isolement `READ COMMITTED` lorsque ce paramètre est activé. Ainsi, vous n'activez généralement le paramètre que lors de l'exécution de requêtes analytiques qui agrègent d'importantes quantités de données et ne nécessitent pas une précision absolue. Si vous n'avez pas ces types de requêtes de longue durée en même temps qu'une charge de travail gourmande en écriture, vous n'avez probablement pas besoin du paramètre `aurora_read_replica_read_committed`. Sans la combinaison de requêtes de longue durée et une charge de travail gourmande en écriture, il est peu probable que vous rencontriez des problèmes avec la longueur de la liste de l'historique.

Example Requêtes illustrant le comportement d'isolement pour READ COMMITTED sur les réplicas Aurora

L'exemple suivant montre comment les requêtes `READ COMMITTED` sur un réplica Aurora peuvent retourner des résultats non reproductibles si les transactions modifient en même temps les tables associées. La table `BIG_TABLE` contient 1 million de lignes avant le démarrage des requêtes. D'autres instructions en langage de manipulation de données (DML) ajoutent, suppriment ou modifient des lignes tandis qu'elles s'exécutent.

Les requêtes sur l'instance principale Aurora sous le niveau d'isolement `READ COMMITTED` produisent des résultats prévisibles. Cependant, la surcharge qu'entraîne la conservation d'une vue cohérente en lecture pendant la durée de vie de chaque requête de longue durée peut conduire par la suite à un nettoyage de la mémoire onéreux.

Les requêtes sur le réplica Aurora sous le niveau d'isolement `READ COMMITTED` sont optimisées pour réduire cette surcharge du nettoyage de la mémoire. Le compromis est que les résultats peuvent varier selon que les requêtes récupèrent ou non les lignes qui sont ajoutées, supprimées ou réorganisées par les transactions qui sont validées pendant que la requête est en cours d'exécution. Les requêtes sont autorisées à prendre en compte ces lignes, mais elles n'y sont pas obligées. À des fins de démonstration, les requêtes vérifient uniquement le nombre de lignes de la table à l'aide de la fonction `COUNT(*)`.

Heure	Instruction DML sur une instance principale Aurora	Requête sur une instance principale Aurora avec READ COMMITTED	Requête sur un réplica Aurora avec READ COMMITTED
T1	`INSERT INTO big_table SELECT * FROM other_table LIMIT 1000000; COMMIT;`		
T2		Q1: `SELECT COUNT(*) FROM big_table;`	Q2: `SELECT COUNT(*) FROM big_table;`
T3	`INSERT INTO big_table (c1, c2) VALUES (1, 'one more row'); COMMIT;`		
T4		Si Q1 se termine maintenant, le résultat est 1 000 000.	Si Q2 se termine maintenant, le résultat est 1 000 000 ou 1 000 001.

Heure	Instruction DML sur une instance principale Aurora	Requête sur une instance principale Aurora avec READ COMMITTED	Requête sur un réplica Aurora avec READ COMMITTED
T5	`DELETE FROM big_table LIMIT 2; COMMIT;`		
T6		Si Q1 se termine maintenant, le résultat est 1 000 000.	Si Q2 se termine maintenant, le résultat est 1 000 000, 1 000 001, 999 999 ou 999 998.
T7	`UPDATE big_table SET c2 = CONCAT(c2 ,c2,c2); COMMIT;`		
T8		Si Q1 se termine maintenant, le résultat est 1 000 000.	Si Q2 se termine maintenant, le résultat est 1 000 000 ou 1 000 001 ou 999 999, ou même un nombre supérieur.
T9		Q3: `SELECT COUNT(*) FROM big_table;`	Q4: `SELECT COUNT(*) FROM big_table;`
T10		Si Q3 se termine maintenant, le résultat est 999 999.	Si Q4 se termine maintenant, le résultat est 999 999.

Heure	Instruction DML sur une instance principale Aurora	Requête sur une instance principale Aurora avec READ COMMITTED	Requête sur un réplica Aurora avec READ COMMITTED
T11		Q5: `SELECT COUNT(*) FROM parent_table p JOIN child_table c ON (p.id = c.id) WHERE p.id = 1000;`	Q6: `SELECT COUNT(*) FROM parent_table p JOIN child_table c ON (p.id = c.id) WHERE p.id = 1000;`
T12	`INSERT INTO parent_table (id, s) VALUES (1000, 'hello'); INSERT INTO child_table (id, s) VALUES (1000, 'world'); COMMIT;`		
T13		Si Q5 se termine maintenant, le résultat est 0.	Si Q6 se termine maintenant, le résultat est 0 ou 1.

Si les requêtes se finissent rapidement avant que d'autres transactions n'exécutent les instructions DML et les validations, les résultats sont prévisibles et identiques entre l'instance principale et le réplica Aurora. Examinons les différences de comportement en détail, en commençant par la première requête.

Les résultats de Q1 sont hautement prévisibles, car `READ COMMITTED` sur l'instance principale utilise un modèle de cohérence forte similaire au niveau d'isolement `REPEATABLE READ`.

Les résultats de Q2 peuvent varier en fonction des transactions qui sont validées pendant que la requête est en cours d'exécution. Par exemple, supposons que d'autres transactions exécutent des

instructions DML et soient validées tandis que les requêtes sont en cours d'exécution. Dans ce cas, la requête sur le réplica Aurora avec le niveau d'isolement `READ COMMITTED` peut ou non prendre en compte les modifications. Les nombres de lignes ne sont pas prévisibles de la même façon que sous le niveau d'isolement `REPEATABLE READ`. De même, ils ne sont pas aussi prévisibles que les requêtes s'exécutant sous le niveau d'isolement `READ COMMITTED` sur l'instance principale ou sur une instance RDS for MySQL.

L'instruction `UPDATE` à T7 ne modifie pas réellement le nombre de lignes de la table. Cependant, en modifiant la longueur d'une colonne de longueur variable, cette instruction peut entraîner une réorganisation interne des lignes. Une transaction `READ COMMITTED` de longue durée peut afficher l'ancienne version d'une ligne, et, par la suite, au sein de la même requête, afficher la nouvelle version de la même ligne. La requête peut également ignorer l'ancienne et la nouvelle version de la ligne, de sorte que le nombre de lignes peut être différent de ce qui est attendu.

Les résultats de Q5 et Q6 peuvent être identiques ou légèrement différents. La requête Q6 sur le réplica Aurora sous `READ COMMITTED` peut afficher, mais elle n'est pas obligée à la faire, les nouvelles lignes validées pendant que la requête est en cours d'exécution. Elle peut également afficher la ligne d'une table, mais pas de l'autre table. Si la requête de jointure ne trouve pas de ligne correspondante dans les deux tables, elle retourne un nombre égal à 0 (zéro). Si la requête retrouve bel et bien les nouvelles lignes dans `PARENT_TABLE` et dans `CHILD_TABLE`, la requête retourne un nombre égal à 1 (un). Dans une requête de longue durée, les recherches à partir des tables jointes peuvent se produire à d'importants intervalles de temps.

> **Note**
>
> Ces différences de comportement dépendent du moment où les transactions sont validées et de celui où les requêtes traitent les lignes de la table sous-jacente. Ainsi, il est fort probable que vous rencontriez de telles différences dans les requêtes sur les rapports qui nécessitent plusieurs minutes ou heures, et qui s'exécutent sur des clusters Aurora traitant simultanément les transactions OLTP. Ce sont les types de charges de travail mixtes qui tirent le meilleur parti du niveau d'isolement `READ COMMITTED` sur les réplicas Aurora.

Indicateurs Aurora MySQL

Vous pouvez utiliser des indicateurs SQL avec des requêtes Aurora MySQL pour ajuster les performances. Vous pouvez également utiliser des indicateurs pour empêcher que les plans d'exécution des requêtes importantes ne changent en fonction de conditions imprévisibles.

> **Tip**
>
> Pour vérifier l'effet d'un indicateur sur une requête, examinez le plan de requête produit par l'instruction `EXPLAIN`. Comparez les plans de requête avec et sans l'indicateur.

Dans Aurora MySQL version 3, vous pouvez utiliser tous les indicateurs disponibles dans MySQL Community Edition 8.0. Pour plus d'informations sur ces indicateurs, consultez Optimizer Hints (Indicateurs d'optimiseur) dans le Manuel de référence MySQL.

Les indicateurs suivants sont disponibles dans Aurora MySQL version 2. Ces indicateurs s'appliquent aux requêtes qui utilisent la fonction de jointure par hachage dans Aurora MySQL version 2, notamment les requêtes utilisant l'optimisation via les requêtes parallèles.

PQ, NO_PQ

Spécifie s'il faut forcer l'optimiseur à utiliser une requête parallèle par table ou par requête.

`PQ` force l'optimiseur à utiliser une requête parallèle pour les tables spécifiées ou pour l'ensemble de la requête (bloc). `NO_PQ` empêche l'optimiseur d'utiliser une requête parallèle pour des tables spécifiées ou pour l'ensemble de la requête (bloc).

Cet indicateur est disponible dans Aurora MySQL versions 2.11 et ultérieures. Les exemples suivants vous montrent comment utiliser cet indicateur.

> **Note**
>
> La spécification d'un nom de table force l'optimiseur à appliquer l'indicateur `PQ`/`NO_PQ` uniquement aux tables sélectionnées. Le fait de ne pas spécifier de nom de table force l'indicateur `PQ`/`NO_PQ` sur toutes les tables concernées par le bloc de requête.

```
EXPLAIN SELECT /*+ PQ() */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ PQ(t1) */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ PQ(t1,t2) */ f1, f2
```

```
    FROM num1 t1, num1 t2 WHERE t1.f1 = t2.f21;

EXPLAIN SELECT /*+ NO_PQ() */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ NO_PQ(t1) */ f1, f2
    FROM num1 t1 WHERE f1 > 10 and f2 < 100;

EXPLAIN SELECT /*+ NO_PQ(t1,t2) */ f1, f2
    FROM num1 t1, num1 t2 WHERE t1.f1 = t2.f21;
```

HASH_JOIN, NO_HASH_JOIN

Active ou désactive la capacité de l'optimiseur de requête parallèle à choisir d'utiliser la méthode d'optimisation de jointure de hachage pour une requête. `HASH_JOIN` laisse l'optimiseur utiliser la jointure de hachage si ce mécanisme est plus efficace. `NO_HASH_JOIN` empêche l'optimiseur d'utiliser la jointure de hachage pour la requête. Cet indicateur est disponible dans Aurora MySQL versions 2.08 et ultérieures. Il est sans effet dans Aurora MySQL version 3.

Les exemples suivants vous montrent comment utiliser cet indicateur.

```
EXPLAIN SELECT/*+ HASH_JOIN(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ NO_HASH_JOIN(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

HASH_JOIN_PROBING, NO_HASH_JOIN_PROBING

Dans une requête de jointure de hachage, il indique s'il faut utiliser la table spécifiée pour le côté sonde de la jointure. La requête vérifie si les valeurs de colonne de la table de build existent dans la table de sonde, au lieu de lire l'intégralité du contenu de cette dernière. Vous pouvez utiliser `HASH_JOIN_PROBING` et `HASH_JOIN_BUILDING` pour spécifier comment les requêtes de jointure de hachage sont traitées sans réordonner les tables dans le texte de la requête. Cet indicateur est disponible dans Aurora MySQL versions 2.08 et ultérieures. Il est sans effet dans Aurora MySQL version 3.

Les exemples suivants montrent comment utiliser cet indicateur. La spécification de l'indicateur `HASH_JOIN_PROBING` pour la table `T2` a le même effet que la spécification `NO_HASH_JOIN_PROBING` pour la table `T1`.

```
EXPLAIN SELECT /*+ HASH_JOIN(t2) HASH_JOIN_PROBING(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ HASH_JOIN(t2) NO_HASH_JOIN_PROBING(t1) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

HASH_JOIN_BUILDING, NO_HASH_JOIN_BUILDING

Dans une requête de jointure de hachage, spécifie s'il faut utiliser la table spécifiée pour le côté build de la jointure. La requête traite toutes les lignes de cette table pour créer la liste des valeurs de colonne à recouper avec l'autre table. Vous pouvez utiliser `HASH_JOIN_PROBING` et `HASH_JOIN_BUILDING` pour spécifier comment les requêtes de jointure de hachage sont traitées sans réordonner les tables dans le texte de la requête. Cet indicateur est disponible dans Aurora MySQL versions 2.08 et ultérieures. Il est sans effet dans Aurora MySQL version 3.

L'exemple suivant vous montre comment utiliser cet indicateur. La spécification de l'indicateur `HASH_JOIN_BUILDING` pour la table `T2` a le même effet que la spécification `NO_HASH_JOIN_BUILDING` pour la table `T1`.

```
EXPLAIN SELECT /*+ HASH_JOIN(t2) HASH_JOIN_BUILDING(t2) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;

EXPLAIN SELECT /*+ HASH_JOIN(t2) NO_HASH_JOIN_BUILDING(t1) */ f1, f2
  FROM t1, t2 WHERE t1.f1 = t2.f1;
```

JOIN_FIXED_ORDER

Spécifie que les tables de la requête sont jointes en fonction de l'ordre dans lequel elles sont répertoriées dans la requête. Il est utile pour les requêtes impliquant trois tables ou plus. Il est destiné à remplacer l'indicateur MySQL `STRAIGHT_JOIN` et il est équivalent à l'indicateur MySQL JOIN_FIXED_ORDER. Cet indicateur est disponible dans Aurora MySQL versions 2.08 et ultérieures.

L'exemple suivant vous montre comment utiliser cet indicateur.

```
EXPLAIN SELECT /*+ JOIN_FIXED_ORDER() */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_ORDER

Spécifie l'ordre de jointure des tables de la requête. Il est utile pour les requêtes impliquant trois tables ou plus. Il est équivalent à l'indicateur MySQL JOIN_ORDER. Cet indicateur est disponible dans Aurora MySQL versions 2.08 et ultérieures.

L'exemple suivant vous montre comment utiliser cet indicateur.

```
EXPLAIN SELECT /*+ JOIN_ORDER (t4, t2, t1, t3) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_PREFIX

Spécifie les tables à placer en premier dans l'ordre de jointure. Il est utile pour les requêtes impliquant trois tables ou plus. Il est équivalent à l'indicateur MySQL JOIN_PREFIX. Cet indicateur est disponible dans Aurora MySQL versions 2.08 et ultérieures.

L'exemple suivant vous montre comment utiliser cet indicateur.

```
EXPLAIN SELECT /*+ JOIN_PREFIX (t4, t2) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

JOIN_SUFFIX

Spécifie les tables à mettre en dernier dans l'ordre de jointure. Il est utile pour les requêtes impliquant trois tables ou plus. Il est équivalent à l'indicateur MySQL JOIN_SUFFIX. Cet indicateur est disponible dans Aurora MySQL versions 2.08 et ultérieures.

L'exemple suivant vous montre comment utiliser cet indicateur.

```
EXPLAIN SELECT /*+ JOIN_SUFFIX (t1) */ f1, f2
  FROM t1 JOIN t2 USING (id) JOIN t3 USING (id) JOIN t4 USING (id);
```

Pour plus d'informations sur l'utilisation des requêtes de jointure de hachage, consultez Optimisation des requêtes de jointure MySQL Aurora volumineuses avec des jointures de hachage.

Référence des procédures stockées Aurora MySQL

Vous pouvez gérer votre cluster de bases de données Aurora MySQL en appelant des procédures stockées intégrées.

Rubriques

- Collecte et maintenance de l’historique global des statuts
- Configuration, démarrage et arrêt de la réplication des journaux binaires (binlog)
- Mettre fin à une session ou à une requête
- Réplication des transactions à l’aide des GTID
- Rotation des journaux de requêtes
- Configuration et affichage de la configuration du journal binaire

Collecte et maintenance de l’historique global des statuts

Amazon RDS fournit un ensemble de procédures qui prennent des instantanés des valeurs des variables d’état au fil du temps et les écrivent dans une table, ainsi que toutes les modifications intervenues depuis le dernier instantané. Cette infrastructure porte le nom d’historique global des statuts. Pour plus d’informations, consultez Gestion de l’historique global des statuts.

Les procédures stockées suivantes gèrent la manière dont l’historique global des statuts est collecté et conservé.

Rubriques

- mysql.rds_collect_global_status_history
- mysql.rds_disable_gsh_collector
- mysql.rds_disable_gsh_rotation
- mysql.rds_enable_gsh_collector
- mysql.rds_enable_gsh_rotation
- mysql.rds_rotate_global_status_history
- mysql.rds_set_gsh_collector
- mysql.rds_set_gsh_rotation

mysql.rds_collect_global_status_history

Prend un instantané sur demande pour l’historique global des statuts.

Syntaxe

```
CALL mysql.rds_collect_global_status_history;
```

mysql.rds_disable_gsh_collector

Désactive les instantanés pris par l’historique global des statuts.

Syntaxe

```
CALL mysql.rds_disable_gsh_collector;
```

mysql.rds_disable_gsh_rotation

Désactive la rotation de la table `mysql.global_status_history`.

Syntaxe

```
CALL mysql.rds_disable_gsh_rotation;
```

mysql.rds_enable_gsh_collector

Active l'historique global des statuts pour prendre des instantanés par défaut aux intervalles spécifiés par `rds_set_gsh_collector`.

Syntaxe

```
CALL mysql.rds_enable_gsh_collector;
```

mysql.rds_enable_gsh_rotation

Active la rotation du contenu de la table `mysql.global_status_history` en `mysql.global_status_history_old` aux intervalles spécifiés par `rds_set_gsh_rotation`.

Syntaxe

```
CALL mysql.rds_enable_gsh_rotation;
```

mysql.rds_rotate_global_status_history

Effectue une rotation du contenu de la table `mysql.global_status_history` en `mysql.global_status_history_old` à la demande.

Syntaxe

```
CALL mysql.rds_rotate_global_status_history;
```

mysql.rds_set_gsh_collector

Spécifie l'intervalle, en minutes, entre les instantanés pris par l'historique global des statuts.

Syntaxe

```
CALL mysql.rds_set_gsh_collector(intervalPeriod);
```

Paramètres

intervalPeriod

Intervalle, en minutes, entre les instantanés. La valeur par défaut est 5.

mysql.rds_set_gsh_rotation

Spécifie l'intervalle, en jours, entre deux rotations de la table `mysql.global_status_history`.

Syntaxe

```
CALL mysql.rds_set_gsh_rotation(intervalPeriod);
```

Paramètres

intervalPeriod

Intervalle, en jours, entre deux rotations de table. La valeur par défaut est 7.

Configuration, démarrage et arrêt de la réplication des journaux binaires (binlog)

Vous pouvez appeler les procédures stockées suivantes lorsque vous êtes connecté à l'instance principale dans un cluster Aurora MySQL. Ces procédures contrôlent la façon dont les transactions sont répliquées à partir d'une base de données externe dans Aurora MySQL, ou à partir de Aurora MySQL vers une base de données externe.

Rubriques

- mysql.rds_disable_session_binlog (Aurora MySQL version 2)
- mysql.rds_enable_session_binlog (Aurora MySQL version 2)
- mysql.rds_import_binlog_ssl_material
- mysql.rds_next_master_log (Aurora MySQL version 2)
- mysql.rds_next_source_log (Aurora MySQL version 3)
- mysql.rds_remove_binlog_ssl_material
- mysql.rds_reset_external_master (Aurora MySQL version 2)
- mysql.rds_reset_external_source (Aurora MySQL version 3)
- mysql.rds_set_binlog_source_ssl (Aurora MySQL version 3)
- mysql.rds_set_external_master (Aurora MySQL version 2)
- mysql.rds_set_external_source (Aurora MySQL version 3)
- mysql.rds_set_external_master_with_auto_position (Aurora MySQL version 2)
- mysql.rds_set_external_source_with_auto_position (Aurora MySQL version 3)
- mysql.rds_set_master_auto_position (Aurora MySQL version 2)
- mysql.rds_set_read_only (Aurora MySQL version 3)
- mysql.rds_set_session_binlog_format (Aurora MySQL version 2)
- mysql.rds_set_source_auto_position (Aurora MySQL version 3)
- mysql.rds_skip_repl_error
- mysql.rds_start_replication
- mysql.rds_start_replication_until (Aurora MySQL version 3)
- mysql.rds_stop_replication

mysql.rds_disable_session_binlog (Aurora MySQL version 2)

Désactive la journalisation binaire pour la session en cours en définissant la variable `sql_log_bin` sur `OFF`.

Syntaxe

```
CALL mysql.rds_disable_session_binlog;
```

Paramètres

Aucune

Notes d'utilisation

Pour un cluster de bases de données Aurora MySQL, vous appelez cette procédure stockée lorsque vous êtes connecté à l'instance principale.

Pour Aurora, cette procédure est prise en charge pour Aurora MySQL version 2.12 et les versions ultérieures, compatibles avec MySQL 5.7.

> **Note**
>
> Dans Aurora MySQL version 3, vous pouvez utiliser la commande suivante pour désactiver la journalisation binaire pour la session en cours si vous disposez du privilège `SESSION_VARIABLES_ADMIN` :
>
> ```
> SET SESSION sql_log_bin = OFF;
> ```

mysql.rds_enable_session_binlog (Aurora MySQL version 2)

Active la journalisation binaire pour la session en cours en définissant la variable `sql_log_bin` sur `ON`.

Syntaxe

```
CALL mysql.rds_enable_session_binlog;
```

Paramètres

Aucune

Notes d'utilisation

Pour un cluster de bases de données Aurora MySQL, vous appelez cette procédure stockée lorsque vous êtes connecté à l'instance principale.

Pour Aurora, cette procédure est prise en charge pour Aurora MySQL version 2.12 et les versions ultérieures, compatibles avec MySQL 5.7.

> **Note**
>
> Dans Aurora MySQL version 3, vous pouvez utiliser la commande suivante pour activer la journalisation binaire pour la session en cours si vous disposez du privilège `SESSION_VARIABLES_ADMIN` :
>
> ```
> SET SESSION sql_log_bin = ON;
> ```

mysql.rds_import_binlog_ssl_material

Importe le certificat d'autorité de certification, le certificat client et la clé client dans un cluster de bases de données Aurora MySQL. Les informations sont requises pour la communication SSL et la réplication chiffrée.

> **Note**
>
> Actuellement, cette procédure est prise en charge pour Aurora MySQL version 2 : 2.09.2, 2.10.0, 2.10.1 et 2.11.0 ; et version 3 : 3.01.1 et versions ultérieures.

Syntaxe

```
CALL mysql.rds_import_binlog_ssl_material (
  ssl_material
);
```

Paramètres

ssl_material

Données utiles JSON contenant le contenu des fichiers au format .pem suivants pour un client MySQL :

- « ssl_ca » : » » *Certificate authority certificate*
- « certificat SSL » : » » *Client certificate*
- « clé_SSL » : » » *Client key*

Notes d'utilisation

Avant d'exécuter cette procédure, préparez-vous à la réplication chiffrée :

- Si le protocole SSL n'est pas activé sur l'instance de base de données source MySQL externe et que vous ne disposez pas d'une clé client et d'un certificat client prêts, activez le protocole SSL sur le serveur de base de données MySQL et générez la clé client et le certificat client requis.
- Si le protocole SSL est activé sur l'instance de base de données source externe, fournissez une clé et un certificat client pour le cluster de bases de données Aurora MySQL. En leur absence, générez une nouvelle clé et un nouveau certificat pour le cluster de bases de données Aurora MySQL. Pour signer le certificat client, vous devez disposer de la clé d'autorité de certification utilisée pour configurer le protocole SSL sur l'instance de base de données source MySQL externe.

Pour plus d'informations, consultez Création de certificats et clés SSL à l'aide d'openssl dans la documentation MySQL.

> **Important**
>
> Après vous être préparé à la réplication chiffrée, utilisez une connexion SSL pour exécuter cette procédure. La clé du client ne doit pas être transférée au moyen d'une connexion non sécurisée.

Cette procédure permet d'importer des informations SSL entre une base de données MySQL externe et un cluster de bases de données Aurora MySQL. Les informations SSL se trouvent dans des fichiers au format .pem contenant les informations SSL du cluster de bases de données Aurora MySQL. Pendant la réplication chiffrée, le cluster de bases de données Aurora MySQL agit comme

client du serveur de base de données MySQL. Les certificats et les clés privées du client Aurora MySQL sont au format .pem dans les fichiers.

Vous pouvez copier les informations de ces fichiers dans le paramètre `ssl_material` des données utiles JSON correspondantes. Pour prendre en charge la réplication chiffrée, importez les informations SSL dans le cluster de bases de données Aurora MySQL.

Les données utiles JSON doivent être au format suivant.

```
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
ssl_ca_pem_body_code
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
ssl_cert_pem_body_code
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
ssl_key_pem_body_code
-----END RSA PRIVATE KEY-----\n"}'
```

Exemples

L'exemple suivant importe des informations SSL dans Aurora MySQL. Dans les fichiers au format .pem, le code du corps est généralement plus long que le code du corps affiché dans l'exemple.

```
call mysql.rds_import_binlog_ssl_material(
'{"ssl_ca":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_cert":"-----BEGIN CERTIFICATE-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END CERTIFICATE-----\n","ssl_key":"-----BEGIN RSA PRIVATE KEY-----
AAAAB3NzaC1yc2EAAAADAQABAAABAQClKsfkNkuSevGj3eYhCe53pcjqP3maAhDFcvBS7O6V
hz2ItxCih+PnDSUaw+WNQn/mZphTk/a/gU8jEzoOWbkM4yxyb/wB96xbiFveSFJuOp/d6RJhJOI0iBXr
lsLnBItntckiJ7FbtxJMXLvvwJryDUilBMTjYtwB+QhYXUMOzce5Pjz5/i8SeJtjnV3iAoG/cQk+0FzZ
```

```
qaeJAAHco+CY/5WrUBkrHmFJr6HcXkvJdWPkYQS3xqC0+FmUZofz221CBt5IMucxXPkX4rWi+z7wB3Rb
BQoQzd8v7yeb7OzlPnWOyN0qFU0XA246RA8QFYiCNYwI3f05p6KLxEXAMPLE
-----END RSA PRIVATE KEY-----\n"}');
```

mysql.rds_next_master_log (Aurora MySQL version 2)

Modifie la position du journal de l'instance de base de données source au début du journal binaire suivant sur l'instance de base de données source. Utilisez cette procédure uniquement si vous recevez l' I/O erreur de réplication 1236 sur une réplique en lecture.

Syntaxe

```
CALL mysql.rds_next_master_log(
curr_master_log
);
```

Paramètres

curr_master_log

Index du fichier journal maître actif. Par exemple, si le fichier en cours se nomme `mysql-bin-changelog.012345`, l'index est 12345. Pour déterminer le nom du fichier journal maître actif, exécutez la commande `SHOW REPLICA STATUS` et affichez le champ `Master_Log_File`.

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_next_master_log`.

> **Warning**
>
> Appelez `mysql.rds_next_master_log` uniquement si la réplication échoue après le basculement d'une instance de base de données multi-AZ qui est la source de réplication et si le `Last_IO_Errno` champ de `SHOW REPLICA STATUS` signale l' I/O erreur 1236. L'appel de `mysql.rds_next_master_log` peut se traduire par une perte de données dans le réplica en lecture si les transactions de l'instance source n'ont pas été écrites dans le journal binaire sur disque avant que l'événement de basculement se produise.

Exemples

Supposons que la réplication échoue sur un réplica en lecture Aurora MySQL. L'exécution de `SHOW REPLICA STATUS\G` sur le réplica en lecture renvoie le résultat suivant :

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Master: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading data from
 binary log: 'Client requested master to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
```

```
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

Le champ `Last_IO_Errno` montre que l'instance reçoit une erreur 1236 d'I/O. Le champ `Master_Log_File` montre que le nom du fichier est `mysql-bin-changelog.012345`, ce qui signifie que l'index du fichier journal est `12345`. Pour résoudre l'erreur, vous pouvez appeler `mysql.rds_next_master_log` avec le paramètre suivant :

```
CALL mysql.rds_next_master_log(12345);
```

mysql.rds_next_source_log (Aurora MySQL version 3)

Modifie la position du journal de l'instance de base de données source au début du journal binaire suivant sur l'instance de base de données source. Utilisez cette procédure uniquement si vous recevez l' I/O erreur de réplication 1236 sur une réplique en lecture.

Syntaxe

```
CALL mysql.rds_next_source_log(
curr_source_log
);
```

Paramètres

curr_source_log

Index du fichier journal source actuel. Par exemple, si le fichier en cours se nomme `mysql-bin-changelog.012345`, l'index est 12345. Pour déterminer le nom du fichier journal actuel, exécutez la commande `SHOW REPLICA STATUS` et affichez le champ `Source_Log_File`.

Notes d'utilisation

L'utilisateur administratif doit exécuter la procédure `mysql.rds_next_source_log`.

> **Warning**
>
> Appelez `mysql.rds_next_source_log` uniquement si la réplication échoue après le basculement d'une instance de base de données multi-AZ qui est la source de réplication et si le `Last_IO_Errno` champ de `SHOW REPLICA STATUS` signale l' I/O erreur 1236. L'appel de `mysql.rds_next_source_log` peut se traduire par une perte de données dans le réplica en lecture si les transactions de l'instance source n'ont pas été écrites dans le journal binaire sur disque avant que l'événement de basculement se produise. Vous pouvez réduire la probabilité que cela se produise en définissant les paramètres d'instance source `sync_binlog` et `innodb_support_xa` sur 1, même si cela peut compromettre les performances.

Exemples

Supposons que la réplication échoue sur un réplica en lecture Aurora MySQL. L'exécution de `SHOW REPLICA STATUS\G` sur le réplica en lecture renvoie le résultat suivant :

```
*************************** 1. row ***************************
             Replica_IO_State:
                  Source_Host: myhost.XXXXXXXXXXXXXX.rr-rrrr-1.rds.amazonaws.com
                  Source_User: MasterUser
                  Source_Port: 3306
                Connect_Retry: 10
              Source_Log_File: mysql-bin-changelog.012345
          Read_Source_Log_Pos: 1219393
               Relay_Log_File: relaylog.012340
                Relay_Log_Pos: 30223388
        Relay_Source_Log_File: mysql-bin-changelog.012345
           Replica_IO_Running: No
          Replica_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Source_Log_Pos: 30223232
```

```
              Relay_Log_Space: 5248928866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Source_SSL_Allowed: No
           Source_SSL_CA_File:
           Source_SSL_CA_Path:
              Source_SSL_Cert:
            Source_SSL_Cipher:
               Source_SSL_Key:
        Seconds_Behind_Source: NULL
Source_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 1236
                Last_IO_Error: Got fatal error 1236 from source when reading data from
 binary log: 'Client requested source to start replication from impossible position;
 the first event 'mysql-bin-changelog.013406' at 1219393, the last event read from
 '/rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4, the last byte read from '/
rdsdbdata/log/binlog/mysql-bin-changelog.012345' at 4.'
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Source_Server_Id: 67285976
```

Le champ `Last_IO_Errno` montre que l'instance reçoit une erreur 1236 d'I/O. Le champ `Source_Log_File` montre que le nom du fichier est `mysql-bin-changelog.012345`, ce qui signifie que l'index du fichier journal est `12345`. Pour résoudre l'erreur, vous pouvez appeler `mysql.rds_next_source_log` avec le paramètre suivant :

```
CALL mysql.rds_next_source_log(12345);
```

mysql.rds_remove_binlog_ssl_material

Supprime le certificat de l'autorité de certification, le certificat du client et la clé du client pour la communication SSL et la réplication chiffrée. Ces informations sont importées à l'aide de mysql.rds_import_binlog_ssl_material.

Syntaxe

```
CALL mysql.rds_remove_binlog_ssl_material;
```

mysql.rds_reset_external_master (Aurora MySQL version 2)

Reconfigure une instance de base de données Aurora MySQL comme n'étant plus un réplica en lecture d'une instance de MySQL s'exécutant à l'extérieur d'Amazon RDS.

> **Important**
>
> Pour exécuter cette procédure, `autocommit` doit être activé. Pour l'activer, définissez le paramètre `autocommit` sur 1. Pour plus d'informations sur la modification des paramètres d'instance, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora.

Syntaxe

```
CALL mysql.rds_reset_external_master;
```

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_reset_external_master`. Cette procédure doit être exécutée sur l'instance de base de données MySQL à supprimer comme réplica en lecture d'une instance MySQL s'exécutant en dehors d'Amazon RDS.

> **Note**
>
> Nous proposons ces procédures stockées avant tout pour permettre la réplication avec les instances MySQL s'exécutant en dehors d'Amazon RDS. Nous vous conseillons d'utiliser les réplicas Aurora pour gérer la réplication au sein d'un cluster de bases de données Aurora MySQL, dès que possible. Pour plus d'informations sur la gestion de la réplication dans des clusters de bases de données Aurora MySQL, consultez Utilisation de réplicas Aurora.

Pour plus d'informations sur l'utilisation de la réplication pour importer des données à partir d'une instance de MySQL s'exécutant à l'extérieur d'Aurora MySQL, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

mysql.rds_reset_external_source (Aurora MySQL version 3)

Reconfigure une instance de base de données Aurora MySQL comme n'étant plus un réplica en lecture d'une instance de MySQL s'exécutant à l'extérieur d'Amazon RDS.

> **Important**
>
> Pour exécuter cette procédure, `autocommit` doit être activé. Pour l'activer, définissez le paramètre `autocommit` sur 1. Pour plus d'informations sur la modification des paramètres d'instance, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora.

Syntaxe

```
CALL mysql.rds_reset_external_source;
```

Notes d'utilisation

L'utilisateur administratif doit exécuter la procédure `mysql.rds_reset_external_source`. Cette procédure doit être exécutée sur l'instance de base de données MySQL à supprimer comme réplica en lecture d'une instance MySQL s'exécutant en dehors d'Amazon RDS.

> **Note**
>
> Nous proposons ces procédures stockées avant tout pour permettre la réplication avec les instances MySQL s'exécutant en dehors d'Amazon RDS. Nous vous conseillons d'utiliser les réplicas Aurora pour gérer la réplication au sein d'un cluster de bases de données Aurora MySQL, dès que possible. Pour plus d'informations sur la gestion de la réplication dans des clusters de bases de données Aurora MySQL, consultez Utilisation de réplicas Aurora.

mysql.rds_set_binlog_source_ssl (Aurora MySQL version 3)

Active le chiffrement `SOURCE_SSL` pour la réplication des journaux binaires. Pour plus d'informations, consultez Instruction CHANGE REPLICATION SOURCE TO dans la documentation MySQL.

Syntaxe

```
CALL mysql.rds_set_binlog_source_ssl(mode);
```

Paramètres

mode

Valeur qui indique si le chiffrement SOURCE_SSL est activé :

- 0 : le chiffrement SOURCE_SSL est désactivé. La valeur par défaut est 0.
- 1 : le chiffrement SOURCE_SSL est activé. Vous pouvez configurer le chiffrement à l'aide du protocole SSL ou TLS.

Notes d'utilisation

Cette procédure est prise en charge pour les versions 3.06 et supérieures d'Aurora MySQL.

mysql.rds_set_external_master (Aurora MySQL version 2)

Configure une instance de base de données Aurora MySQL comme réplica en lecture d'une instance de MySQL s'exécutant à l'extérieur d'Amazon RDS.

La procédure `mysql.rds_set_external_master` est obsolète et sera supprimée dans une version future. Utilisez `mysql.rds_set_external_source` à la place.

> **Important**
>
> Pour exécuter cette procédure, `autocommit` doit être activé. Pour l'activer, définissez le paramètre `autocommit` sur 1. Pour plus d'informations sur la modification des paramètres d'instance, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora.

Syntaxe

```
CALL mysql.rds_set_external_master (
  host_name
```

```
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Paramètres

host_name

Nom d'hôte ou adresse IP de l'instance MySQL s'exécutant à l'extérieur d'Amazon RDS pour devenir l'instance de base de données source.

host_port

Port utilisé par l'instance MySQL s'exécutant à l'extérieur d'Amazon RDS et à configurer comme instance de base de données source. Si votre configuration réseau inclut une réplication de port Secure Shell (SSH) qui convertit le numéro de port, spécifiez le numéro de port qui est exposé par SSH.

replication_user_name

ID d'un utilisateur disposant des autorisations `REPLICATION CLIENT` et `REPLICATION SLAVE` sur l'instance MySQL s'exécutant à l'extérieur d'Amazon RDS. Nous vous recommandons de fournir un compte qui soit utilisé uniquement pour la réplication avec l'instance externe.

replication_user_password

Mot de passe de l'ID utilisateur spécifié dans `replication_user_name`.

mysql_binary_log_file_name

Nom du journal binaire sur l'instance de base de données source qui contient les informations de réplication.

mysql_binary_log_file_location

Emplacement dans le journal binaire `mysql_binary_log_file_name` à partir duquel la réplication commence à lire les informations de réplication.

Vous pouvez déterminer le nom et l'emplacement du fichier journal binaire en exécutant `SHOW MASTER STATUS` sur l'instance de base de données source.

ssl_encryption

Valeur indiquant si le chiffrement Secure Socket Layer (SSL) est utilisé sur la connexion de réplication. La valeur 1 spécifie d'utiliser le chiffrement SSL, et la valeur 0 de ne pas l'utiliser. La valeur par défaut est 0.

> **Note**
>
> L'option `MASTER_SSL_VERIFY_SERVER_CERT` n'est pas prise en charge. Cette option est définie sur 0, ce qui signifie que la connexion est chiffrée, mais que les certificats ne sont pas vérifiés.

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_set_external_master`. Cette procédure doit être exécutée sur l'instance de base de données MySQL qui doit être configurée comme réplica en lecture d'une instance MySQL s'exécutant en dehors d'Amazon RDS.

Avant d'exécuter `mysql.rds_set_external_master`, vous devez configurer l'instance de MySQL s'exécutant en dehors de Amazon RDS comme instance de base de données source. Pour vous connecter à l'instance MySQL en cours d'exécution en dehors de Amazon RDS, vous devez spécifier des valeurs `replication_user_name` et `replication_user_password` qui indiquent un utilisateur de réplication possédant des autorisations `REPLICATION CLIENT` et `REPLICATION SLAVE` sur l'instance externe de MySQL.

Pour configurer une instance externe de MySQL en tant qu'instance de base de données source

1. A l'aide du client MySQL de votre choix, connectez-vous à l'instance externe de MySQL et créez un compte d'utilisateur à utiliser pour la réplication. Voici un exemple.

   MySQL 5.7

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED WITH mysql_native_password BY
    'password';
   ```

> **Note**
>
> Spécifiez un mot de passe autre que celui indiqué ici, en tant que bonne pratique de sécurité.

2. Sur l'instance externe de MySQL, accordez les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE` à votre utilisateur de réplication. L'exemple suivant accorde les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE` sur toutes les bases de données pour l'utilisateur « repl_user » de votre domaine.

   MySQL 5.7

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
    IDENTIFIED BY 'password';
   ```

   MySQL 8.0

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
   ```

Pour utiliser la réplication chiffrée, configurez l'instance de base de données source de façon à utiliser les connexions SSL. De même, importez le certificat de l'autorité de certification, le certificat du client et la clé du client dans l'instance de base de données ou le cluster de bases de données à l'aide de la procédure mysql.rds_import_binlog_ssl_material.

> **Note**
>
> Nous proposons ces procédures stockées avant tout pour permettre la réplication avec les instances MySQL s'exécutant en dehors d'Amazon RDS. Nous vous conseillons d'utiliser les réplicas Aurora pour gérer la réplication au sein d'un cluster de bases de données Aurora MySQL, dès que possible. Pour plus d'informations sur la gestion de la réplication dans des clusters de bases de données Aurora MySQL, consultez Utilisation de réplicas Aurora.

Après avoir appelé `mysql.rds_set_external_master` pour configurer une instance de base de données Amazon RDS comme réplica en lecture, vous pouvez appeler mysql.rds_start_replication sur le réplica en lecture pour démarrer le processus de réplication. Vous pouvez appeler

mysql.rds_reset_external_master (Aurora MySQL version 2) pour supprimer la configuration du réplica en lecture.

Quand la procédure `mysql.rds_set_external_master` est appelée, Amazon RDS enregistre l'heure, l'utilisateur et une action de `set master` dans les tables `mysql.rds_history` et `mysql.rds_replication_status`.

Exemples

Lors d'une exécution sur une instance de base de données MySQL, l'exemple suivant configure l'instance de base de données comme réplica en lecture d'une instance de MySQL s'exécutant à l'extérieur d'Amazon RDS.

```
call mysql.rds_set_external_master(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_source (Aurora MySQL version 3)

Configure une instance de base de données Aurora MySQL comme réplica en lecture d'une instance de MySQL s'exécutant à l'extérieur d'Amazon RDS.

> **Important**
>
> Pour exécuter cette procédure, `autocommit` doit être activé. Pour l'activer, définissez le paramètre `autocommit` sur 1. Pour plus d'informations sur la modification des paramètres d'instance, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora.

Syntaxe

```
CALL mysql.rds_set_external_source (
  host_name
```

```
  , host_port
  , replication_user_name
  , replication_user_password
  , mysql_binary_log_file_name
  , mysql_binary_log_file_location
  , ssl_encryption
);
```

Paramètres

host_name

Nom d'hôte ou adresse IP de l'instance MySQL s'exécutant à l'extérieur d'Amazon RDS pour devenir l'instance de base de données source.

host_port

Port utilisé par l'instance MySQL s'exécutant à l'extérieur d'Amazon RDS et à configurer comme instance de base de données source. Si votre configuration réseau inclut une réplication de port Secure Shell (SSH) qui convertit le numéro de port, spécifiez le numéro de port qui est exposé par SSH.

replication_user_name

ID d'un utilisateur disposant des autorisations `REPLICATION CLIENT` et `REPLICATION SLAVE` sur l'instance MySQL s'exécutant à l'extérieur d'Amazon RDS. Nous vous recommandons de fournir un compte qui soit utilisé uniquement pour la réplication avec l'instance externe.

replication_user_password

Mot de passe de l'ID utilisateur spécifié dans `replication_user_name`.

mysql_binary_log_file_name

Nom du journal binaire sur l'instance de base de données source qui contient les informations de réplication.

mysql_binary_log_file_location

Emplacement dans le journal binaire `mysql_binary_log_file_name` à partir duquel la réplication commence à lire les informations de réplication.

Vous pouvez déterminer le nom et l'emplacement du fichier journal binaire en exécutant `SHOW MASTER STATUS` sur l'instance de base de données source.

ssl_encryption

Valeur indiquant si le chiffrement Secure Socket Layer (SSL) est utilisé sur la connexion de réplication. La valeur 1 spécifie d'utiliser le chiffrement SSL, et la valeur 0 de ne pas l'utiliser. La valeur par défaut est 0.

> **Note**
>
> Vous devez avoir importé un certificat SSL personnalisé mysql.rds_import_binlog_ssl_material pour activer cette option. Si vous n'avez pas importé de certificat SSL personnalisé, définissez ce paramètre sur 0 et utilisez mysql.rds_set_binlog_source_ssl (Aurora MySQL version 3) pour activer le protocole SSL pour la réplication des journaux binaires.
> L'option `SOURCE_SSL_VERIFY_SERVER_CERT` n'est pas prise en charge. Cette option est définie sur 0, ce qui signifie que la connexion est chiffrée, mais que les certificats ne sont pas vérifiés.

Notes d'utilisation

L'utilisateur administratif doit exécuter la procédure `mysql.rds_set_external_source`. Cette procédure doit être exécutée sur l'instance de base de données Aurora MySQL qui doit être configurée comme réplica en lecture d'une instance MySQL s'exécutant en dehors d'Amazon RDS.

Avant d'exécuter `mysql.rds_set_external_source`, vous devez configurer l'instance de MySQL s'exécutant en dehors de Amazon RDS comme instance de base de données source. Pour vous connecter à l'instance MySQL en cours d'exécution en dehors de Amazon RDS, vous devez spécifier des valeurs `replication_user_name` et `replication_user_password` qui indiquent un utilisateur de réplication possédant des autorisations `REPLICATION CLIENT` et `REPLICATION SLAVE` sur l'instance externe de MySQL.

Pour configurer une instance externe de MySQL en tant qu'instance de base de données source

1. A l'aide du client MySQL de votre choix, connectez-vous à l'instance externe de MySQL et créez un compte d'utilisateur à utiliser pour la réplication. Voici un exemple.

```
CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'password';
```

> **Note**
>
> Spécifiez un mot de passe autre que celui indiqué ici, en tant que bonne pratique de sécurité.

2. Sur l'instance externe de MySQL, accordez les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE` à votre utilisateur de réplication. L'exemple suivant accorde les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE` sur toutes les bases de données pour l'utilisateur « repl_user » de votre domaine.

```
GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com';
```

Pour utiliser la réplication chiffrée, configurez l'instance de base de données source de façon à utiliser les connexions SSL. De plus, importez le certificat de l'autorité de certification, le certificat du client et la clé du client dans l'instance de base de données ou le cluster de bases de données à l'aide de la procédure mysql.rds_import_binlog_ssl_material.

> **Note**
>
> Nous proposons ces procédures stockées avant tout pour permettre la réplication avec les instances MySQL s'exécutant en dehors d'Amazon RDS. Nous vous conseillons d'utiliser les réplicas Aurora pour gérer la réplication au sein d'un cluster de bases de données Aurora MySQL, dès que possible. Pour plus d'informations sur la gestion de la réplication dans des clusters de bases de données Aurora MySQL, consultez Utilisation de réplicas Aurora.

Après avoir appelé `mysql.rds_set_external_source` pour configurer une instance de base de données Aurora MySQL comme réplica en lecture, vous pouvez appeler mysql.rds_start_replication sur le réplica en lecture pour démarrer le processus de réplication. Vous pouvez appeler mysql.rds_reset_external_source (Aurora MySQL version 3) pour supprimer la configuration du réplica en lecture.

Quand la procédure `mysql.rds_set_external_source` est appelée, Amazon RDS enregistre l'heure, l'utilisateur et une action de `set master` dans les tables `mysql.rds_history` et `mysql.rds_replication_status`.

Exemples

Lors d'une exécution sur une instance de base de données Aurora MySQL, l'exemple suivant configure l'instance de base de données comme réplica en lecture d'une instance de MySQL s'exécutant à l'extérieur d'Amazon RDS.

```
call mysql.rds_set_external_source(
  'Externaldb.some.com',
  3306,
  'repl_user',
  'password',
  'mysql-bin-changelog.0777',
  120,
  1);
```

mysql.rds_set_external_master_with_auto_position (Aurora MySQL version 2)

Configure une instance principale Aurora MySQL afin d'accepter la réplication entrante à partir d'une instance MySQL externe. Cette procédure configure également la réplication en fonction des identifiants de transaction globaux ()GTIDs.

Cette procédure ne configure pas la réplication retardée, car Aurora MySQL ne prend pas en charge la réplication retardée.

Syntaxe

```
CALL mysql.rds_set_external_master_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
);
```

Paramètres

host_name

Nom d'hôte ou adresse IP de l'instance MySQL s'exécutant à l'extérieur d'Aurora pour devenir la source de réplication.

host_port

Port utilisé par l'instance MySQL s'exécutant à l'extérieur d'Aurora et à configurer comme source de réplication. Si votre configuration réseau inclut une réplication de port Secure Shell (SSH) qui convertit le numéro de port, spécifiez le numéro de port qui est exposé par SSH.

replication_user_name

ID d'un utilisateur disposant des autorisations `REPLICATION CLIENT` et `REPLICATION SLAVE` sur l'instance MySQL s'exécutant à l'extérieur d'Aurora. Nous vous recommandons de fournir un compte qui soit utilisé uniquement pour la réplication avec l'instance externe.

replication_user_password

Mot de passe de l'ID utilisateur spécifié dans `replication_user_name`.

ssl_encryption

Cette option n'est pas actuellement implémentée. La valeur par défaut est 0.

Notes d'utilisation

Pour un cluster de bases de données Aurora MySQL, vous appelez cette procédure stockée lorsque vous êtes connecté à l'instance principale.

L'utilisateur principal doit exécuter la procédure `mysql.rds_set_external_master_with_auto_position`. L'utilisateur principal exécute cette procédure sur l'instance principale d'un cluster de bases de données Aurora MySQL qui agit en tant que cible de réplication. Il peut s'agir de la cible de réplication d'une instance de base de données MySQL externe ou d'un cluster de bases de données Aurora MySQL.

Cette procédure est prise en charge pour Aurora MySQL version 2. Pour Aurora MySQL version 3, utilisez la procédure mysql.rds_set_external_source_with_auto_position (Aurora MySQL version 3).

Avant d'exécuter `mysql.rds_set_external_master_with_auto_position`, configurez l'instance de base de données MySQL comme source de réplication. Pour vous connecter à l'instance MySQL externe, spécifiez des valeurs pour `replication_user_name` et `replication_user_password`. Ces valeurs doivent indiquer un utilisateur de réplication disposant des autorisations `REPLICATION CLIENT` et `REPLICATION SLAVE` sur l'instance MySQL externe.

Pour configurer une instance MySQL externe comme source de réplication

1. À l'aide du client MySQL de votre choix, connectez-vous à l'instance MySQL externe et créez un compte utilisateur à utiliser pour la réplication. Voici un exemple de.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. Sur l'instance MySQL externe, attribuez les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE` à votre utilisateur de réplication. L'exemple suivant accorde les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE` sur toutes les bases de données pour l'utilisateur `'repl_user'` de votre domaine.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   IDENTIFIED BY 'SomePassW0rd'
   ```

Lorsque vous appelez `mysql.rds_set_external_master_with_auto_position`, Amazon RDS enregistre certaines informations. Il s'agit de l'heure, de l'utilisateur et d'une action de `"set master"` dans les tables `mysql.rds_history` et `mysql.rds_replication_status`.

Pour ignorer une transaction basée sur des identifiants de transaction globaux spécifique qui est réputée entraîner un problème, vous pouvez utiliser la procédure stockée mysql.rds_skip_transaction_with_gtid (Aurora MySQL versions 2 et 3). Pour plus d'informations sur la gestion d'une réplication basée sur des identifiants de transaction globaux, consultez Utilisation de la réplication basée sur des identifiants de transaction globaux (GTID).

Exemples

Lorsqu'il est exécuté sur une instance principale Aurora, l'exemple suivant configure le cluster Aurora pour qu'il agisse comme réplica en lecture d'une instance de MySQL s'exécutant à l'extérieur d'Aurora.

```
call mysql.rds_set_external_master_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd');
```

mysql.rds_set_external_source_with_auto_position (Aurora MySQL version 3)

Configure une instance principale Aurora MySQL afin d'accepter la réplication entrante à partir d'une instance MySQL externe. Cette procédure configure également la réplication en fonction des identifiants de transaction globaux ()GTIDs.

Syntaxe

```
CALL mysql.rds_set_external_source_with_auto_position (
  host_name
  , host_port
  , replication_user_name
  , replication_user_password
  , ssl_encryption
);
```

Paramètres

host_name

Nom d'hôte ou adresse IP de l'instance MySQL s'exécutant à l'extérieur d'Aurora pour devenir la source de réplication.

host_port

Port utilisé par l'instance MySQL s'exécutant à l'extérieur d'Aurora et à configurer comme source de réplication. Si votre configuration réseau inclut une réplication de port Secure Shell (SSH) qui convertit le numéro de port, spécifiez le numéro de port qui est exposé par SSH.

replication_user_name

ID d'un utilisateur disposant des autorisations `REPLICATION CLIENT` et `REPLICATION SLAVE` sur l'instance MySQL s'exécutant à l'extérieur d'Aurora. Nous vous recommandons de fournir un compte qui soit utilisé uniquement pour la réplication avec l'instance externe.

replication_user_password

Mot de passe de l'ID utilisateur spécifié dans `replication_user_name`.

ssl_encryption

Cette option n'est pas actuellement implémentée. La valeur par défaut est 0.

> **Note**
>
> Utilisez mysql.rds_set_binlog_source_ssl (Aurora MySQL version 3) pour activer SSL pour la réplication des journaux binaires.

Notes d'utilisation

Pour un cluster de bases de données Aurora MySQL, vous appelez cette procédure stockée lorsque vous êtes connecté à l'instance principale.

L'utilisateur administratif doit exécuter la procédure `mysql.rds_set_external_source_with_auto_position`. L'utilisateur administratif exécute cette procédure sur l'instance principale d'un cluster de bases de données Aurora MySQL qui agit en tant que cible de réplication. Il peut s'agir de la cible de réplication d'une instance de base de données MySQL externe ou d'un cluster de bases de données Aurora MySQL.

Cette procédure est prise en charge pour Aurora MySQL version 3. Cette procédure ne configure pas la réplication retardée, car Aurora MySQL ne prend pas en charge la réplication retardée.

Avant d'exécuter `mysql.rds_set_external_source_with_auto_position`, configurez l'instance de base de données MySQL comme source de réplication. Pour vous connecter à l'instance MySQL externe, spécifiez des valeurs pour `replication_user_name` et `replication_user_password`. Ces valeurs doivent indiquer un utilisateur de réplication disposant des autorisations `REPLICATION CLIENT` et `REPLICATION SLAVE` sur l'instance MySQL externe.

Pour configurer une instance MySQL externe comme source de réplication

1. À l'aide du client MySQL de votre choix, connectez-vous à l'instance MySQL externe et créez un compte utilisateur à utiliser pour la réplication. Voici un exemple de.

   ```
   CREATE USER 'repl_user'@'mydomain.com' IDENTIFIED BY 'SomePassW0rd'
   ```

2. Sur l'instance MySQL externe, attribuez les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE` à votre utilisateur de réplication. L'exemple suivant accorde les privilèges `REPLICATION CLIENT` et `REPLICATION SLAVE` sur toutes les bases de données pour l'utilisateur `'repl_user'` de votre domaine.

   ```
   GRANT REPLICATION CLIENT, REPLICATION SLAVE ON *.* TO 'repl_user'@'mydomain.com'
   ```

```
IDENTIFIED BY 'SomePassW0rd'
```

Lorsque vous appelez `mysql.rds_set_external_source_with_auto_position`, Amazon RDS enregistre certaines informations. Il s'agit de l'heure, de l'utilisateur et d'une action de `"set master"` dans les tables `mysql.rds_history` et `mysql.rds_replication_status`.

Pour ignorer une transaction basée sur des identifiants de transaction globaux spécifique qui est réputée entraîner un problème, vous pouvez utiliser la procédure stockée mysql.rds_skip_transaction_with_gtid (Aurora MySQL versions 2 et 3). Pour plus d'informations sur la gestion d'une réplication basée sur des identifiants de transaction globaux, consultez Utilisation de la réplication basée sur des identifiants de transaction globaux (GTID).

Exemples

Lorsqu'il est exécuté sur une instance principale Aurora, l'exemple suivant configure le cluster Aurora pour qu'il agisse comme réplica en lecture d'une instance de MySQL s'exécutant à l'extérieur d'Aurora.

```
call mysql.rds_set_external_source_with_auto_position(
  'Externaldb.some.com',
  3306,
  'repl_user'@'mydomain.com',
  'SomePassW0rd');
```

mysql.rds_set_master_auto_position (Aurora MySQL version 2)

Définit le mode de réplication en fonction des positions du fichier journal binaire ou des identificateurs de transaction globaux (GTIDs).

Syntaxe

```
CALL mysql.rds_set_master_auto_position (
auto_position_mode
);
```

Paramètres

auto_position_mode

Valeur qui indique si la réplication à utiliser est la réplication basée sur la position de fichier ou la réplication basée sur les identifiants de transaction globaux :

- `0` – Utiliser la méthode de réplication basée sur la position du fichier journal binaire. La valeur par défaut est `0`.
- `1` – Utiliser la méthode de réplication basée sur les identifiants de transaction globaux.

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_set_master_auto_position`.

Cette procédure est prise en charge pour Aurora MySQL version 2.

mysql.rds_set_read_only (Aurora MySQL version 3)

Active ou désactive le mode `read_only` de manière globale pour l'instance de base de données.

Syntaxe

```
CALL mysql.rds_set_read_only(mode);
```

Paramètres

mode

Une valeur qui indique si le mode `read_only` est activé ou désactivé globalement pour l'instance de base de données :

- `0` – `OFF`. La valeur par défaut est `0`.
- `1` – `ON`

Notes d'utilisation

La procédure `mysql.rds_set_read_only` stockée modifie uniquement le paramètre `read_only`. Le paramètre `innodb_read_only` ne peut pas être modifié sur les instances de base de données du lecteur.

Le changement du paramètre `read_only` ne persiste pas au redémarrage. Pour apporter des modifications permanentes à `read_only`, vous devez utiliser le paramètre `read_only` du cluster de bases de données.

Cette procédure est prise en charge pour les versions 3.06 et supérieures d'Aurora MySQL.

mysql.rds_set_session_binlog_format (Aurora MySQL version 2)

Définit le format de journal binaire pour la session en cours.

Syntaxe

```
CALL mysql.rds_set_session_binlog_format(format);
```

Paramètres

format

Valeur qui indique le format de journal binaire pour la session en cours :

- `STATEMENT` : la source de réplication écrit les événements dans le journal binaire sur la base d'instructions SQL.
- `ROW` : la source de réplication écrit les événements dans le journal binaire qui indiquent les modifications apportées aux lignes individuelles des tables.
- `MIXED` : la journalisation est généralement basée sur des instructions SQL, mais passe aux lignes sous certaines conditions. Pour plus d'informations, consultez Format mixte de journalisation binaire (langue française non garantie) dans la documentation MySQL.

Notes d'utilisation

Pour un cluster de bases de données Aurora MySQL, vous appelez cette procédure stockée lorsque vous êtes connecté à l'instance principale.

Pour utiliser cette procédure stockée, la journalisation binaire doit être configurée pour la session en cours.

Pour Aurora, cette procédure est prise en charge pour Aurora MySQL version 2.12 et les versions ultérieures, compatibles avec MySQL 5.7.

mysql.rds_set_source_auto_position (Aurora MySQL version 3)

Définit le mode de réplication en fonction des positions du fichier journal binaire ou des identificateurs de transaction globaux (GTIDs).

Syntaxe

```
CALL mysql.rds_set_source_auto_position (auto_position_mode);
```

Paramètres

auto_position_mode

Valeur qui indique si la réplication à utiliser est la réplication basée sur la position de fichier ou la réplication basée sur les identifiants de transaction globaux :

- `0` – Utiliser la méthode de réplication basée sur la position du fichier journal binaire. La valeur par défaut est `0`.
- `1` – Utiliser la méthode de réplication basée sur les identifiants de transaction globaux.

Notes d'utilisation

Pour un cluster de bases de données Aurora MySQL, vous appelez cette procédure stockée lorsque vous êtes connecté à l'instance principale.

L'utilisateur administratif doit exécuter la procédure `mysql.rds_set_source_auto_position`.

mysql.rds_skip_repl_error

Ignore et supprime une erreur de réplication sur un réplica en lecture d'une base de données MySQL.

Syntaxe

```
CALL mysql.rds_skip_repl_error;
```

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_skip_repl_error` sur un réplica en lecture. Pour plus d'informations sur cette procédure, consultez Ignorer une erreur de réplication.

Pour déterminer s'il y a des erreurs, exécutez la commande MySQL `SHOW REPLICA STATUS\G`. Si une erreur de réplication n'est pas critique, vous pouvez exécuter `mysql.rds_skip_repl_error` pour ignorer l'erreur. S'il y a plusieurs erreurs, `mysql.rds_skip_repl_error` supprime la première erreur, puis avertit qu'il y a d'autres erreurs. Vous pouvez alors utiliser `SHOW REPLICA STATUS\G` pour déterminer l'action appropriée pour l'erreur suivante. Pour obtenir des informations sur les valeurs renvoyées, consultez Instruction SHOW REPLICA STATUS dans la documentation sur MySQL.

Pour plus d'informations sur le traitement des erreurs de réplication avec Aurora MySQL, consultez Diagnostic et résolution d'une défaillance de la réplication en lecture MySQL.

Erreur d'arrêt de réplication

Lorsque vous appelez la procédure `mysql.rds_skip_repl_error`, un message d'erreur peut s'afficher pour indiquer que le réplica a rencontré une erreur ou est désactivé.

Ce message d'erreur s'affiche si vous exécutez la procédure sur l'instance principale plutôt que sur le réplica en lecture. Vous devez exécuter cette procédure sur le réplica en lecture pour que la procédure fonctionne.

Ce message d'erreur peut également s'afficher si vous exécutez la procédure sur le réplica en lecture, mais que la réplication ne peut pas être redémarrée correctement.

Si vous avez besoin d'ignorer un grand nombre d'erreurs, le retard de réplication peut augmenter et dépasser la période de rétention par défaut pour les fichiers journaux binaires (binlog). Dans ce cas, vous pouvez rencontrer une erreur irrécupérable due à des fichiers journaux binaires purgés avant d'avoir été réutilisés sur le réplica en lecture. Cette purge entraîne l'arrêt de la réplication et vous ne pouvez plus appeler la commande `mysql.rds_skip_repl_error` pour ignorer les erreurs de réplication.

Vous pouvez atténuer ce problème en augmentant le nombre d'heures pendant lequel les fichiers journaux binaires sont conservés sur votre instance de base de données source. Une fois que vous avez augmenté le temps de rétention de journaux binaires, vous pouvez redémarrer la réplication et appeler la commande `mysql.rds_skip_repl_error` en fonction des besoins.

Pour définir la période de rétention des journaux binaires, utilisez la procédure mysql.rds_set_configuration et spécifiez un paramètre de configuration `'binlog retention hours'`, ainsi que le nombre d'heures pendant lequel conserver les fichiers journaux binaires sur le cluster de bases de données. L'exemple suivant définit la période de rétention des fichiers journaux binaires à 48 heures.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

mysql.rds_start_replication

Lance la réplication à partir d'un cluster de bases de données Aurora MySQL.

> **Note**
>
> Vous pouvez utiliser la procédure stockée mysql.rds_start_replication_until (Aurora MySQL version 3) ou mysql.rds_start_replication_until_gtid (Aurora MySQL version 3) pour lancer la réplication à partir d'une instance de base de données Aurora MySQL et arrêter la réplication à la position spécifiée dans le fichier journal binaire.

Syntaxe

```
CALL mysql.rds_start_replication;
```

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_start_replication`.

Pour importer des données à partir d'une instance de MySQL externe à Amazon RDS, appelez `mysql.rds_start_replication` sur le réplica en lecture pour démarrer le processus de réplication après avoir appelé mysql.rds_set_external_master (Aurora MySQL version 2)ou mysql.rds_set_external_source (Aurora MySQL version 3) pour créer la configuration de réplication. Pour plus d'informations, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

Pour exporter des données vers une instance de MySQL extérieure à Amazon RDS, appelez `mysql.rds_start_replication` et `mysql.rds_stop_replication` sur le réplica en lecture pour contrôler certaines actions de réplication, telles que la purge des journaux binaires. Pour plus d'informations, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

Vous pouvez aussi appeler `mysql.rds_start_replication` sur le réplica en lecture pour redémarrer un processus de réplication que vous avez précédemment arrêté en appelant `mysql.rds_stop_replication`. Pour plus d'informations, consultez Erreur d'arrêt de réplication.

mysql.rds_start_replication_until (Aurora MySQL version 3)

Lance la réplication à partir d'un cluster de bases de données Aurora MySQL et arrête la réplication à la position spécifiée dans le fichier journal binaire.

Syntaxe

```
CALL mysql.rds_start_replication_until (
replication_log_file
  , replication_stop_point
);
```

Paramètres

replication_log_file

Nom du journal binaire sur l'instance de base de données source qui contient les informations de réplication.

replication_stop_point

Position dans le journal binaire `replication_log_file` à laquelle la réplication s'arrêtera.

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_start_replication_until`.

Cette procédure est prise en charge pour Aurora MySQL versions 3.04 et ultérieures.

La procédure `mysql.rds_start_replication_until` stockée n'est pas prise en charge pour la réplication gérée, qui inclut les éléments suivants :

- Réplication de clusters de bases de données Amazon Aurora MySQL dans différentes Régions AWS
- Migration des données d'une instance de base de données RDS for MySQL vers un cluster de bases de données Amazon Aurora MySQL à l'aide d'un réplica en lecture Aurora

Le nom de fichier spécifié pour le paramètre `replication_log_file` doit correspondre au nom du fichier binlog de l'instance de base de données source.

Lorsque le paramètre `replication_stop_point` spécifie une position d'arrêt survenant dans le passé, la réplication est arrêtée immédiatement.

Exemples

L'exemple suivant lance la réplication et réplique les modifications jusqu'à ce qu'il atteigne la position `120` dans le fichier journal binaire `mysql-bin-changelog.000777`.

```
call mysql.rds_start_replication_until(
  'mysql-bin-changelog.000777',
  120);
```

mysql.rds_stop_replication

Arrête la réplication à partir d'une instance de base de données MySQL.

Syntaxe

```
CALL mysql.rds_stop_replication;
```

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_stop_replication`.

Si vous configurez la réplication pour importer des données à partir d'une instance de MySQL s'exécutant à l'extérieur d'Amazon RDS, vous appelez `mysql.rds_stop_replication` sur le réplica en lecture pour arrêter le processus de réplication après que l'importation soit terminée. Pour plus d'informations, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

Si vous configurez la réplication pour exporter les données vers une instance de MySQL extérieure à Amazon RDS, vous appelez `mysql.rds_start_replication` et `mysql.rds_stop_replication` sur le réplica en lecture pour contrôler certaines actions de réplication, telles que la purge des journaux binaires. Pour plus d'informations, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires).

La procédure `mysql.rds_stop_replication` stockée n'est pas prise en charge pour la réplication gérée, qui inclut les éléments suivants :

- Réplication de clusters de bases de données Amazon Aurora MySQL dans différentes Régions AWS
- Migration des données d’une instance de base de données RDS for MySQL vers un cluster de bases de données Amazon Aurora MySQL à l’aide d’un réplica en lecture Aurora

Mettre fin à une session ou à une requête

Les procédures stockées suivantes mettent fin à une session ou à une requête.

Rubriques

- mysql.rds_kill
- mysql.rds_kill_query

mysql.rds_kill

Termine une connexion au serveur MySQL.

Syntaxe

```
CALL mysql.rds_kill(processID);
```

Paramètres

processID

Identité du thread de connexion à terminer.

Notes d'utilisation

Chaque connexion au serveur MySQL s'exécute dans un thread distinct. Pour terminer une connexion, utilisez la procédure `mysql.rds_kill` et transmettez-lui l'ID de thread de cette connexion. Pour obtenir l'ID de thread, utilisez la commande MySQL SHOW PROCESSLIST.

Exemples

L'exemple suivant termine une connexion avec l'ID de thread 4243 :

```
CALL mysql.rds_kill(4243);
```

mysql.rds_kill_query

Termine une requête s'exécutant sur le serveur MySQL.

Syntaxe

```
CALL mysql.rds_kill_query(processID);
```

Paramètres

processID

Identité du processus ou du thread qui exécute la requête à terminer.

Notes d'utilisation

Pour arrêter une requête en cours d'exécution sur le serveur MySQL, utilisez la procédure `mysql_rds_kill_query` et transmettez l'ID de connexion du thread qui exécute la requête. La procédure met alors fin à la connexion.

Pour obtenir l'ID, interrogez la table MySQL INFORMATION_SCHEMA PROCESSLIST ou utilisez la commande MySQL SHOW PROCESSLIST. La valeur figurant dans la colonne ID de `SHOW PROCESSLIST` ou `SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST` est le *processID*.

Exemples

L'exemple suivant arrête une requête dont l'ID de thread de requête est 230040 :

```
CALL mysql.rds_kill_query(230040);
```

Réplication des transactions à l'aide des GTID

Les procédures stockées suivantes contrôlent la réplication des transactions à l'aide des identifiants de transaction globaux (GTID) avec Aurora MySQL. Pour apprendre à utiliser la réplication basée sur les GTID avec Aurora MySQL, consultez Utilisation de la réplication basée sur des identifiants de transaction globaux (GTID).

Rubriques

- mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL version 3)
- mysql.rds_gtid_purged (Aurora MySQL version 3)
- mysql.rds_skip_transaction_with_gtid (Aurora MySQL versions 2 et 3)
- mysql.rds_start_replication_until_gtid (Aurora MySQL version 3)

mysql.rds_assign_gtids_to_anonymous_transactions (Aurora MySQL version 3)

Configure l'option `ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS` de l'instruction `CHANGE REPLICATION SOURCE TO`. Elle force le canal de réplication à attribuer un GTID à des transactions répliquées qui n'en possèdent pas. Vous pouvez ainsi effectuer une réplication de journaux binaires à partir d'une source qui n'utilise pas la réplication GTID vers un réplica qui l'utilise. Pour plus d'informations, consultez Instruction CHANGE REPLICATION SOURCE TO et Réplication depuis une source sans GTID vers un réplica avec GTID dans le Manuel de référence MySQL.

Syntaxe

```
CALL mysql.rds_assign_gtids_to_anonymous_transactions(gtid_option);
```

Paramètres

`gtid_option`

Valeur de chaîne. Les valeurs autorisées sont : `OFF`, `LOCAL` ou un UUID spécifié.

Notes d'utilisation

Cette procédure a le même effet que l'émission de l'instruction `CHANGE REPLICATION SOURCE TO ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS = gtid_option` dans Community MySQL.

Le GTID doit être orienté vers `ON` pour *`gtid_option`* pour être défini sur `LOCAL` ou un UUID spécifique.

La valeur par défaut est `OFF`, ce qui signifie que la fonctionnalité n'est pas utilisée.

`LOCAL` attribue un GTID incluant le propre UUID du réplica (paramètre `server_uuid`).

La transmission d'un paramètre correspondant à un UUID attribue un GTID qui inclut l'UUID spécifié, tel que le paramètre `server_uuid` pour le serveur source de réplication.

Exemples

Pour désactiver cette fonction :

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('OFF');
+-------------------------------------------------------------+
| Message  |
+-------------------------------------------------------------+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: OFF |
+-------------------------------------------------------------+
1 row in set (0.07 sec)
```

Pour utiliser l'UUID du réplica :

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('LOCAL');
+------------------------------------------------------------+
| Message  |
+------------------------------------------------------------+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: LOCAL |
+------------------------------------------------------------+
1 row in set (0.07 sec)
```

Pour utiliser un UUID spécifié :

```
mysql> call mysql.rds_assign_gtids_to_anonymous_transactions('317a4760-
f3dd-3b74-8e45-0615ed29de0e');
+-----------------------------------------------------------------------------------------------
+
| Message |
+-----------------------------------------------------------------------------------------------
+
| ASSIGN_GTIDS_TO_ANONYMOUS_TRANSACTIONS has been set to: 317a4760-
f3dd-3b74-8e45-0615ed29de0e |
+-----------------------------------------------------------------------------------------------
+
```

```
1 row in set (0.07 sec)
```

mysql.rds_gtid_purged (Aurora MySQL version 3)

Définit la valeur globale de la variable système `gtid_purged` sur un ensemble d'identifiants de transaction globaux (GTID) donné. La variable système `gtid_purged` est un ensemble d'identifiants GTID composé des GTID de toutes les transactions qui ont été validées sur le serveur, mais qui n'existent dans aucun fichier journal binaire du serveur.

Pour permettre la compatibilité avec MySQL 8.0, il existe deux manières de définir la valeur de `gtid_purged` :

- Remplacez la valeur de `gtid_purged` par l'ensemble d'identifiants GTID que vous avez spécifié.
- Ajoutez l'ensemble GTID que vous avez spécifié à l'ensemble d'identifiants GTID que `gtid_purged` contient déjà.

Syntaxe

Pour remplacer la valeur de `gtid_purged` par l'ensemble d'identifiants GTID que vous avez spécifié :

```
CALL mysql.rds_gtid_purged (gtid_set);
```

Pour ajouter la valeur de `gtid_purged` à votre ensemble d'identifiants GTID que vous avez spécifié :

```
CALL mysql.rds_gtid_purged (+gtid_set);
```

Paramètres

`gtid_set`

La valeur de *`gtid_set`* doit être un sur-ensemble de la valeur actuelle de `gtid_purged`, et ne doit pas se croiser avec`gtid_subtract(gtid_executed,gtid_purged)`. C'est-à-dire que le nouvel ensemble d'identifiants GTID doit inclure tous les GTID qui étaient déjà présents dans `gtid_purged`, et ne peut inclure aucun des GTID figurant dans `gtid_executed`, qui n'ont pas encore été purgés. Le paramètre *`gtid_set`* ne peut pas non plus inclure les GTID qui

se trouvent dans l'ensemble `gtid_owned` global, les GTID pour les transactions en cours de traitement sur le serveur.

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_gtid_purged`.

Cette procédure est prise en charge pour Aurora MySQL versions 3.04 et ultérieures.

Exemples

L'exemple suivant attribue le GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23` à la variable globale `gtid_purged`.

```
CALL mysql.rds_gtid_purged('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_skip_transaction_with_gtid (Aurora MySQL versions 2 et 3)

Ignore la réplication d'une transaction avec l'identifiant de transaction global (GTID) spécifié sur une instance principale Aurora.

Vous pouvez utiliser cette procédure pour la reprise après sinistre lorsqu'il est avéré qu'une transaction GTID entraîne des problèmes. Utilisez cette procédure stockée pour ignorer la transaction problématique. Les transactions problématiques sont par exemple celles qui désactivent la réplication, suppriment des données importantes ou entraînent l'indisponibilité de l'instance de base de données.

Syntaxe

```
CALL mysql.rds_skip_transaction_with_gtid (
gtid_to_skip
);
```

Paramètres

`gtid_to_skip`

GTID de la transaction de réplication à ignorer.

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_skip_transaction_with_gtid`.

Cette procédure est prise en charge pour Aurora MySQL versions 2 et 3.

Exemples

L'exemple suivant ignore la réplication de la transaction avec le GTID `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`.

```
CALL mysql.rds_skip_transaction_with_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

mysql.rds_start_replication_until_gtid (Aurora MySQL version 3)

Lance la réplication à partir d'un cluster de bases de données Aurora MySQL et arrête la réplication immédiatement après l'identifiant de transaction global (GTID) spécifié.

Syntaxe

```
CALL mysql.rds_start_replication_until_gtid(gtid);
```

Paramètres

gtid

Identifiant de transaction global (GTID) après lequel la réplication s'arrête.

Notes d'utilisation

L'utilisateur principal doit exécuter la procédure `mysql.rds_start_replication_until_gtid`.

Cette procédure est prise en charge pour Aurora MySQL versions 3.04 et ultérieures.

La procédure `mysql.rds_start_replication_until_gtid` stockée n'est pas prise en charge pour la réplication gérée, qui inclut les éléments suivants :

- Réplication de clusters de bases de données Amazon Aurora MySQL dans différentes Régions AWS

- Migration des données d’une instance de base de données RDS for MySQL vers un cluster de bases de données Amazon Aurora MySQL à l’aide d’un réplica en lecture Aurora

Lorsque le paramètre `gtid` spécifie une transaction ayant déjà été exécutée par le réplica, la réplication est immédiatement arrêtée.

Exemples

L’exemple suivant lance la réplication et réplique les modifications jusqu’à ce que le GTID soit atteint `3E11FA47-71CA-11E1-9E33-C80AA9429562:23`.

```
call mysql.rds_start_replication_until_gtid('3E11FA47-71CA-11E1-9E33-C80AA9429562:23');
```

Rotation des journaux de requêtes

Les procédures stockées suivantes effectuent la rotation des journaux MySQL vers des tables de sauvegarde. Pour plus d'informations, consultez Fichiers journaux de base de données Aurora MySQL.

Rubriques

- mysql.rds_rotate_general_log
- mysql.rds_rotate_slow_log

mysql.rds_rotate_general_log

Convertit la table `mysql.general_log` en table de sauvegarde.

Syntaxe

```
CALL mysql.rds_rotate_general_log;
```

Notes d'utilisation

Vous pouvez convertir la table `mysql.general_log` en table de sauvegarde en appelant la procédure `mysql.rds_rotate_general_log`. Lors de la rotation des tables de journaux, la table de journal actuelle est copiée vers une table de journal de sauvegarde et les entrées de la table de journal actuelle sont supprimées. Si la table du journal de sauvegarde existe déjà, elle est supprimée avant que la table du journal active ne soit copiée dans la sauvegarde. Si besoin, vous pouvez interroger la table de journal de sauvegarde. La table de journal de sauvegarde de la table `mysql.general_log` est nommée `mysql.general_log_backup`.

Vous ne pouvez exécuter cette procédure que lorsque le paramètre `log_output` est défini sur `TABLE`.

mysql.rds_rotate_slow_log

Convertit la table `mysql.slow_log` en table de sauvegarde.

Syntaxe

```
CALL mysql.rds_rotate_slow_log;
```

Notes d’utilisation

Vous pouvez convertir la table `mysql.slow_log` en table de sauvegarde en appelant la procédure `mysql.rds_rotate_slow_log`. Lors de la rotation des tables de journaux, la table de journal actuelle est copiée vers une table de journal de sauvegarde et les entrées de la table de journal actuelle sont supprimées. Si la table du journal de sauvegarde existe déjà, elle est supprimée avant que la table du journal active ne soit copiée dans la sauvegarde.

Si besoin, vous pouvez interroger la table de journal de sauvegarde. La table de journal de sauvegarde de la table `mysql.slow_log` est nommée `mysql.slow_log_backup`.

Configuration et affichage de la configuration du journal binaire

Les procédures stockées suivantes définissent et affichent les paramètres de configuration, tels que la conservation des fichiers journaux binaires.

Rubriques

- mysql.rds_set_configuration
- mysql.rds_show_configuration

mysql.rds_set_configuration

Spécifie le nombre d'heures pendant lequel les journaux binaires doivent être conservés ou le nombre de secondes pendant lequel retarder la réplication.

Syntaxe

```
CALL mysql.rds_set_configuration(name,value);
```

Paramètres

nom

Nom du paramètre de configuration à définir.

valeur

Valeur du paramètre de configuration.

Notes d'utilisation

La procédure `mysql.rds_set_configuration` prend en charge des paramètres de configuration suivants :

- nombre d'heures de conservation du journal binaire

Les paramètres de configuration sont stockés de manière permanente et survivent à tout redémarrage ou basculement d'une instance de base de données.

nombre d’heures de conservation du journal binaire

Le paramètre `binlog retention hours` est utilisé pour spécifier le nombre d’heures de rétention des fichiers journaux binaires. Amazon Aurora purge normalement un journal binaire dès que possible, mais il se peut que le journal binaire soit encore requis pour la réplication avec une base de données MySQL extérieure à Aurora.

La valeur par défaut de `binlog retention hours` est `NULL`. Pour Aurora MySQL, `NULL` signifie que les journaux binaires sont nettoyés lentement. Les journaux binaires Aurora MySQL peuvent rester dans le système pendant un certain temps, qui ne dépasse généralement pas un jour.

Pour spécifier le nombre d’heures pendant lesquelles conserver les journaux binaires sur un cluster de bases de données, utilisez la procédure stockée `mysql.rds_set_configuration` et spécifiez une période suffisamment longue pour que la réplication se produise, comme illustré dans l’exemple suivant.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

> **Note**
>
> Vous ne pouvez pas utiliser la valeur `0` pour `binlog retention hours`.

Pour des clusters de bases de données Aurora MySQL versions 2.11.0 et ultérieures et version 3, la valeur `binlog retention hours` maximale est 2 160 (90 jours).

Après avoir défini la période de rétention, surveillez l’utilisation du stockage de l’instance de base de données afin de garantir que les journaux binaires conservés n’utilisent pas un espace de stockage trop grand.

mysql.rds_show_configuration

Nombre d’heures pendant lequel les journaux binaires sont conservés.

Syntaxe

```
CALL mysql.rds_show_configuration;
```

Notes d'utilisation

Pour vérifier le nombre d'heures pendant lequel Amazon RDS conserve les journaux binaires, utilisez la procédure stockée `mysql.rds_show_configuration`.

Exemples

L'exemple suivant affiche la période de rétention :

```
call mysql.rds_show_configuration;
                name                          value     description
                binlog retention hours        24        binlog retention hours specifies
 the duration in hours before binary logs are automatically deleted.
```

Tables information_schema spécifiques à Aurora MySQL

Aurora MySQL possède certaines tables `information_schema` spécifiques à Aurora.

information_schema.aurora_global_db_instance_status

La table `information_schema.aurora_global_db_instance_status` contient des informations sur l'état de toutes les instances de base de données dans les clusters de bases de données principal et secondaire d'une base de données globale. Les colonnes que vous pouvez utiliser sont indiquées dans le tableau suivant. Les colonnes restantes sont destinées à un usage interne d'Aurora uniquement.

> **Note**
>
> Cette table de schéma d'informations n'est disponible qu'avec les bases de données globales Aurora MySQL 3.04.0 et versions ultérieures.

Colonne	Type de données	Description
SERVER_ID	varchar(100)	Identifiant de l'instance DB.
SESSION_ID	varchar(100)	Identifiant unique de la session en cours. La valeur `MASTER_SESSION_ID` identifie l'instance de base

Colonne	Type de données	Description
		de données d’enregistreur (principale).
AWS_REGION	varchar(100)	La Région AWS dans laquelle cette instance de base de données globale s’exécute . Pour obtenir la liste des régions, consultez Disponibilité dans les Régions.
DURABLE_LSN	bigint unsigned	Numéro de séquence de journal (LSN) rendu durable dans le stockage. Un numéro de séquence de journal (LSN) est un numéro séquentiel unique qui identifie un enregistrement dans le journal des transactions de la base de données. Les LSN sont classés de telle sorte qu’un LSN plus grand représente une transaction ultérieure.
HIGHEST_LSN_RCVD	bigint unsigned	LSN le plus élevé reçu par l’instance de base de données en provenance de l’instance de base de données d’enregistreur.
OLDEST_READ_VIEW_TRX_ID	bigint unsigned	ID de la transaction la plus ancienne vers laquelle l’instance de base de données d’enregistreur peut effectuer une purge.

Colonne	Type de données	Description
OLDEST_READ_VIEW_LSN	bigint unsigned	LSN le plus ancien utilisé par l'instance de base de données pour lire à partir du stockage.
VISIBILITY_LAG_IN_MSEC	float(10,0) unsigned	Pour les lecteurs dans le cluster de bases de données principal, retard accumulé par cette instance de base de données par rapport à l'instance de base de données d'enregistreur en millisecondes. Pour les lecteurs dans un cluster de bases de données secondaire, retard accumulé par cette instance de base de données par rapport au volume secondaire en millisecondes.

information_schema.aurora_global_db_status

La table `information_schema.aurora_global_db_status` contient des informations sur divers aspects du retard de la base de données globale Aurora, en particulier le retard du stockage Aurora sous-jacent (appelé « retard de durabilité ») et le retard entre l'objectif de point de reprise (RPO). Les colonnes que vous pouvez utiliser sont indiquées dans le tableau suivant. Les colonnes restantes sont destinées à un usage interne d'Aurora uniquement.

> **Note**
>
> Cette table de schéma d'informations n'est disponible qu'avec les bases de données globales Aurora MySQL 3.04.0 et versions ultérieures.

Colonne	Type de données	Description
AWS_REGION	varchar(100)	La Région AWS dans laquelle cette instance de base de données globale s'exécute . Pour obtenir la liste des régions, consultez Disponibilité dans les Régions.
HIGHEST_LSN_WRITTEN	bigint unsigned	Numéro de séquence de journal (LSN) le plus élevé qui existe actuellement sur ce cluster de bases de données. Un numéro de séquence de journal (LSN) est un numéro séquentiel unique qui identifie un enregistrement dans le journal des transactions de la base de données. Les LSN sont classés de telle sorte qu'un LSN plus grand représente une transaction ultérieure.
DURABILITY_LAG_IN_MILLISECONDS	float(10,0) unsigned	Différence dans les valeurs d'horodatage entre `HIGHEST_LSN_WRITTEN` sur un cluster de bases de données secondaire et `HIGHEST_LSN_WRITTEN` sur le cluster de bases de données principal . Cette valeur est toujours égale à 0 sur le cluster de bases de données principal de

Colonne	Type de données	Description
		la base de données globale Aurora.
RPO_LAG_IN_MILLISECONDS	float(10,0) unsigned	Retard de l’objectif de point de reprise (RPO). Le retard RPO correspond au temps nécessaire au stockage de la transaction utilisateur COMMIT la plus récente sur un cluster de bases de données secondaire, après qu’elle a été stockée sur le cluster de bases de données principal de la base de données globale Aurora. Cette valeur est toujours égale à 0 sur le cluster de bases de données principal de la base de données globale Aurora. En termes simples, cette métrique calcule l’objectif de point de reprise pour chaque cluster de bases de données Aurora MySQL dans la base de données globale Aurora, c’est-à-dire la quantité de données qui risque d’être perdue en cas de panne. Comme pour la latence, le RPO est mesuré dans le temps.

Colonne	Type de données	Description
LAST_LAG_CALCULATION_TIMESTAMP	datetime	Horodatage qui spécifie l'heure à laquelle les valeurs ont été calculées pour la dernière fois pour `DURABILITY_LAG_IN_MILLISECONDS` et `RPO_LAG_IN_MILLISECONDS`. Une valeur temporelle telle que `1970-01-01 00:00:00+00` signifie qu'il s'agit du cluster de bases de données principal.
OLDEST_READ_VIEW_TRX_ID	bigint unsigned	ID de la transaction la plus ancienne vers laquelle l'instance de base de données d'enregistreur peut effectuer une purge.

information_schema.replica_host_status

La table `information_schema.replica_host_status` contient des informations de réplication. Les colonnes que vous pouvez utiliser sont indiquées dans la table suivante. Les colonnes restantes sont destinées à un usage interne d'Aurora uniquement.

Colonne	Type de données	Description
CPU	double	Pourcentage d'utilisation du processeur par l'hôte de réplication.
IS_CURRENT	tinyint	Si la réplique est à jour.

Colonne	Type de données	Description
LAST_UPDATE_TIMESTAMP	datetime(6)	Heure de la dernière mise à jour. Utilisé pour déterminer si un enregistrement est périmé.
REPLICA_LAG_IN_MILLISECONDS	double	Le retard de réplica en millisecondes.
SERVER_ID	varchar(100)	ID du serveur de base de données.
SESSION_ID	varchar(100)	ID de session de la base de données. Utilisé pour déterminer si une instance de base de données est une instance d'enregistreur ou de lecture.

> **Note**
>
> Lorsqu'une instance de réplica prend du retard, les informations demandées dans sa table `information_schema.replica_host_status` peuvent être obsolètes. Dans ce cas, nous vous recommandons plutôt d'effectuer une requête à partir de l'instance d'enregistreur. La table `mysql.ro_replica_status` contient des informations similaires, mais nous vous déconseillons de l'utiliser.

information_schema.aurora_forwarding_processlist

La table `information_schema.aurora_forwarding_processlist` contient des informations sur les processus impliqués dans le transfert d'écriture.

Le contenu de cette table est visible uniquement sur l'instance de base de données d'enregistreur pour un cluster de bases de données sur lequel le transfert d'écriture global ou intracluster est activé. Un jeu de résultats vide est renvoyé sur les instances de base de données de lecteur.

Champ	Type de données	Description
ID	bigint	L'identifiant de la connexion sur l'instance de base de données d'enregistreur. Cet identifiant est la même valeur que celle affichée dans la colonne `Id` de l'instruction `SHOW PROCESSLIST` et renvoyée par la fonction `CONNECTION_ID()` dans le thread.
USER	varchar(32)	Utilisateur MySQL qui a émis l'instruction.
HOST	varchar(255)	Client MySQL qui a émis l'instruction. Pour les instructions transférées, ce champ indique l'adresse hôte du client d'application qui a établi la connexion sur l'instance de base de données du lecteur de transfert.
BdD	varchar(64)	Base de données par défaut pour le thread.
COMMAND	varchar(16)	Le type de commande que le thread exécute pour le compte du client, ou `Sleep` si la session est inactive. Pour une description des commandes de thread, consultez la documentation MySQL sur Valeurs de Command de thread (langue française non garantie) dans la documentation MySQL.
TIME	int	Durée en secondes pendant laquelle le thread est resté dans son état actuel.
STATE	varchar(64)	Action, événement ou état qui indique ce que fait le thread. Pour une description des valeurs d'état, consultez États de thread généraux (langue française non garantie) dans la documentation MySQL.
INFO	longtext	Instruction que le thread exécute, ou `NULL` s'il n'exécute pas d'instruction. L'instruction peut être celle qui est envoyée

Champ	Type de données	Description
		au serveur ou une instruction interne si l'instruction exécute d'autres instructions.
IS_FORWARDED	bigint	Indique si le thread est transféré depuis une instance de base de données de lecteur.
REPLICA_SESSION_ID	bigint	Identifiant de connexion sur le réplica Aurora. Cet identifiant est la même valeur que celle affichée dans la colonne `Id` de l'instruction `SHOW PROCESSLIST` sur l'instance de base de données du lecteur Aurora de transfert.
REPLICA_INSTANCE_IDENTIFIER	varchar(64)	Identifiant de l'instance de base de données du thread de transfert.
REPLICA_CLUSTER_NAME	varchar(64)	Identifiant du cluster de bases de données du thread de transfert. Pour le transfert d'écriture intracluster, cet identifiant est le même pour le cluster de bases de données et pour l'instance de base de données d'enregistreur.
REPLICA_REGION	varchar(64)	La Région AWS d'où provient le thread de transfert. Pour le transfert d'écriture intracluster, cette région est la même Région AWS que pour l'instance de base de données d'enregistreur.

Mises à jour du moteur de base de données pour Amazon Aurora MySQL

Amazon Aurora publie régulièrement des mises à jour. Ces mises à jour sont appliquées aux clusters de bases de données Aurora au cours des fenêtres de maintenance du système. L'horaire d'application des mises à jour dépend de la région et du paramètre de fenêtre de maintenance configuré pour le cluster de bases de données, ainsi que du type de mise à jour.

Les versions d'Amazon Aurora sont mises à la disposition de toutes les régions AWS pendant plusieurs jours. Certaines régions peuvent afficher temporairement une version du moteur qui n'est pas encore disponible dans une autre région.

Les mises à jour sont appliquées simultanément à toutes les instances d'un cluster de bases de données. Une mise à jour nécessite un redémarrage de la base de données sur toutes les instances d'un cluster de bases de données, si bien que vous connaîtrez 20 à 30 secondes d'indisponibilité, après quoi vous pourrez reprendre l'utilisation de votre ou de vos clusters de bases de données. Vous pouvez consulter ou modifier vos paramètres de créneau de maintenance dans AWS Management Console.

Pour plus de détails sur les versions d'Aurora MySQL prises en charge par Amazon Aurora, consultez Notes de mise à jour d'Aurora MySQL.

Ensuite, vous pouvez apprendre à choisir la bonne version d'Aurora MySQL pour votre cluster, à spécifier la version lorsque vous créez ou mettez à niveau un cluster, mais aussi à utiliser les procédures pour mettre à niveau un cluster d'une version à une autre avec une interruption minimale.

Rubriques

Vérification des numéros de version d'Aurora MySQL

Bien qu'Aurora MySQL Éditiond compatible soit compatible avec les moteurs de base de données MySQL, Aurora MySQL inclut des fonctions et des corrections de bogue spécifiques à des versions Aurora MySQL particulières. Les développeurs d'applications peuvent vérifier la version Aurora MySQL dans leurs applications à l'aide de SQL. Les administrateurs de base de données peuvent vérifier et spécifier des versions Aurora MySQL lors de la création ou de la mise à niveau de clusters de bases de données et d'instances de base de données Aurora MySQL

Rubriques

Vérification ou spécification de versions du moteur Aurora MySQL via AWS

Lorsque vous effectuez des tâches administratives à l'aide de l'AWS Management Console, de l'AWS CLI ou de l'API RDS, vous spécifiez la version d'Aurora MySQL dans un format alphanumérique descriptif.

À partir d'Aurora MySQL version 2, les versions du moteur Aurora ont la syntaxe suivante.

```
mysql-major-version.mysql_aurora.aurora-mysql-version
```

La valeur de la partie *mysql-major-version-* est `5.7` ou `8.0`. Cette valeur représente la version du protocole client et le niveau général de prise en charge des fonctions MySQL pour la version Aurora MySQL correspondante.

La partie *aurora-mysql-version* est une valeur en trois parties séparées par des points : la version Aurora MySQL majeure, la version Aurora MySQL mineure et le niveau de correctif. La version majeure est 2 ou 3. Ces valeurs représentent des versions d'Aurora MySQL compatibles avec MySQL 5.7 ou 8.0, respectivement. La version mineure représente la version de la fonction dans la série 2.x ou 3.x. Le niveau de correctif commence à `0` pour chaque version mineure et représente l'ensemble des corrections de bogue suivantes s'appliquant à la version mineure. Parfois, une nouvelle fonction est intégrée à une version mineure sans être rendue immédiatement visible. Dans ces cas, la fonction fait l'objet d'un paramétrage précis et est rendue publique dans un niveau de correctif ultérieur.

Toutes les versions du moteur Aurora MySQL 2.x sont compatibles avec Community MySQL 5.7.12 ou version ultérieure. Toutes les versions du moteur Aurora MySQL 3.x sont compatibles avec MySQL 8.0.23 ou version ultérieure. Vous pouvez vous référer aux notes de mise à jour de la version 3.x spécifique pour trouver la version compatible MySQL correspondante.

Par exemple, les versions du moteur pour Aurora MySQL 3.04.0 et 2.11.2 sont les suivantes.

```
8.0.mysql_aurora.3.04.0
5.7.mysql_aurora.2.11.2
```

> **Note**
>
> Il n'y a pas de correspondance individuelle entre les versions communautaires de MySQL et les versions Aurora MySQL 2.x. Pour Aurora MySQL version 3, il existe un mappage plus direct. Pour vérifier les corrections de bogues et les nouvelles fonctionnalités d'une version particulière d'Aurora MySQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 3, et Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 2 dans Notes de mise à jour d'Aurora MySQL. Pour obtenir une liste chronologique des nouvelles fonctions et versions, consultez Historique du document. Pour vérifier la version minimale requise pour un correctif lié à la sécurité, consultez Vulnérabilités de sécurité corrigées dans Aurora MySQL dans Notes de mise à jour d'Aurora MySQL.

Vous spécifiez la version du moteur Aurora MySQL dans certaines commandes de l'AWS CLI et opérations de l'API RDS. Par exemple, vous spécifiez l'option `--engine-version` lorsque vous exécutez les commandes create-db-cluster et modify-db-cluster de l'AWS CLI. Vous spécifiez le paramètre `EngineVersion` lorsque vous exécutez les opérations d'API RDS CreateDBCluster et ModifyDBCluster.

Dans Aurora MySQL versions 2 et ultérieures, la version du moteur dans la AWS Management Console inclut également la version d'Aurora. La mise à niveau du cluster modifie la valeur affichée. Cette modification vous permet de spécifier et de vérifier les versions Aurora MySQL précises, sans avoir besoin de vous connecter au cluster ou d'exécuter des commandes SQL.

> **Tip**
>
> Pour les clusters Aurora gérés via CloudFormation, cette modification du paramètre `EngineVersion` peut déclencher des actions de CloudFormation. Pour découvrir comment CloudFormation traite les modifications du paramètre `EngineVersion`, consultez la documentation CloudFormation.

Vérification des versions Aurora MySQL avec SQL

Les numéros de version Aurora que vous pouvez récupérer dans votre application à l'aide de requêtes SQL utilisent le format *`<major version>`*`.`*`<minor version>`*`.`*`<patch version>`*. Vous pouvez obtenir ce numéro de version pour n'importe quelle instance de base de données de votre cluster Aurora MySQL en interrogeant la variable système `AURORA_VERSION`. Pour obtenir ce numéro de version, utilisez une des requêtes suivantes.

```
select aurora_version();
select @@aurora_version;
```

Ces requêtes produisent une sortie similaire à la suivante.

```
mysql> select aurora_version(), @@aurora_version;
+------------------+------------------+
| aurora_version() | @@aurora_version |
+------------------+------------------+
| 3.05.2           | 3.05.2           |
+------------------+------------------+
```

Les numéros de version renvoyés par la console, la CLI ou l'API RDS à l'aide des techniques décrites dans Vérification ou spécification de versions du moteur Aurora MySQL via AWS sont généralement plus détaillés.

Versions à long terme (LTS) et versions bêta d'Amazon Aurora MySQL

Aurora MySQL fournit des versions à long terme (LTS) et des versions bêta pour certaines versions du moteur Aurora MySQL.

Rubriques

Versions à long terme (LTS) d'Aurora MySQL

Chaque nouvelle version d'Aurora MySQL reste disponible pendant un certain temps afin que vous puissiez l'utiliser pour créer ou mettre à niveau un cluster de bases de données. Une fois cette période écoulée, vous devez mettre à niveau tout cluster utilisant cette version. Vous pouvez mettre à niveau votre cluster manuellement avant la fin de la période de prise en charge. Aurora peut également effectuer sa mise à niveau automatiquement une fois que sa version d'Aurora MySQL n'est plus prise en charge.

Aurora désigne certaines versions d'Aurora MySQL comme étant des versions « long-term support (LTS) », ou versions à long terme. Les clusters de bases de données qui utilisent des versions LTS peuvent conserver la même version plus longtemps et faire l'objet de cycles de mise à niveau moins nombreux que les clusters qui utilisent des versions non-LTS. Les clusters de bases de données qui utilisent des versions LTS peuvent conserver la même version mineure pendant au moins trois ans, ou jusqu'à la fin du support standard pour la version majeure, selon la première éventualité. Lorsqu'un cluster de bases de données disposant d'une version LTS nécessite une mise à niveau, Aurora le met à niveau vers la version LTS suivante. Ainsi, le cluster n'a plus besoin de mise à niveau pendant longtemps.

Pendant toute la durée de vie d'une version LTS d'Aurora MySQL, de nouveaux niveaux de correctifs LTS introduisent des correctifs concernant des problèmes importants. Ces niveaux de correctifs ne comportent aucune nouvelle fonctionnalité. Vous pouvez choisir d'appliquer ou non ces correctifs aux clusters de bases de données qui exécutent la version LTS. Nous recommandons aux clients utilisant des versions LTS de passer au dernier correctif de la version LTS mineure au moins une fois par an afin de bénéficier des correctifs opérationnels et de sécurité de haute gravité. Pour certains correctifs critiques, Amazon peut effectuer une mise à niveau gérée vers un niveau de correctif de la même version LTS. Ces mises à niveau gérées sont exécutées automatiquement au sein de la fenêtre de maintenance du cluster. Toutes les versions d'Aurora MySQL (versions LTS et autres) sont soumises à des tests de stabilité et de fonctionnement approfondis. Certaines versions mineures sont désignées comme des versions LTS afin de permettre aux clients de rester sur ces versions mineures plus longtemps sans passer à une version mineure plus récente.

Nous vous recommandons d'effectuer la mise à niveau vers la dernière version, au lieu d'utiliser la version LTS, pour la plupart de vos clusters Aurora MySQL. Cela vous permet de bénéficier des

avantages d'Aurora en tant que service géré et vous donne accès aux dernières fonctionnalités et aux derniers correctifs de bogues. Les versions LTS sont destinées aux clusters avec les caractéristiques suivantes :

- Vous ne pouvez pas vous permettre d'avoir une durée d'indisponibilité de votre application Aurora MySQL pour les mises à niveau, en dehors des rares cas où des correctifs critiques doivent être installés.
- Le cycle de test du cluster et des applications associées dure très longtemps pour chaque mise à niveau du moteur de base de données Aurora MySQL.
- La version de la base de données de votre cluster Aurora MySQL dispose de toutes les fonctionnalités de moteur de base de données et de tous les correctifs de bogues dont votre application a besoin.

Les versions LTS actuelles d'Aurora MySQL sont les suivantes :

- Aurora MySQL version 3.10.*.
- Aurora MySQL version 3.04.*.

Pour plus d'informations sur cette version LTS, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 3 dans Notes de mise à jour d'Aurora MySQL.

> **Note**
>
> Nous vous recommandons de désactiver les mises à niveau automatiques des versions mineures pour les versions LTS. Définissez le paramètre `AutoMinorVersionUpgrade` sur `false` ou décochez la case Activer la mise à niveau automatique des versions mineures dans la AWS Management Console.
> Si vous ne désactivez pas cette fonctionnalité, votre cluster de bases de données pourra être mis à niveau vers une version non LTS.

Versions bêta d'Aurora MySQL

Une version bêta d'Aurora MySQL est un correctif de sécurité précoce, disponible uniquement dans un nombre limité de. Régions AWS Ces correctifs seront déployés plus largement dans toutes les régions avec la prochaine version de correctifs.

La numérotation d'une version bêta est similaire à celle d'une version mineure d'Aurora MySQL, mais avec un quatrième chiffre supplémentaire, par exemple 2.12.0.1 ou 3.05.0.1.

Pour plus d'informations, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 2 et Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 3 dans Notes de mise à jour d'Aurora MySQL.

Préparation à la fin du support standard de l'Édition compatible d'Amazon Aurora MySQL version 2

La fin du support standard de l'Édition compatible d'Amazon Aurora MySQL version 2 (avec compatibilité MySQL 5.7) est prévue pour le 31 octobre 2024. Nous vous recommandons de mettre à niveau tous les clusters exécutant Aurora MySQL version 2 vers la version par défaut Aurora MySQL version 3 (avec compatibilité MySQL 8.0) ou vers une version ultérieure avant la fin du support standard d'Aurora MySQL version 2. Le 31 octobre 2024, Amazon RDS inscrira automatiquement vos bases de données dans le support étendu Amazon RDS. Si vous exécutez Amazon Aurora MySQL version 2 (avec compatibilité MySQL 5.7) dans un cluster Aurora Serverless version 1, cela ne vous concerne pas. Si vous souhaitez mettre à niveau vos clusters Aurora Serverless version 1 vers Aurora MySQL version 3, consultez Chemin de mise à niveau pour les clusters de bases de données Aurora Serverless v1.

Vous trouverez les dates de fin de support à venir pour les versions majeures d'Aurora MySQL dans le Calendrier des versions majeures d'Aurora MySQL.

Si vous avez des clusters exécutant Aurora MySQL version 2, vous recevrez des notifications périodiques contenant les dernières informations sur la procédure de mise à niveau à mesure que la date de fin du support standard approchera. Nous mettrons régulièrement cette page à jour.

Dates de fin du support standard

1. Jusqu'au 31 octobre 2024 – Vous pouvez à tout moment mettre à niveau des clusters Aurora MySQL version 2 (avec compatibilité MySQL 5.7) vers Aurora MySQL version 3 (avec compatibilité MySQL 8.0).
2. 31 octobre 2024 — À cette date, le support standard de la version 2 d'Aurora MySQL prend fin. Amazon RDS inscrira automatiquement vos clusters dans le support étendu Amazon RDS.

Nous vous inscrirons automatiquement dans le support étendu RDS. Pour plus d'informations, consultez Support étendu Amazon RDS avec Amazon Aurora.

Recherche des clusters affectés par ce processus de fin de vie

Pour trouver les clusters affectés par ce processus de fin de vie, utilisez les procédures suivantes.

> **Important**
>
> Assurez-vous d'exécuter ces instructions dans chaque Région AWS et pour chaque Compte AWS où vos ressources sont situées.

Console

Pour rechercher un cluster Aurora MySQL version 2

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Dans la zone Filtrer par bases de données, saisissez 5.7.
4. Recherchez les occurrences Aurora MySQL dans la colonne du moteur.

AWS CLI

Pour rechercher les clusters affectés par ce processus de fin de vie à l'aide de l'AWS CLI, effectuez un appel à la commande describe-db-clusters. Vous pouvez utiliser l'exemple de script suivant.

Example

```
aws rds describe-db-clusters --include-share --query 'DBClusters[?(Engine==`aurora-
mysql` && contains(EngineVersion,`5.7.mysql_aurora`))].{EngineVersion:EngineVersion,
 DBClusterIdentifier:DBClusterIdentifier, EngineMode:EngineMode}' --output table
 --region us-east-1

        +-------------------------------------------------------------+
        |                     DescribeDBClusters                      |
        +---------------------+---------------+-------------------------+
        |         DBCI        |      EM       |          EV             |
        +---------------------+---------------+-------------------------+
        |    aurora-mysql2    |  provisioned  | 5.7.mysql_aurora.2.11.3 |
        | aurora-serverlessv1 |   serverless  | 5.7.mysql_aurora.2.11.3 |
```

```
+---------------------+---------------+-------------------------+
```

API RDS

Pour trouver les clusters de bases de données Aurora MySQL exécutant Aurora MySQL version 2, utilisez l'opération d'API RDS DescribeDBClusters avec les paramètres requis suivants :

- `DescribeDBClusters`
  - Filters.Filter.N
    - Nom
      - engine
    - Values.Value.N
      - ['aurora']

Support étendu Amazon RDS

Vous pouvez utiliser le support étendu Amazon RDS sur la version communautaire MySQL 5.7 gratuitement jusqu'à la date de fin du support, le 31 octobre 2024. Le 31 octobre 2024, Amazon RDS inscrira automatiquement vos bases de données dans le support étendu RDS pour Aurora MySQL version 2. Le support étendu RDS pour Aurora est un service payant offrant 28 mois supplémentaires de support pour Aurora MySQL version 2, soit jusqu'en février 2027. Le support étendu RDS ne sera proposé que pour les versions mineures 2.11 et 2.12 d'Aurora MySQL. Si vous prévoyez d'utiliser Amazon Aurora MySQL version 2 après la fin du support standard, prévoyez d'exécuter vos bases de données sur l'une de ces versions mineures avant le 31 octobre 2024.

Pour plus d'informations sur le support étendu RDS, telles que les frais et d'autres éléments à prendre en compte, consultez Support étendu Amazon RDS avec Amazon Aurora.

Réalisation d'une mise à niveau

La mise à niveau entre les versions majeures nécessite une planification et des tests plus étendus que pour une version mineure. Le processus peut prendre beaucoup de temps. Nous considérons la mise à niveau comme un processus en trois étapes, comprenant des activités avant, pendant et après la mise à niveau.

Avant la mise à niveau :

Avant la mise à niveau, nous vous recommandons de vérifier la compatibilité des applications, les performances, les procédures de maintenance et autres considérations similaires pour le cluster mis à niveau, afin de vous assurer que vos applications fonctionneront comme prévu après la mise à niveau. Voici cinq recommandations qui vous permettront de bénéficier d'une meilleure expérience de mise à niveau.

- Tout d'abord, il est essentiel de comprendre le Fonctionnement de la mise à niveau sur place d'une version majeure de Aurora MySQL.
- Ensuite, explorez les techniques de mise à niveau à votre disposition lors de la Mise à niveau d'Aurora MySQL version 2 vers la version 3.
- Pour vous aider à choisir le bon moment et la bonne approche pour la mise à niveau, découvrez les différences entre Aurora MySQL version 3 et votre environnement actuel avec Comparaison d'Aurora MySQL version 2 et Aurora MySQL version 3.
- Une fois que vous avez choisi l'option la plus pratique et la plus efficace, simulez une mise à niveau sur place sur un cluster cloné en vous reportant à Planification d'une mise à niveau de version majeure d'un cluster Aurora MySQL.
- Prenez connaissance des Vérifications préalables aux mises à niveau de version majeure pour Aurora MySQL. La vérification préalable de la mise à niveau peut déterminer si la base de données peut être mise à niveau avec succès et s'il y a des problèmes d'incompatibilité entre les applications après la mise à niveau, ainsi que des considérations relatives aux performances, aux procédures de maintenance et autres aspects similaires.
- Tous les types ou versions de clusters Aurora MySQL ne peuvent pas utiliser le mécanisme de mise à niveau sur place. Pour plus d'informations, consultez Chemins de mise à niveau d'une version majeure Aurora MySQL.

En cas de question ou de doute, contactez l'équipe AWS Support sur les forums de la communauté et l'assistance Premium.

Réalisation de la mise à niveau :

Vous pouvez utiliser l'une des techniques de mise à niveau suivantes. La durée d'indisponibilité de votre système dépend de la technique choisie.

- Déploiements bleu/vert : lorsque la priorité absolue est de réduire la durée d'indisponibilité de l'application, vous pouvez utiliser les déploiements bleu/vert Amazon RDS pour effectuer la mise à niveau de la version majeure dans les clusters de bases de données Amazon Aurora provisionnés. Un déploiement bleu/vert crée un environnement intermédiaire qui copie l'environnement de

production. Vous pouvez modifier le cluster de bases de données Aurora dans l'environnement vert (intermédiaire) sans affecter les charges de travail de production. La bascule prend généralement moins d'une minute, et n'engendre pas de perte de données. Pour plus d'informations, consultez Présentation des (Amazon Aurora Blue/Green). Cette méthode permet de minimiser la durée d'indisponibilité, mais nécessite des ressources supplémentaires pendant la mise à niveau.

- Mises à niveau sur place : vous pouvez effectuer une mise à niveau sur place au cours de laquelle Aurora réalisera automatiquement une vérification préalable pour vous, mettra le cluster hors ligne, le sauvegardera, appliquera la mise à niveau et remettra votre cluster en ligne. Une mise à niveau sur place d'une version majeure peut être effectuée en quelques clics et ne nécessite aucune autre coordination ni aucun basculement vers d'autres clusters, mais elle implique une durée d'indisponibilité. Pour plus d'informations, consultez Comment effectuer une mise à niveau sur place.
- Restauration à partir d'un instantané : vous pouvez mettre à niveau votre cluster Aurora MySQL version 2 en restaurant un instantané Aurora MySQL version 2 dans un cluster Aurora MySQL version 3. Pour ce faire, vous devez suivre le processus de capture d'instantané et de restauration à partir de ce dernier. Ce processus implique une interruption de la base de données puisque vous effectuez la restauration à partir d'un instantané.

Après la mise à niveau :

Après la mise à niveau, vous devez surveiller de près votre système (application et base de données) et procéder à des optimisations le cas échéant. En suivant scrupuleusement les étapes préalables à la mise à niveau, vous réduisez au minimum les modifications nécessaires. Pour plus d'informations, consultez Résolution des problèmes de performances de base de données Amazon Aurora MySQL.

Pour en savoir plus sur les méthodes, la planification, les tests et le dépannage des mises à niveau de la version majeure d'Aurora MySQL, assurez-vous de lire attentivement Mise à niveau de la version majeure d'un cluster de bases de données Amazon Aurora MySQL, y compris Dépannage de la mise à niveau sur place d'Aurora MySQL. Notez que certains types d'instances ne sont pas pris en charge par Aurora MySQL version 3. Pour plus d'informations, consultez Classes d'instance de base de données Amazon Aurora.

Chemin de mise à niveau pour les clusters de bases de données Aurora Serverless v1

La mise à niveau entre les versions majeures nécessite une planification et des tests plus étendus que pour une version mineure. Le processus peut prendre beaucoup de temps. Nous considérons

la mise à niveau comme un processus en trois étapes, comprenant des activités avant, pendant et après la mise à niveau.

Aurora MySQL version 2 (avec compatibilité MySQL 5.7) continue de bénéficier du support standard pour les clusters Aurora Serverless v1.

Si vous souhaitez passer à Aurora MySQL version 3 (avec compatibilité MySQL 8.0) et continuer d'exécuter Aurora Serverless, vous pouvez utiliser Amazon Aurora Serverless v2. Pour comprendre les différences entre Aurora Serverless v1 et Aurora Serverless v2, consultez Comparaison d'Aurora Serverless v2 avec Aurora Serverless v1.

Mise à niveau vers Aurora Serverless v2 : vous pouvez mettre à niveau un cluster Aurora Serverless v1 vers Aurora Serverless v2. Pour plus d'informations, consultez Mise à niveau d'un cluster Aurora Serverless v1 vers Aurora Serverless v2.

Préparation à la fin de vie d'Amazon Aurora Édition compatible avec MySQL version 1

La fin de vie d'Amazon Aurora Édition compatible avec MySQL version 1 (avec compatibilité MySQL 5.6) est prévue pour le 28 février 2023. Amazon recommande de mettre à niveau tous les clusters (mis en service et Aurora Serverless) exécutant Aurora MySQL version 1 vers Aurora MySQL version 2 (avec compatibilité MySQL 5.7) ou Aurora MySQL version 3 (avec compatibilité MySQL 8.0). Faites-le avant que la version 1 de Aurora MySQL n'arrive à la fin de sa période de support.

Plusieurs méthodes permettent d'effectuer la mise à niveau des clusters de bases de données provisionnées par Aurora, d'Aurora MySQL version 1 vers Aurora MySQL version 2. Vous trouverez des instructions sur le mécanisme de mise à niveau sur place dans Comment effectuer une mise à niveau sur place. Une autre façon de réaliser la mise à niveau consiste à prendre un instantané d'un cluster Aurora MySQL version 1 et à restaurer cet instantané dans un cluster Aurora MySQL version 2. Vous pouvez également suivre un processus en plusieurs étapes qui exécute l'ancien et le nouveau cluster côte à côte. Pour en savoir plus sur chaque méthode, consultez Mise à niveau de la version majeure d'un cluster de bases de données Amazon Aurora MySQL.

Pour les clusters de bases de données Aurora Serverless v1, vous pouvez effectuer une mise à niveau sur place d'Aurora MySQL version 1 vers Aurora MySQL version 2. Pour en savoir plus sur cette méthode, consultez Modification d'un cluster de bases de données Aurora Serverless v1.

Pour les clusters de bases de données provisionnées par Aurora, vous pouvez effectuer les mises à niveau d'Aurora MySQL version 1 vers Aurora MySQL version 3, en suivant un processus de mise à niveau en deux étapes :

1. Mettez à niveau Aurora MySQL version 1 vers Aurora MySQL version 2 en utilisant les méthodes décrites précédemment.
2. Mettez à niveau Aurora MySQL version 2 vers Aurora MySQL version 3 en utilisant les mêmes méthodes que pour effectuer une mise à niveau de la version 1 vers la version 2. Pour en savoir plus, consultez Mise à niveau d'Aurora MySQL version 2 vers la version 3. Notez le Différences de fonctions entre Aurora MySQL version 2 et 3.

Vous trouverez les dates de fin de vie à venir pour les versions majeures d'Aurora dans Versions d'Amazon Aurora. Amazon met automatiquement à niveau tous les clusters que vous ne mettez pas à niveau vous-même avant la date de fin de vie. Après la date de fin de vie, ces mises à niveau automatiques vers la version majeure ultérieure se produisent pendant une fenêtre de maintenance planifiée pour les clusters.

Voici des étapes supplémentaires pour la mise à niveau des clusters Aurora MySQL version 1 (mis en service et Aurora Serverless) qui arrivent en fin de vie. Pour chacun d'eux, l'heure de début est 0h00 UTC (temps universel coordonné).

1. Jusqu'au 28 février 2023 : vous pouvez à tout moment procéder à la mise à niveau des clusters Aurora MySQL version 1 (avec compatibilité MySQL 5.6) vers Aurora MySQL version 2 (avec compatibilité MySQL 5.7). À partir de la version 2 d'Aurora MySQL, vous pouvez effectuer une mise à niveau supplémentaire vers la version 3 d'Aurora MySQL (avec compatibilité avec MySQL 8.0) pour les clusters de bases de données provisionnées par Aurora.
2. 16 janvier 2023 : après cette période, vous ne pourrez plus créer de nouveaux clusters ou instances Aurora MySQL version 1 à partir de la AWS Management Console ou de AWS Command Line Interface (AWS CLI). Vous ne pouvez pas non plus ajouter de nouvelles régions secondaires à une base de données globale Aurora. Cela peut affecter votre capacité à vous remettre d'une interruption non planifiée, comme indiqué à la section Reprise d'une base de données Amazon Aurora globale à partir d'une panne non planifiée, car vous ne pouvez pas effectuer les étapes 5 et 6 après ce délai. Vous ne serez pas non plus en mesure de créer un nouveau réplica en lecture entre plusieurs régions exécutant Aurora MySQL version 1. Vous pouvez toujours effectuer les opérations suivantes pour les clusters Aurora MySQL version 1 existants jusqu'au 28 février 2023 :

- Restaurez un instantané pris d'un cluster Aurora MySQL version 1 à la même version que le cluster instantané d'origine.
- Ajoutez des réplicas en lecture (non applicable pour les clusters de bases de données Aurora Serverless).
- Modifiez la configuration de l'instance.
- Effectuez une restauration à un instant donné.
- Créez des clones de clusters de version 1 existants.
- Créez un nouveau réplica en lecture entre plusieurs régions exécutant Aurora MySQL version 2 ou ultérieure.

3. 28 février 2023 : après ce délai, nous prévoyons de mettre automatiquement à niveau les clusters Aurora MySQL version 1 vers la version par défaut d'Aurora MySQL version 2 dans la fenêtre de maintenance planifiée suivante. La restauration des instantanés de bases de données Aurora MySQL version 1 entraîne une mise à niveau automatique du cluster restauré vers la version par défaut d'Aurora MySQL version 2 au moment de la restauration.

La mise à niveau entre les versions majeures nécessite une planification et des tests plus étendus que pour une version mineure. Le processus peut prendre beaucoup de temps.

Dans les situations où la priorité absolue est de réduire la durée d'indisponibilité, vous pouvez également utiliser des déploiements bleu/vert pour effectuer la mise à niveau de la version majeure dans les clusters de bases de données Amazon Aurora provisionnés. Un déploiement bleu/vert crée un environnement intermédiaire qui copie l'environnement de production. Vous pouvez apporter des modifications au cluster de bases de données Aurora dans l'environnement vert (intermédiaire) sans affecter les charges de travail de production. La bascule prend généralement moins d'une minute, sans perte de données et sans qu'il soit nécessaire de modifier les applications. Pour plus d'informations, consultez Présentation des (Amazon Aurora Blue/Green).

Une fois la mise à niveau terminée, il se peut que vous ayez également un travail de suivi à faire. Par exemple, vous pourriez avoir besoin d'un suivi en raison de différences dans la compatibilité SQL, du fonctionnement de certaines fonctions liées à MySQL ou de la configuration des paramètres entre l'ancienne et la nouvelle version.

Pour en savoir plus sur les méthodes, la planification, les tests et le dépannage des mises à niveau de la version majeure de Aurora MySQL, assurez-vous de lire attentivement Mise à niveau de la version majeure d'un cluster de bases de données Amazon Aurora MySQL.

Recherche de clusters affectés par ce processus de fin de vie

Pour trouver les clusters affectés par ce processus de fin de vie, utilisez les procédures suivantes.

> **Important**
>
> Assurez-vous d'exécuter ces instructions dans chaque Région AWS et pour chaque Compte AWS où vos ressources sont situées.

Console

Pour trouver un cluster Aurora MySQL version 1

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Dans la case Filter by databases (Filtrer par bases de données), saisissez 5.6.
4. Recherchez les occurrences Aurora MySQL dans la colonne du moteur.

AWS CLI

Pour rechercher les clusters affectés par ce processus de fin de vie à l'aide de l'AWS CLI, effectuez un appel à la commande describe-db-clusters. Vous pouvez utiliser l'exemple de script suivant.

Example

```
aws rds describe-db-clusters --include-share --query 'DBClusters[?Engine==`aurora`].
{EV:EngineVersion, DBCI:DBClusterIdentifier, EM:EngineMode}' --output table --region
 us-east-1

        +-----------------------------------------+
        |            DescribeDBClusters           |
        +---------------+-------------+-----------+
        |     DBCI      |     EM      |    EV     |
        +---------------+-------------+-----------+
        |  my-database-1|  serverless |  5.6.10a  |
        +---------------+-------------+-----------+
```

API RDS

Pour trouver les clusters de bases de données Aurora MySQL exécutant Aurora MySQL version 1, utilisez l'opération d'API RDS DescribeDBClusters avec les paramètres requis suivants :

- `DescribeDBClusters`
  - Filters.Filter.N
    - Nom
      - engine
    - Values.Value.N
      - ['aurora']

Mise à niveau des clusters de bases de données Amazon Aurora MySQL

Vous pouvez mettre à niveau un cluster de bases de données Aurora MySQL pour obtenir des correctifs de bogues ou de nouvelles fonctions Aurora MySQL ou pour passer à une version entièrement nouvelle du moteur de base de données sous-jacent. Les sections suivantes vous expliquent comment.

> **Note**
>
> Le type de mise à niveau que vous effectuez dépend de la durée d'indisponibilité acceptable pour votre cluster, du nombre de tests de vérification que vous prévoyez d'effectuer et de l'importance des correctifs de bogues spécifiques ou des nouvelles fonctionnalités pour votre cas d'utilisation. Déterminez également si vous comptez réaliser de petites mises à niveau régulièrement ou des mises à niveau occasionnelles qui ignorent plusieurs versions intermédiaires. Pour chaque mise à niveau, vous pouvez modifier la version majeure, la version mineure et le niveau de correctif de votre cluster. Si vous ne connaissez pas la différence entre les versions majeures et les versions mineures d'Aurora MySQL et les niveaux de correctif, vous pouvez lire les informations de base sur Vérification des numéros de version d'Aurora MySQL.

> **Tip**
>
> Vous pouvez minimiser la durée d'indisponibilité nécessaire à la mise à niveau d'un cluster de bases de données en utilisant un déploiement bleu/vert. Pour plus d'informations, consultez

Utilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données.

Rubriques

- Mise à niveau de la version mineure ou du niveau de correctif d'un cluster de bases de données Aurora MySQL
- Mise à niveau de la version majeure d'un cluster de bases de données Amazon Aurora MySQL

Mise à niveau de la version mineure ou du niveau de correctif d'un cluster de bases de données Aurora MySQL

Vous pouvez utiliser les méthodes suivantes pour mettre à niveau la version mineure d'un cluster de bases de données ou appliquer un correctif à un cluster de bases de données :

- Mise à niveau d'Aurora MySQL par modification de la version du moteur (pour Aurora MySQL versions 2 et 3)
- Activation des mises à niveau automatiques entre versions mineures Aurora MySQL

Pour plus d'informations sur la façon dont l'application de correctifs sans interruption peut réduire les interruptions pendant le processus de mise à niveau, consultez Utilisation des correctifs sans durée d'indisponibilité.

Pour en savoir plus sur la mise à niveau d'une version mineure de votre cluster de bases de données Aurora MySQL, consultez les rubriques suivantes.

Rubriques

- Avant d'effectuer une mise à niveau de version mineure
- Vérifications préalables aux mises à niveau de version mineure pour Aurora MySQL
- Mise à niveau d'Aurora MySQL par modification de la version du moteur
- Activation des mises à niveau automatiques entre versions mineures Aurora MySQL
- Utilisation des correctifs sans durée d'indisponibilité
- Technique alternative de mise à niveau bleu/vert

Avant d’effectuer une mise à niveau de version mineure

Nous vous recommandons d’effectuer les actions suivantes pour réduire la durée d’indisponibilité lors d’une mise à niveau de version mineure :

- La maintenance du cluster de bases de données Aurora doit être effectuée pendant une période de faible trafic. Utilisez Performance Insights pour identifier ces périodes afin de configurer correctement les fenêtres de maintenance. Pour plus d’informations sur Performance Insights, consultez Surveillance de la charge de la base de données avec Performance Insights sur Amazon RDS. Pour plus d’informations sur la fenêtre de maintenance du cluster de bases de données, consultez Ajustement du créneau de maintenance préféré pour un cluster de bases de données.
- Utilisez des kits AWS SDK qui prennent en charge le backoff exponentiels et l’instabilité en tant que bonne pratique. Pour plus d’informations, consultez Backoff exponentiel et instabilité.

Vérifications préalables aux mises à niveau de version mineure pour Aurora MySQL

Lorsque vous lancez une mise à niveau d’une version mineure, Amazon Aurora exécute automatiquement des vérifications préalables.

Ces vérifications préalables sont obligatoires. Vous ne pouvez pas choisir de les ignorer. Elles offrent les avantages suivants :

- Elles vous permettent d’éviter toute durée d’indisponibilité non planifiée pendant la mise à niveau.
- En cas d’incompatibilités, Amazon Aurora empêche la mise à niveau et vous fournit un journal vous permettant d’en savoir plus. Vous pouvez ensuite utiliser ce journal pour préparer votre base de données pour la mise à niveau en réduisant ces incompatibilités. Pour obtenir des informations détaillées sur la suppression des incompatibilités, consultez Préparation de votre installation pour la mise à niveau dans la documentation MySQL.

Les vérifications préalables s’exécutent avant que l’instance de base de données soit arrêtée pour la mise à niveau, ce qui signifie que leur exécution n’entraîne aucune durée d’indisponibilité. Si les vérifications préalables identifient une incompatibilité, Aurora annule automatiquement la mise à niveau avant que l’instance de base de données soit arrêtée. Aurora génère également un événement pour cette incompatibilité. Pour plus d’informations sur les événements Amazon Aurora, consultez Utiliser la notification d’événements d’Amazon RDS.

Aurora enregistre des informations détaillées sur chaque incompatibilité dans le fichier journal `PrePatchCompatibility.log`. Dans la plupart des cas, l’entrée de journal inclut un lien vers

la documentation MySQL permettant de corriger l'incompatibilité. Pour plus d'informations sur l'affichage des fichiers journaux, consultez Liste et affichage des fichiers journaux de base de données.

En raison de la nature des vérifications préalables, elle analysent les objets dans votre base de données. L'analyse entraîne la consommation de ressources et augmente le temps nécessaire pour la mise à niveau.

Mise à niveau d'Aurora MySQL par modification de la version du moteur

La mise à niveau de la version mineure d'un cluster de bases de données Aurora MySQL applique des correctifs supplémentaires et de nouvelles fonctionnalités à un cluster existant.

Ce type de mise à niveau s'applique aux clusters Aurora MySQL où la version d'origine et la version mise à niveau ont toutes deux la même version majeure d'Aurora MySQL (version 2 ou 3). Le processus est rapide et simple, car il n'implique ni conversion pour les métadonnées Aurora MySQL, ni réorganisation de vos données de table.

Vous effectuez ce type de mise à niveau en modifiant la version du moteur du cluster de bases de données à l'aide d'AWS Management Console, de l'AWS CLI ou de l'API RDS. Par exemple, si votre cluster exécute Aurora MySQL 3.x, choisissez une version 3.x ultérieure.

Si vous effectuez une mise à niveau mineure sur une base de données globale Aurora, mettez à niveau tous les clusters secondaires avant de mettre à niveau le cluster principal.

> **Note**
>
> Pour effectuer une mise à niveau de version mineure vers Aurora MySQL version 3.04.* ou ultérieure, ou version 2.12.*, procédez comme suit :
>
> 1. Supprimez toutes les régions secondaires du cluster global. Suivez les étapes de Dissociation d'un cluster d'une base de données Amazon Aurora globale.
> 2. Mettez à niveau la version du moteur de la région principale vers la version 3.04.* ou ultérieure, ou vers la version 2.12.*, selon le cas. Suivez les étapes de To modify the engine version of a DB cluster.
> 3. Ajoutez des régions secondaires au cluster global. Suivez les étapes de Ajout d'une Région AWS à une base de données Amazon Aurora globale.

Pour modifier la version du moteur d'un cluster de bases de données

- Avec la console – Modifiez les propriétés de votre cluster. Dans la fenêtre Modifier le cluster de bases de données, modifiez la version du moteur Aurora MySQL dans la zone Version du moteur de base de données. Si vous n'êtes pas familier avec la procédure générale de modification d'un cluster, suivez les instructions à l'adresse Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API.
- Avec la AWS CLI – Appelez la commande de l'AWS CLI modify-db-cluster et spécifiez le nom de votre cluster de bases de données pour l'option `--db-cluster-identifier` et la version du moteur pour l'option `--engine-version`.

  Par exemple, pour une mise à niveau vers Aurora MySQL version 3.04.1, définissez l'option `--engine-version` sur `8.0.mysql_aurora.3.04.1`. Spécifiez l'option `--apply-immediately` pour mettre à jour immédiatement la version du moteur de votre cluster de bases de données.
- Avec l'API RDS – Appelez l'opération de l'API ModifyDBCluster et spécifiez le nom de votre cluster de bases de données pour le paramètre `DBClusterIdentifier` et la version du moteur pour le paramètre `EngineVersion`. Définissez le paramètre `ApplyImmediately` sur `true` pour mettre à jour immédiatement la version du moteur du cluster de bases de données.

Activation des mises à niveau automatiques entre versions mineures Aurora MySQL

Pour un cluster de bases de données Amazon Aurora MySQL, vous pouvez spécifier qu'Aurora met automatiquement à niveau le cluster de bases de données vers de nouvelles versions mineures. Pour ce faire, définissez la propriété `AutoMinorVersionUpgrade` (Mise à niveau automatique des versions mineures dans la AWS Management Console) du cluster de bases de données.

Des mises à niveau automatiques se produisent dans la fenêtre de maintenance. Si les différentes instances de base de données du cluster de bases de données ont des fenêtres de maintenance différentes de la fenêtre de maintenance du cluster, cette dernière est prioritaire.

La mise à niveau automatique des versions mineures ne s'applique pas aux types de clusters Aurora MySQL suivants :

- Clusters faisant partie d'une base de données globale Aurora
- Clusters ayant des réplicas entre régions

La durée de l'indisponibilité varie en fonction de la charge de travail, de la taille du cluster, de la quantité de données du journal binaire et de la possibilité pour Aurora d'utiliser la fonction d'application de correctifs sans durée d'indisponibilité. Aurora redémarre le cluster de bases de

données. Une courte période d'indisponibilité est donc possible avant que vous puissiez reprendre l'utilisation de votre cluster. En particulier, la quantité de données consignées dans le journal binaire affecte la durée de récupération. L'instance de base de données traite les données de journal binaire pendant la restauration. Ainsi, un volume élevé de données de journal binaire augmente le temps de restauration.

> **Note**
>
> Aurora n'effectue des mises à niveau automatiques que si le paramètre `AutoMinorVersionUpgrade` est activé pour toutes les instances de base de données de votre cluster de bases de données. Pour en savoir plus sur la façon de le définir et sur la façon dont ce paramètre fonctionne lorsqu'il est appliqué aux niveaux du cluster et de l'instance, consultez Mises à niveau automatiques des versions mineures pour les clusters de bases de données Aurora.
>
> Ensuite, s'il existe un chemin de mise à niveau pour les instances du cluster de bases de données vers une version mineure du moteur de base de données dont la valeur `AutoUpgrade` est définie sur true, la mise à niveau aura lieu. Le paramètre `AutoUpgrade` est dynamique et est défini par RDS.
>
> Des mises à niveau automatiques de version mineure sont effectuées vers la version mineure par défaut.

Vous pouvez utiliser une commande CLI telle que la suivante pour vérifier le statut du paramètre `AutoMinorVersionUpgrade` pour toutes les instances de base de données figurant dans vos clusters Aurora MySQL.

```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Le résultat de cette commande est semblable à ce qui suit :

```
[
  {
      "DBInstanceIdentifier": "db-t2-medium-instance",
      "DBClusterIdentifier": "cluster-57-2020-06-03-6411",
      "AutoMinorVersionUpgrade": true
  },
  {
```

```
        "DBInstanceIdentifier": "db-t2-small-original-size",
        "DBClusterIdentifier": "cluster-57-2020-06-03-6411",
        "AutoMinorVersionUpgrade": false
    },
    {
        "DBInstanceIdentifier": "instance-2020-05-01-2332",
        "DBClusterIdentifier": "cluster-57-2020-05-01-4615",
        "AutoMinorVersionUpgrade": true
    },
... output omitted ...
```

Dans cet exemple, la paramètre Activer la mise à niveau automatique des versions mineures est désactivé pour le cluster de bases de données `cluster-57-2020-06-03-6411`, car il est désactivé pour l'une des instances de base de données du cluster.

Utilisation des correctifs sans durée d'indisponibilité

L'exécution de mises à niveau pour les clusters de bases de données Aurora MySQL implique la possibilité d'une panne lorsque la base de données est arrêtée et pendant sa mise à niveau. Par défaut, si vous démarrez la mise à niveau alors que la base de données est occupée, vous perdez toutes les connexions et transactions traitées par le cluster de bases de données. Si vous attendez que la base de données soit inactive pour effectuer la mise à niveau, vous devrez peut-être attendre longtemps.

La fonctionnalité d'application de correctifs sans durée d'indisponibilité (ZDP) tente, dans un souci d'optimisation, de conserver les connexions client tout au long de la mise à niveau d'Aurora MySQL. Si l'application de correctifs sans durée d'indisponibilité s'exécute correctement, les sessions d'application sont conservées et le moteur de base de données redémarre pendant que la mise à niveau est en cours. Le redémarrage du moteur de base de données peut entraîner une chute du débit qui dure de quelques secondes à environ une minute.

L'application de correctifs sans durée d'indisponibilité ne s'applique pas aux éléments suivants :

- Correctifs et mises à niveau du système d'exploitation
- Mises à niveau de version majeure.

ZDP est disponible pour toutes les versions d'Aurora MySQL et classes d'instance de base de données prises en charge.

ZDP n'est pas pris en charge pour les bases de données globales Aurora ou Aurora Serverless v1.

> **Note**
>
> Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à la production. Pour plus de détails sur les classes d'instance T, consultez Utilisation de classes d'instance T pour le développement et les tests.

Vous pouvez voir les métriques des attributs importants pendant l'application de correctifs sans durée d'indisponibilité dans le journal des erreurs MySQL. Vous pouvez également voir des informations sur les moments où Aurora MySQL utilise ZDP ou choisit de ne pas l'utiliser sur la page Events (Événements) de la AWS Management Console.

Dans Aurora MySQL, Aurora peut appliquer un correctif sans durée d'indisponibilité, que la réplication des journaux binaires soit activée ou non. Si la réplication des journaux binaires est activée, Aurora MySQL supprime automatiquement la connexion à la cible des journaux binaires pendant une opération d'application de correctifs sans durée d'indisponibilité. Aurora MySQL se reconnecte automatiquement à la cible des journaux binaires et reprend la réplication une fois le redémarrage terminé.

L'application de correctifs sans durée d'indisponibilité fonctionne également en combinaison avec les améliorations du redémarrage dans Aurora MySQL. Le correctif de l'instance de base de données d'enregistreur corrige automatiquement les lecteurs en même temps. Après avoir exécuté le correctif, Aurora restaure les connexions sur les instances de base de données d'enregistreur et de lecteur.

L'application de correctifs sans durée d'indisponibilité peut ne pas s'exécuter correctement dans les conditions suivantes :

- Des transactions ou des requêtes de longue durée sont en cours. Si Aurora peut exécuter l'application de correctifs sans durée d'indisponibilité dans ce cas, les connexions ouvertes sont conservées.
- Des tables temporaires, des verrous utilisateur ou des verrous de table sont utilisés, par exemple lorsque des instructions de langage de définition de données (DDL) s'exécutent. Aurora abandonne ces connexions.
- Des modifications de paramètre sont en attente.

Si aucun créneau adéquat pour l'application de correctifs sans durée d'indisponibilité ne devient disponible en raison d'une ou de plusieurs de ces conditions, l'application de correctifs adopte de nouveau le comportement standard.

Bien que les connexions restent intactes après une opération d'application de correctifs sans durée d'indisponibilité, certaines variables et fonctions sont réinitialisées. Les types d'informations suivants ne sont pas conservés lors d'un redémarrage causé par une application de correctifs sans durée d'indisponibilité :

- Variables globales Aurora restaure les variables de session, mais pas les variables globales après le redémarrage.
- Variables d'état. En particulier, la valeur de disponibilité signalée par le statut du moteur est réinitialisée après un redémarrage utilisant les mécanismes ZDR ou ZDP.
- `LAST_INSERT_ID`.
- État `auto_increment` en mémoire pour les tables. L'état d'auto-incrémentation en mémoire est réinitialisé. Pour plus d'informations sur les valeurs d'auto-incrémentation, reportez-vous au manuel de référence MySQL.
- Les informations de diagnostic des tableaux `INFORMATION_SCHEMA` et `PERFORMANCE_SCHEMA`. Ces informations de diagnostic apparaissent également dans la sortie de commandes telles que `SHOW PROFILE` et `SHOW PROFILES`.

Les activités suivantes liées au redémarrage sans interruption sont signalées sur la page Events (Événements) :

- Tentative de mise à niveau de la base de données sans interruption.
- Tentative de mise à niveau de la base de données terminée sans aucune interruption. L'événement indique combien de temps le processus a duré. L'événement indique également combien de connexions ont été préservées pendant le redémarrage et combien de connexions ont été annulées. Vous pouvez consulter le journal des erreurs de la base de données pour en savoir plus sur ce qui s'est passé pendant le redémarrage.

Technique alternative de mise à niveau bleu/vert

Dans certains cas, votre priorité absolue est d'effectuer une bascule immédiate de l'ancien cluster vers un cluster mis à niveau. Dans certains cas, vous pouvez également suivre un processus en plusieurs étapes qui exécute l'ancien et le nouveau cluster côte à côte. Dans ce cas, répliquez les

données de l’ancien cluster au nouveau jusqu’à ce que ce dernier soit prêt à prendre le relais. Pour en savoir plus, consultez Utilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données.

Mise à niveau de la version majeure d’un cluster de bases de données Amazon Aurora MySQL

Dans un numéro de version Aurora MySQL tel que 3.04.1, le 3 représente la version majeure. Aurora MySQL version 2 est déjà compatible avec MySQL 5.7. Aurora MySQL version 3 est déjà compatible avec MySQL 8.0.

La mise à niveau entre les versions majeures nécessite une planification et des tests plus étendus que pour une version mineure. Le processus peut prendre beaucoup de temps. Une fois la mise à niveau terminée, il se peut que vous ayez également un travail de suivi à faire, Par exemple, cela peut se produire en raison des différences de compatibilité SQL ou du fonctionnement de certaines fonctions liées à MySQL. Cela peut également se produire en raison de paramètres différents entre l’ancienne et la nouvelle version.

Table des matières

Mise à niveau d’Aurora MySQL version 2 vers la version 3

Si vous avez un cluster compatible MySQL 5.7 et souhaitez le mettre à niveau vers un cluster compatible MySQL 8.0, vous pouvez le faire en exécutant un processus de mise à niveau sur le cluster lui-même. Ce type de mise à niveau est appelé mise à niveau sur place, en opposition aux mises à niveau que vous effectuez en créant un nouveau cluster. Cette technique conserve le point de terminaison et d’autres caractéristiques du cluster. La mise à niveau est relativement rapide car elle ne nécessite pas la copie de toutes vos données vers un nouveau volume de cluster. Cette stabilité permet de réduire au minimum les modifications de configuration de vos applications. Elle aide également à réduire la quantité de tests du cluster mis à niveau. En effet, le nombre d’instances de base de données et leurs classes d’instance restent les mêmes.

Le mécanisme de mise à niveau sur place implique l’arrêt de votre cluster de bases de données pendant que l’opération. Aurora effectue un arrêt propre et termine les opérations en suspens telles que la restauration des transactions et la purge des annulations. Pour plus d’informations, consultez Fonctionnement de la mise à niveau sur place d’une version majeure de Aurora MySQL.

La méthode de mise à niveau sur place est pratique, car elle est simple à effectuer et réduit au minimum les modifications de configuration des applications associées. Par exemple, une mise à niveau sur place conserve les points de terminaison et l'ensemble d'instances de base de données de votre cluster. Toutefois, le temps nécessaire pour une mise à niveau sur place peut varier en fonction des propriétés de votre schéma et du niveau d'activité de votre cluster. Ainsi, en fonction des besoins de votre cluster, vous pouvez choisir parmi les techniques de mise à niveau :

- Mises à niveau sur place
- Déploiement bleu/vert
- Restauration d'instantané

> **Note**
>
> Si vous utilisez l'AWS CLI ou l'API RDS pour la méthode de mise à niveau par restauration des instantanés, vous devez exécuter une opération ultérieure pour créer une instance de base de données d'enregistreur dans le cluster de bases de données restauré.

Pour obtenir des informations générales sur Aurora MySQL version 3, ainsi que ses nouvelles fonctions, consultez Aurora MySQL version 3 compatible avec MySQL 8.0.

Pour plus de détails sur la planification d'une mise à niveau, consultez Planification d'une mise à niveau de version majeure d'un cluster Aurora MySQL et Comment effectuer une mise à niveau sur place.

Chemins de mise à niveau d'une version majeure Aurora MySQL

Tous les types ou versions de clusters Aurora MySQL ne peuvent pas utiliser le mécanisme de mise à niveau sur place. Consultez le tableau suivant pour connaître le chemin de mise à niveau approprié pour chaque cluster Aurora MySQL.

Type de cluster de bases de données Aurora MySQL	Peut-il utiliser la mise à niveau sur place ?	Action
Cluster Aurora MySQL provisionné, version 2	Oui	La mise à niveau sur place est prise en charge pour les clusters Aurora MySQL compatibles avec MySQL 5.7. Pour en savoir plus sur la mise à niveau vers Aurora MySQL version 3, consultez Planification d'une mise à niveau de version majeure d'un cluster Aurora MySQL et Comment effectuer une mise à niveau sur place.
Cluster Aurora MySQL provisionné, version 3	Ne s'applique pas	Utilisez une procédure de mise à niveau de version mineure pour passer aux versions Aurora MySQL version 3.
Aurora Serverless v1Cluster	Ne s'applique pas	Aurora Serverless v1 est pris en charge uniquement pour Aurora MySQL version 2.
Aurora Serverless v2Cluster	Ne s'applique pas	Aurora Serverless v2 est pris en charge uniquement pour Aurora MySQL version 3.
Cluster d'une base de données Aurora globale	Oui	Pour mettre à niveau Aurora MySQL de la version 2 vers la version 3, suivez la procédure de mise à niveau sur place pour les clusters d'une base de données Aurora globale. Effectuez la mise à niveau sur le cluster global. Aurora met à niveau le cluster principal et tous les clusters secondaires dans la base de données globale en même temps. Si vous utilisez la AWS CLI ou l'API RDS, appelez la commande `modify-global-cluster` ou l'opérati

Type de cluster de bases de données Aurora MySQL	Peut-il utiliser la mise à niveau sur place ?	Action
		on `ModifyGlobalCluster` au lieu de `modify-db-cluster` ou de `ModifyDBCluster` . Vous ne pouvez effectuer une mise à niveau sur place d'Aurora MySQL version 2 vers la version 3 que si le paramètre `lower_case_table_names` est défini par défaut et que vous redémarrez votre base de données globale. Pour plus d'informations, consultez Mises à niveau de version majeure..
Cluster compatible avec les requêtes parallèles	Oui	Vous pouvez effectuer une mise à niveau sur place.
Cluster cible de la réplication de journaux binaires	Peut-être	Si la réplication de journaux binaires provient d'un cluster Aurora MySQL, vous pouvez effectuer une mise à niveau sur place. Vous ne pouvez pas effectuer la mise à niveau si la réplication de journaux binaires provient d'une instance de RDS for MySQL ou d'une instance de base de données MySQL sur site. Dans ce cas, vous pouvez effectuer la mise à niveau à l'aide du mécanisme de restauration d'instantané.

Type de cluster de bases de données Aurora MySQL	Peut-il utiliser la mise à niveau sur place ?	Action
Cluster sans instances de base de données	Non	À l'aide de l'AWS CLI ou de l'API RDS, vous pouvez créer un cluster Aurora MySQL sans instances de base de données attachées. De la même manière, vous pouvez supprimer toutes les instances de base de données d'un cluster Aurora MySQL tout en laissant les données du volume de cluster intactes. Lorsqu'un cluster ne comporte aucune instance de base de données, vous ne pouvez pas effectuer une mise à niveau sur place. Le mécanisme de mise à niveau nécessite une instance de scripteur dans le cluster pour effectuer des conversio ns sur les tables système, les fichiers de données, etc. Dans ce cas, utilisez la AWS CLI ou l'API RDS pour créer une instance de scripteur pour le cluster. Ensuite, vous pouvez effectuer une mise à niveau sur place.
Cluster avec retour en arrière activé	Oui	Vous pouvez effectuer une mise à niveau sur place pour un cluster Aurora MySQL utilisant la fonctionnalité de retour en arrière. Toutefois, après la mise à niveau, vous ne pouvez pas effectuer de retour en arrière du cluster à un moment antérieur à la mise à niveau.

Fonctionnement de la mise à niveau sur place d'une version majeure de Aurora MySQL

Aurora MySQL effectue les mises à niveau de version majeure en tant que processus en plusieurs étapes. Vous pouvez vérifier l'état actuel d'une mise à niveau. Certaines étapes de la mise à niveau fournissent également des informations sur l'état d'avancement. Au début de chaque étape, Aurora MySQL enregistre un événement. Vous pouvez examiner les événements lorsqu'ils ont lieu sur la

page Events (Événements) de la console RDS. Pour en savoir plus sur l'utilisation des événements, consultez Utiliser la notification d'événements d'Amazon RDS.

> **Important**
>
> Une fois que le processus a démarré, il se poursuit jusqu'à ce que la mise à niveau réussisse ou échoue. Vous ne pouvez pas annuler la mise à niveau tant qu'elle est en cours. Si la mise à niveau échoue, Aurora annule toutes les modifications et votre cluster garde la même version du moteur, ainsi que les mêmes métadonnées et autres éléments qu'auparavant.

Le processus de mise à niveau comporte les étapes suivantes :

1. Aurora effectue une série de vérifications préalables avant de lancer le processus de mise à niveau. Votre cluster continue de fonctionner pendant que Aurora effectue ces vérifications. Par exemple, le cluster ne peut pas avoir de transactions XA à l'état préparé ou être en train de traiter des instructions en langage de définition de données (DDL). Par exemple, il se peut que vous deviez arrêter les applications qui soumettent certains types d'instructions SQL. Vous pouvez également attendre simplement que certaines instructions à longue exécution soient terminées. Ensuite, tentez de relancer la mise à niveau. Certaines vérifications consistent à examiner les conditions n'empêchant pas la mise à niveau, mais peuvent prendre beaucoup de temps.

   Si Aurora détecte que les conditions requises ne sont pas réunies, modifiez les conditions identifiées dans les détails de l'événement. Suivez les instructions dans Dépannage de la mise à niveau sur place d'Aurora MySQL. Si Aurora détecte des conditions susceptibles de ralentir la mise à niveau, prévoyez de surveiller la mise à niveau sur une longue période.
2. Aurora met votre cluster hors ligne. Aurora effectue ensuite un ensemble de tests similaire à celui de l'étape précédente pour confirmer qu'aucun problème n'est apparu pendant le processus d'arrêt. Si Aurora détecte à ce stade des conditions pouvant empêcher la mise à niveau, Aurora annule la mise à niveau et remet le cluster en ligne. Dans ce cas, indiquez quand les conditions ne s'appliquent plus et relancez la mise à niveau.
3. Aurora crée un instantané de votre volume de cluster. Supposons que vous découvriez la compatibilité ou d'autres types de problèmes une fois la mise à niveau terminée. Ou supposons que vous souhaitiez effectuer des tests à l'aide des clusters d'origine et des clusters mis à niveau. Dans ce cas, vous pouvez effectuer une restauration à partir de cet instantané pour créer un nouveau cluster avec la version du moteur d'origine et les données d'origine.

> **Tip**
>
> Cet instantané est un instantané manuel. Cependant, Aurora peut le créer et poursuivre le processus de mise à niveau, même si vous avez atteint votre quota d'instantanés manuels. Cet instantané sera conservé définitivement (si nécessaire) jusqu'à ce que vous le supprimiez. Une fois tous les tests postérieurs à la mise à niveau terminés, vous pouvez supprimer cet instantané pour réduire les frais de stockage.

4. Aurora clone votre volume de cluster. Le clonage est une opération rapide qui n'implique pas la copie des données de la table elles-mêmes. Si Aurora rencontre un problème lors de la mise à niveau, les données d'origine du volume de cluster cloné sont restaurées et le cluster est remis en ligne. Le volume cloné temporairement pendant la mise à niveau n'est pas soumis à la limite habituelle du nombre de clones pour un seul volume de cluster.
5. Aurora effectue un arrêt propre de l'instance de base de données du scripteur. Pendant l'arrêt propre, les événements de progression sont enregistrés toutes les 15 minutes pour les opérations suivantes. Vous pouvez examiner les événements lorsqu'ils ont lieu sur la page Events (Événements) de la console RDS.
   - Aurora purge les enregistrements d'annulation des anciennes versions des lignes.
   - Aurora restaure toutes les transactions non validées.
6. Aurora met à niveau la version du moteur de l'instance de base de données du scripteur :
   - Aurora installe le fichier binaire de la nouvelle version du moteur de l'instance de base de données du scripteur.
   - Aurora utilise l'instance de base de données du scripteur pour mettre à niveau vos données au format compatible MySQL 5.7. Lors de cette étape, Aurora modifie les tables système et effectue d'autres conversions qui affectent les données de votre volume de cluster. En particulier, Aurora met à niveau les métadonnées de partition dans les tables système pour qu'elles soient compatibles avec le format de partition MySQL 5.7. Cette étape peut prendre beaucoup de temps si les tables de votre cluster ont de nombreuses partitions.

     Si des erreurs se produisent au cours de cette étape, vous pouvez trouver les détails dans les journaux d'erreurs MySQL. Une fois cette étape démarrée, si le processus de mise à niveau échoue pour une raison quelconque, Aurora restaure les données d'origine du volume de cluster cloné.
7. Aurora met à niveau la version du moteur des instances de base de données du scripteur.

8. Le processus de mise à niveau est terminé. Aurora enregistre un événement final pour indiquer que le processus de mise à niveau s'est terminé avec succès. Votre cluster de bases de données exécute désormais la nouvelle version majeure.

Planification d'une mise à niveau de version majeure d'un cluster Aurora MySQL

Pour vous aider à choisir le bon moment et la bonne approche pour la mise à niveau, découvrez les différences entre Aurora MySQL version 3 et votre environnement actuel :

- Si vous effectuez une conversion à partir de RDS for MySQL 8.0 ou MySQL 8.0 Community Edition, consultez Comparaison d'Aurora MySQL version 3 et de MySQL 8.0 Community Edition.
- Si vous effectuez une mise à niveau depuis Aurora MySQL version 2, RDS for MySQL 5.7 ou MySQL 5.7 Community Edition, consultez Comparaison d'Aurora MySQL version 2 et Aurora MySQL version 3.
- Créez de nouvelles versions compatibles avec MySQL 8.0 de tous les groupes de paramètres personnalisés. Appliquez toutes les valeurs de paramètres personnalisés nécessaires aux nouveaux groupes de paramètres. Consultez Modifications des paramètres d'Aurora MySQL version 3 pour en savoir plus sur les changements de paramètres.
- Vérifiez le schéma de votre base de données Aurora MySQL version 2 et les définitions d'objets pour vous assurer de l'utilisation des nouveaux mots-clés réservés introduits dans MySQL 8.0 Community Edition. Faites-le avant la mise à niveau. Pour en savoir plus, consultez MySQL 8.0 New Keywords and Reserved Words (MySQL 8.0 : nouveaux mots-clés et mots réservés) dans la documentation MySQL.

Vous trouverez également d'autres considérations et astuces de mise à niveau spécifiques à MySQL dans Changements dans MySQL 8.0 dans le manuel de référence MySQL. Par exemple, vous pouvez utiliser la commande `mysqlcheck --check-upgrade` pour analyser les bases de données Aurora MySQL existantes et identifier les problèmes potentiels de mise à niveau.

> **Note**
>
> Nous recommandons d'utiliser des classes d'instance de base de données plus importantes lors de la mise à niveau vers Aurora MySQL version 3 en utilisant la technique de mise à niveau sur place ou de restauration des instantanés. Exemples : db.r5.24xlarge et db.r6g.16xlarge. Cela permet au processus de mise à niveau de se terminer plus rapidement en utilisant la majorité de la capacité CPU disponible sur l'instance de base de données.

Vous pouvez passer à la classe d'instance de base de données de votre choix une fois la mise à niveau de la version majeure terminée.

Une fois la mise à niveau terminée, vous pouvez suivre les procédures post-mise à niveau dans Nettoyage postérieur à la mise à niveau pour Aurora MySQL version 3. Enfin, testez les fonctionnalités et les performances de votre application.

Si vous effectuez une conversion à partir de RDS for MySQL ou de la version communautaire de MySQL, suivez la procédure de migration expliquée dans Migration de données vers un cluster de bases de données Amazon Aurora MySQL. Dans certains cas, vous pouvez utiliser la réplication des journaux binaires pour synchroniser vos données avec un cluster Aurora MySQL version 3 dans le cadre de la migration. Le cas échéant, le système source doit exécuter une version compatible avec votre cluster de bases de données cible.

Pour vous assurer que vos applications et procédures d'administration fonctionnent correctement après la mise à niveau d'un cluster entre les versions majeures, effectuez une planification et une préparation anticipées. Pour connaître les types de code de gestion à mettre à jour pour vos scripts AWS CLI ou applications basées sur l'API RDS, consultez Comment les mises à niveau sur place affectent les groupes de paramètres d'un cluster. Voir aussi Modifications apportées aux propriétés du cluster entre les versions d'Aurora MySQL.

Pour découvrir les problèmes que vous pourriez rencontrer lors de la mise à niveau, consultez Dépannage de la mise à niveau sur place d'Aurora MySQL. Pour les problèmes pouvant entraîner une mise à niveau longue, vous pouvez tester ces conditions au préalable et les corriger.

Note

Un mécanisme de mise à niveau sur place implique l'arrêt de votre cluster de bases de données pendant que l'opération a lieu. Aurora effectue un arrêt complet et termine les opérations en cours telles que la purge des annulations. Une mise à niveau peut prendre beaucoup de temps s'il y a de nombreux enregistrements d'annulation à purger. Nous vous recommandons de n'effectuer la mise à niveau qu'une fois que la longueur de la liste d'historique (HLL) sera faible. Une valeur généralement acceptable pour la liste HLL est de 100 000 ou moins. Pour plus d'informations, lisez ce billet de blog.

Simulation de la mise à niveau en clonant votre cluster de bases de données

Vous pouvez vérifier la compatibilité des applications, les performances, les procédures de maintenance et d'autres considérations pour le cluster mis à niveau. Pour ce faire, vous pouvez effectuer une simulation de la mise à niveau avant de procéder à la mise à niveau elle-même. Cette technique peut être particulièrement utile pour les clusters de production. Ici, il est important de limiter la durée d'indisponibilité et de s'assurer que le cluster mis à niveau soit opérationnel dès la fin de la mise à niveau.

Procédez comme suit :

1. Créez un clone du cluster d'origine. Suivez la procédure décrite dans Clonage d'un volume pour un cluster de bases de données Amazon Aurora.
2. Configurez un ensemble d'instances de base de données d'écriture et de lecture similaire à celui du cluster d'origine.
3. Effectuez une mise à niveau sur place du cluster cloné. Suivez la procédure décrite dans Comment effectuer une mise à niveau sur place.

   Démarrez la mise à niveau immédiatement après avoir créé le clone. Ainsi, le volume de cluster reste identique à l'état du cluster d'origine. Si le clone est inactif avant la mise à niveau, Aurora effectue des processus de nettoyage de base de données en arrière-plan. Dans ce cas, la mise à niveau du clone n'est pas une simulation précise de la mise à niveau du cluster d'origine.
4. Testez la compatibilité des applications, les performances, les procédures d'administration, etc., à l'aide du cluster cloné.
5. Si vous rencontrez des problèmes, modifiez vos plans de mise à niveau de manière à en tenir compte. Par exemple, adaptez n'importe quel code d'application pour qu'il soit compatible avec le jeu de fonctions de la version ultérieure. Estimez la durée de la mise à niveau en fonction de la quantité de données dans votre cluster. Vous pouvez également choisir de planifier la mise à niveau à un moment où le cluster n'est pas occupé.
6. Après avoir vérifié le bon fonctionnement de vos applications et de votre charge de travail avec le cluster de test, vous pouvez effectuer la mise à niveau sur place de votre cluster de production.
7. Essayez de limiter la durée d'indisponibilité totale de votre cluster pendant une mise à niveau de version majeure. Pour ce faire, assurez-vous que la charge de travail sur le cluster est faible ou nulle au moment de la mise à niveau. En particulier, assurez-vous qu'aucune transaction longue n'est en cours lorsque vous lancez la mise à niveau.

Déploiements bleu/vert

Dans certains cas, votre priorité absolue est d’effectuer une bascule immédiate de l’ancien cluster vers un cluster mis à niveau. Dans certains cas, vous pouvez également suivre un processus en plusieurs étapes qui exécute l’ancien et le nouveau cluster côte à côte. Dans ce cas, répliquez les données de l’ancien cluster au nouveau jusqu’à ce que ce dernier soit prêt à prendre le relais. Pour en savoir plus, consultez Utilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données.

Vérifications préalables aux mises à niveau de version majeure pour Aurora MySQL

Les mises à niveau de version majeure de MySQL (lorsque vous passez de MySQL 5.7 à MySQL 8.0, par exemple) impliquent des modifications architecturales significatives qui nécessitent une planification et une préparation minutieuses. Contrairement aux mises à niveau de version mineure qui se concentrent principalement sur la mise à jour du logiciel du moteur de base de données et, dans certains cas, des tables système, les mises à niveau majeures de MySQL impliquent souvent des changements fondamentaux dans la façon dont la base de données stocke et gère ses métadonnées.

Pour vous aider à identifier ces incompatibilités, lors de la mise à niveau d'Aurora MySQL version 2 vers la version 3, Aurora exécute automatiquement des vérifications de la compatibilité de la mise à niveau (vérifications préalables) pour examiner les objets de votre cluster de bases de données et identifier les incompatibilités connues susceptibles d'empêcher la mise à niveau d'avoir lieu. Pour plus d'informations sur les vérifications préalables d'Aurora MySQL, consultez Référence des vérifications préalables avec description pour Aurora MySQL. Les vérifications préalables d'Aurora s'exécutent en plus de celles effectuées par l'utilitaire de vérification de mise à niveau de Community MySQL.

Ces vérifications préalables sont obligatoires. Vous ne pouvez pas choisir de les ignorer. Elles offrent les avantages suivants :

- Elles limitent le risque d'échecs de mise à niveau susceptibles d'entraîner une durée d'indisponibilité prolongée.
- En cas d'incompatibilités, Amazon Aurora empêche la mise à niveau d'avoir lieu et vous fournit un journal vous permettant d'en savoir plus. Vous pouvez ensuite utiliser ce journal pour préparer votre base de données pour la mise à niveau vers la version 3 de MySQL en résolvant ces incompatibilités. Pour obtenir des informations détaillées sur la résolution des incompatibilités, consultez Préparation de votre installation pour la mise à niveau dans la documentation MySQL et Mise à niveau vers MySQL 8.0 ? Ce que vous devez savoir… (langue française non garantie) sur le blog MySQL Server.

  Pour plus d'informations sur la mise à niveau vers MySQL 8.0, consultez Mise à niveau de MySQL dans la documentation MySQL.

Les vérifications préalables s'exécutent avant que votre cluster de bases de données ne soit mis hors ligne pour la mise à niveau de la version majeure. Si les vérifications préalables identifient une incompatibilité, Aurora annule automatiquement la mise à niveau avant que l'instance de base de

données soit arrêtée. Aurora génère également un événement pour cette incompatibilité. Pour plus d'informations sur les événements Amazon Aurora, consultez Utiliser la notification d'événements d'Amazon RDS.

Une fois les vérifications préalables terminées, Aurora enregistre des informations détaillées sur chaque incompatibilité dans le fichier `upgrade-prechecks.log`. Dans la plupart des cas, l'entrée de journal inclut un lien vers la documentation MySQL permettant de corriger l'incompatibilité. Pour plus d'informations sur l'affichage des fichiers journaux, consultez Liste et affichage des fichiers journaux de base de données.

> **Note**
>
> En raison de la nature des vérifications préalables, elles analysent les objets dans votre base de données. L'analyse entraîne la consommation de ressources et augmente le temps nécessaire pour la mise à niveau. Pour plus d'informations sur les considérations relatives aux performances de la vérification préalable, consultez Processus de vérification préalable pour Aurora MySQL.

Table des matières

Processus de vérification préalable pour Aurora MySQL

Comme décrit précédemment, le processus de mise à niveau d'Aurora MySQL implique l'exécution de vérifications de la compatibilité (vérifications préalables) sur votre base de données avant de procéder à la mise à niveau de la version majeure.

Pour les mises à niveau sur place, les vérifications préalables s'exécutent sur votre instance de base de données d'enregistreur lorsqu'elle est en ligne. Si la vérification préalable aboutit, la mise à niveau a lieu. Si des erreurs sont détectées, elles sont journalisées dans le fichier `upgrade-prechecks.log`, et la mise à niveau est annulée. Avant de tenter une nouvelle mise à niveau, corrigez les erreurs renvoyées dans le fichier `upgrade-prechecks.log`.

Pour les mises à niveau basées sur la restauration d'un instantané, la vérification préalable s'exécute pendant le processus de restauration. En cas de succès, votre base de données sera mise à niveau vers la nouvelle version d'Aurora MySQL. Si des erreurs sont détectées, elles sont journalisées dans le fichier `upgrade-prechecks.log`, et la mise à niveau est annulée. Avant de tenter une nouvelle mise à niveau, corrigez les erreurs renvoyées dans le fichier `upgrade-prechecks.log`.

Pour plus d'informations, consultez Identification de la cause des échecs de mise à niveau de version majeure Aurora MySQL et Référence des vérifications préalables avec description pour Aurora MySQL.

Pour surveiller l'état de la vérification préalable, vous pouvez consulter les événements suivants sur votre cluster de bases de données.

État de la vérification préalable	Message d'événement	Action
Démarré(e)	Préparation de la mise à niveau en cours : lancement des vérifications préalables à la mise à niveau en ligne.	Aucun
Échec	Le cluster de bases de données est dans un état qui ne permet pas la mise à niveau : les vérifications préalables à la mise à niveau ont échoué. Pour plus de	Recherchez les erreurs dans `upgrade-prechecks.log` . Corrigez les erreurs. Réessayez la mise à niveau.

État de la vérification préalable	Message d'événement	Action
	détails, consultez le fichier upgrade-prechecks.log. Pour plus d'informations sur le dépannage de la cause de l'échec de la mise à niveau, consultez https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Upgrading.Troubleshooting.html	
Réussi(e)	Préparation de la mise à niveau en cours : vérifications préalables à la mise à niveau en ligne terminées.	La vérification préalable a abouti et aucune erreur n'a été renvoyée. Recherchez les avertissements et les notifications dans `upgrade-prechecks.log` .

Pour plus d'informations sur l'affichage des événements, consultez Affichage d'événements Amazon RDS.

Format du journal des vérifications préalables pour Aurora MySQL

Une fois que les vérifications de la compatibilité de la mise à niveau (vérifications préalables) sont terminées, vous pouvez consulter le fichier `upgrade-prechecks.log`. Ce fichier journal contient les résultats, les objets concernés et les informations de correction de chaque vérification préalable.

Toute erreur bloque la mise à niveau. Vous devez les résoudre avant de réessayer la mise à niveau.

Les avertissements et les notifications sont moins critiques, mais nous vous recommandons tout de même de les examiner attentivement pour vous assurer qu'il n'y a aucun problème de compatibilité avec la charge de travail de l'application. Résolvez rapidement tous les problèmes identifiés.

Le fichier journal a le format suivant :

- `targetVersion` : version compatible avec MySQL de la mise à niveau d'Aurora MySQL.
- `auroraServerVersion` : version d'Aurora MySQL sur laquelle la vérification préalable a été exécutée.
- `auroraTargetVersion` : version d'Aurora MySQL vers laquelle vous effectuez la mise à niveau.
- `checksPerformed` : contient la liste des vérifications préalables effectuées.
- `id` : nom de la vérification préalable en cours d'exécution.
- `title` : description de la vérification préalable en cours d'exécution.
- `status` : n'indique pas si la vérification préalable a réussi ou échoué, mais indique l'état de la requête correspondante :
  - `OK` : la requête de vérification préalable a été exécutée et s'est terminée avec succès.
  - `ERROR` : la requête de vérification préalable n'a pas pu être exécutée. Cela peut être dû à des problèmes tels que des contraintes de ressources, des redémarrages inattendus d'instances ou l'interruption de la requête de vérification préalable de la compatibilité.

    Pour plus d'informations, consultez cet exemple.
- `description` : description générale de l'incompatibilité et de la procédure à suivre pour remédier au problème.
- `documentationLink` : le cas échéant, un lien vers la documentation pertinente d'Aurora MySQL ou de MySQL est indiqué ici. Pour plus d'informations, consultez Référence des vérifications préalables avec description pour Aurora MySQL.
- `detectedProblems` : si la vérification préalable renvoie une erreur, un avertissement ou une notification, cela indique les détails de l'incompatibilité et les objets incompatibles le cas échéant :
  - `level` : niveau d'incompatibilité détecté par la vérification préalable. Les niveaux valides sont les suivants :
    - `Error` : la mise à niveau ne peut pas être effectuée tant que vous n'avez pas résolu l'incompatibilité.
    - `Warning` : la mise à niveau peut avoir lieu, mais un objet, une syntaxe ou une configuration obsolète a été détecté. Lisez attentivement les avertissements et résolvez-les rapidement afin d'éviter des problèmes dans les futures versions.
    - `Notice` : la mise à niveau peut avoir lieu, mais un objet, une syntaxe ou une configuration obsolète a été détecté. Lisez attentivement les notifications et résolvez-les rapidement afin d'éviter des problèmes dans les futures versions.

  - `dbObject` : nom de l'objet de base de données dans lequel l'incompatibilité a été détectée.
  - `description` : description détaillée de l'incompatibilité et de la procédure à suivre pour remédier au problème.
- `errorCount` : nombre d'erreurs d'incompatibilité détectées. Ces erreurs bloquent la mise à niveau.
- `warningCount` : nombre d'avertissements d'incompatibilité détectés. Ils ne bloquent pas la mise à niveau, mais traitez-les rapidement afin d'éviter des problèmes dans les futures versions.
- `noticeCount` : nombre de notifications d'incompatibilité détectées. Elles ne bloquent pas la mise à niveau, mais traitez-les rapidement afin d'éviter des problèmes dans les futures versions.
- `Summary` : résumé du nombre d'erreurs, d'avertissements et de notifications de compatibilité générés par la vérification préalable.

Exemples de sorties du journal de vérification préalable pour Aurora MySQL

Les exemples suivants montrent la sortie du journal de vérification préalable que vous pourriez voir. Pour plus de détails sur les vérifications préalables exécutées, consultez Référence des vérifications préalables avec description pour Aurora MySQL.

État de la vérification préalable OK, aucune incompatibilité détectée

La requête de vérification préalable s'est terminée avec succès. Aucune incompatibilité n'a été détectée.

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinytext",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny text columns",
  "status": "OK",
  "detectedProblems": []
},
```

État de la vérification préalable OK, erreur détectée

La requête de vérification préalable s'est terminée avec succès. Une erreur a été détectée.

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexes",
  "status": "OK",
```

```
 "description": "Consider dropping the prefix indexes of geometry columns and
restart the upgrade.",
 "detectedProblems": [
     {
       "level": "Error",
       "dbObject": "test25.sbtest1",
       "description": "Table `test25`.`sbtest1` has an index `idx_t1` on geometry
column/s. Mysql 8.0 does not support this type of index on a geometry column
https://dev.mysql.com/worklog/task/?id=11808. To upgrade to MySQL 8.0, Run 'DROP
INDEX `idx_t1` ON `test25`.`sbtest1`;"
     },
}
```

État de la vérification préalable OK, avertissement détecté

Des avertissements peuvent être renvoyés en cas de réussite ou d'échec d'une vérification préalable.

Ici, la requête de vérification préalable s'est terminée avec succès. Deux avertissements ont été détectés.

```
{
  "id": "zeroDatesCheck",
  "title": "Zero Date, Datetime, and Timestamp values",
  "status": "OK",
  "description": "Warning: By default zero date/datetime/timestamp values are no
longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are included
in SQL_MODE by default. These modes should be used with strict mode as they will be
merged with strict mode in a future release. If you do not include these modes in
your SQL_MODE setting, you are able to insert date/datetime/timestamp values that
contain zeros. It is strongly advised to replace zero values with valid ones, as
they may not work correctly in the future.",
 "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
 "detectedProblems": [
     {
       "level": "Warning",
       "dbObject": "global.sql_mode",
       "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
which allows insertion of zero dates"
     },
     {
       "level": "Warning",
       "dbObject": "session.sql_mode",
```

```
        "description": " of 10 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
      }
    ]
}
```

État de la vérification préalable ERROR, aucune incompatibilité signalée

La requête de vérification préalable a échoué avec une erreur. Les incompatibilités n'ont donc pas pu être vérifiées.

```
{
  "id": "auroraUpgradeCheckForDatafilePathInconsistency",
  "title": "Check for inconsistency related to ibd file path.",
  "status": "ERROR",
  "description": "Can't connect to MySQL server on 'localhost:3306' (111) at
 13/08/2024 12:22:20 UTC. This failure can occur due to low memory available on the
 instance for executing upgrade prechecks. Please check 'FreeableMemory' Cloudwatch
 metric to verify the available memory on the instance while executing prechecks. If
 instance ran out of memory, we recommend to retry the upgrade on a higher instance
 class."
}
```

Cet échec peut être dû à un redémarrage inattendu de l'instance ou à l'interruption d'une requête de vérification préalable de la compatibilité sur la base de données pendant l'exécution. Par exemple, sur des classes d'instance de base de données de petite taille, vous pouvez rencontrer ce problème lorsque la mémoire disponible sur l'instance est insuffisante.

Vous pouvez utiliser la métrique `FreeableMemory` d'Amazon CloudWatch pour vérifier la mémoire disponible sur l'instance lors de l'exécution des vérifications préalables. Si l'instance n'a pas suffisamment de mémoire, nous vous recommandons de réessayer la mise à niveau sur une classe d'instance de base de données plus grande. Dans certains cas, vous pouvez utiliser un déploiement bleu/vert. Cela permet aux vérifications préalables et aux mises à niveau de s'exécuter sur le cluster de bases de données « vert » indépendamment de la charge de travail de production, qui consomme également des ressources système.

Pour plus d'informations, consultez Résolution des problèmes d'utilisation de la mémoire pour les bases de données Aurora MySQL.

Résumé d'une vérification préalable : une erreur et trois avertissements détectés

Les vérifications préalables de la compatibilité contiennent également des informations sur les versions source et cible d'Aurora MySQL, ainsi qu'un résumé du nombre d'erreurs, d'avertissements et de notifications à l'issue de la vérification préalable.

Par exemple, la sortie suivante indique qu'une tentative de mise à niveau d'Aurora MySQL 2.11.6 vers Aurora MySQL 3.07.1 a été effectuée. Cette mise à niveau a renvoyé une erreur, trois avertissements et aucune notification. Comme les mises à niveau ne peuvent pas être effectuées lorsqu'une erreur est renvoyée, vous devez résoudre le problème de compatibilité routineSyntaxCheck, puis réessayer la mise à niveau.

```
{
  "serverAddress": "/tmp%2Fmysql.sock",
  "serverVersion": "5.7.12 - MySQL Community Server (GPL)",
  "targetVersion": "8.0.36",
  "auroraServerVersion": "2.11.6",
  "auroraTargetVersion": "3.07.1",
  "outfilePath": "/rdsdbdata/tmp/PreChecker.log",
  "checksPerformed": [{
      ... output for each individual precheck ...
      .
      .
      {
        "id": "oldTemporalCheck",
        "title": "Usage of old temporal type",
        "status": "OK",
          "detectedProblems": []
      },
      {
        "id": "routinesSyntaxCheck",
        "title": "MySQL 8.0 syntax check for routine-like objects",
        "status": "OK",
        "description": "The following objects did not pass a syntax check with
 the latest MySQL 8.0 grammar. A common reason is that they reference names that
 conflict with new reserved keywords. You must update these routine definitions and
 `quote` any such references before upgrading.",
        "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
        "detectedProblems": [{
            "level": "Error",
            "dbObject": "test.select_res_word",
            "description": "at line 2,18: unexpected token 'except'"
        }]
```

```
      },
      .
      .
      .
      {
        "id": "zeroDatesCheck",
        "title": "Zero Date, Datetime, and Timestamp values",
        "status": "OK",
        "description": "Warning: By default zero date/datetime/timestamp values
 are no longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are
 included in SQL_MODE by default. These modes should be used with strict mode as
 they will be merged with strict mode in a future release. If you do not include
 these modes in your SQL_MODE setting, you are able to insert date/datetime/
timestamp values that contain zeros. It is strongly advised to replace zero values
 with valid ones, as they may not work correctly in the future.",
        "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
        "detectedProblems": [{
            "level": "Warning",
            "dbObject": "global.sql_mode",
            "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
 which allows insertion of zero dates"
            },
            {
            "level": "Warning",
            "dbObject": "session.sql_mode",
            "description": " of 8 session(s) does not contain either NO_ZERO_DATE or
 NO_ZERO_IN_DATE which allows insertion of zero dates"
            }
          ]
       },
       .
       .
       .
  }],
  "errorCount": 1,
  "warningCount": 3,
  "noticeCount": 0,
  "Summary": "1 errors were found. Please correct these issues before upgrading to
 avoid compatibility issues."
}
```

Vérification préalable des performances pour Aurora MySQL

Les vérifications préalables de la compatibilité s'exécutent avant que l'instance de base de données soit mise hors ligne pour la mise à niveau. Leur exécution n'implique donc aucune durée d'indisponibilité. Cependant, elles peuvent avoir un impact sur la charge de travail de l'application exécutée sur l'instance de base de données d'enregistreur. Les vérifications préalables accèdent au dictionnaire de données via les tables information_schema, un processus qui peut être lent si le nombre d'objets de base de données est élevé. Prenez en compte les facteurs suivants :

- La durée de la vérification préalable varie en fonction du nombre d'objets de base de données tels que les tables, les colonnes, les routines et les contraintes. L'exécution des clusters de bases de données contenant un grand nombre d'objets peut prendre plus de temps.

  Par exemple, removedFunctionsCheck peut prendre plus de temps et utiliser davantage de ressources en fonction du nombre d'objets stockés.
- Pour les mises à niveau sur place, l'utilisation d'une classe d'instance de base de données de grande taille (par exemple, db.r5.24xlarge ou db.r6g.16xlarge) peut accélérer la mise à niveau grâce à davantage de processeur. Vous pourrez réduire la taille de la classe d'instance de base de données après la mise à niveau.
- Les requêtes portant sur `information_schema` entre plusieurs bases de données peuvent être lentes, notamment si le nombre d'objets est élevé et si les instances de base de données sont de petite taille. Dans ce cas, envisagez d'utiliser le clonage, la restauration d'instantané ou un déploiement bleu/vert pour les mises à niveau.
- L'utilisation des ressources (processeur, mémoire) par la vérification préalable peut augmenter avec la hausse du nombre d'objets, ce qui entraîne des durées d'exécution plus longues sur les instances de base de données de petite taille. Dans ce cas, envisagez d'utiliser le clonage, la restauration d'instantané ou un déploiement bleu/vert pour les mises à niveau afin d'effectuer des tests.

  Si les vérifications préalables échouent en raison de ressources insuffisantes, vous pouvez détecter ce problème dans le journal de vérification préalable à l'aide de la sortie d'état :

```
"status": "ERROR",
```

Pour plus d'informations, consultez Fonctionnement de la mise à niveau sur place d'une version majeure de Aurora MySQL et Planification d'une mise à niveau de version majeure d'un cluster Aurora MySQL.

Résumé des vérifications préalables de mise à niveau de Community MySQL

Voici une liste générale des incompatibilités entre MySQL 5.7 et 8.0 :

- Votre cluster de bases de données compatible avec MySQL 5.7 ne doit pas utiliser de fonctionnalités non prises en charge par MySQL 8.0.

  Pour plus d'informations, consultez Fonctionnalités supprimées dans MySQL 8.0 dans la documentation MySQL.
- Il ne doit y avoir aucune violation de mot clé ou de mot réservé. Certains mots clés doivent être réservés dans MySQL 8.0 alors qu'ils ne l'étaient pas par le passé.

  Pour en savoir plus, consultez la section Mots clés et mots réservés dans la documentation MySQL.
- Pour une meilleure prise en charge d'Unicode, envisagez de convertir les objets qui utilisent le jeu de caractères `utf8mb3` pour utiliser le jeu de caractères `utf8mb4`. Le jeu de caractères `utf8mb3` est obsolète. Envisagez également d'utiliser `utf8mb4` pour référencer les jeux de caractères au lieu de `utf8`. Actuellement, `utf8` est un alias du jeu de caractères `utf8mb3`.

  Pour en savoir plus, consultez la section Le jeu de caractères utf8mb3 (encodage Unicode 3 octets en UTF-8) dans la documentation MySQL.
- Il ne doit y avoir aucune table InnoDB avec un format de ligne autre que celui par défaut.
- Il ne doit y avoir aucun attribut de type longueur `ZEROFILL` ou `display`.
- Aucune table partitionnée ne doit utiliser de moteur de stockage dépourvu de prise en charge native du partitionnement.
- Aucune table de la base de données du système `mysql`dans MySQL 5.7 ne doit avoir le même nom que la table utilisée dans le dictionnaire de données MySQL 8.0.
- Les tables ne doivent pas utiliser de fonctions ou de types de données obsolètes.
- Aucun nom de contrainte de clé étrangère ne doit dépasser 64 caractères.
- Aucun mode SQL obsolète ne doit être défini dans la configuration variable de votre système `sql_mode`.
- Aucune table ni procédure stockée avec des éléments de colonne `ENUM` ou `SET` individuels ne doit dépasser 255 caractères de longueur.
- Aucune partition de table ne doit se trouver dans les espaces de table InnoDB partagés.
- Il ne doit pas y avoir aucune référence circulaire dans les chemins des fichiers de données des espaces de table.

- Aucune requête ni définition de programme stockée ne doit utiliser de qualificateur `ASC` ou `DESC` pour les clauses `GROUP BY`.
- Aucune variable système ne doit être supprimée, et les variables système doivent utiliser les nouvelles valeurs par défaut pour MySQL 8.0.
- Aucune valeur de date, de date/heure ou d'horodatage ne doit correspondre à zéro (`0`).
- Il ne doit y avoir aucune incohérence de schéma résultant de la suppression ou de la corruption de fichiers.
- Aucun nom de table ne doit contenir la chaîne de caractères `FTS`.
- Aucune table InnoDB ne doit appartenir à un autre moteur.
- Tous les noms de table ou de schéma doivent être valides pour MySQL 5.7.

Pour plus de détails sur les vérifications préalables exécutées, consultez Référence des vérifications préalables avec description pour Aurora MySQL.

Pour plus d'informations sur la mise à niveau vers MySQL 8.0, consultez Mise à niveau de MySQL dans la documentation MySQL. Pour une description générale des modifications apportées dans MySQL 8.0, consultez Nouveautés de MySQL 8.0 dans la documentation MySQL.

Résumé des vérifications préalables de mise à niveau d'Aurora MySQL

Aurora MySQL a des exigences spécifiques lors de la mise à niveau de la version 2 vers la version 3, notamment les suivantes :

- Il ne doit pas y avoir de syntaxe SQL obsolète, telle que `SQL_CACHE`, `SQL_NO_CACHE` et `QUERY_CACHE`, dans les vues, les routines, les déclencheurs et les événements.
- Aucune colonne `FTS_DOC_ID` ne peut se trouver dans une table sans l'index `FTS`.
- Il ne doit y avoir aucune incompatibilité de définition de colonne entre le dictionnaire de données InnoDB et la définition réelle de la table.
- Tous les noms de bases de données et de tables doivent être en minuscules lorsque le paramètre `lower_case_table_names` est défini sur `1`.
- Les déclencheurs ne doivent pas avoir de definer manquant ou vide, ni de contexte de création non valide.
- Tous les noms de déclencheurs d'une base de données doivent être uniques.
- Les restaurations DDL et Fast DDL ne sont pas prises en charge dans Aurora MySQL version 3. Les bases de données ne doivent contenir aucun artefact lié à ces fonctionnalités.

- Les tables au format de ligne `REDUNDANT` ou `COMPACT` ne peuvent pas avoir d'index supérieurs à 767 octets.
- La longueur du préfixe des index définis sur les colonnes de texte `tiny` ne peut pas dépasser 255 octets. Avec le jeu de caractères `utf8mb4`, cela limite la longueur de préfixe prise en charge à 63 caractères.

  Une longueur de préfixe plus grande était autorisée dans MySQL 5.7 en utilisant le paramètre `innodb_large_prefix`. Ce paramètre est obsolète dans MySQL 8.0.
- Il ne doit y avoir aucune incohérence des métadonnées InnoDB dans la table `mysql.host`.
- Il ne doit pas y avoir de différence de type de données de colonne dans les tables système.
- Il ne doit y avoir aucune transaction XA à l'état `prepared`.
- Les noms de colonne dans les vues ne peuvent pas dépasser 64 caractères.
- Les caractères spéciaux des procédures stockées ne peuvent pas être incohérents.
- Les tables ne peuvent pas avoir d'incohérences entre les chemins des fichiers de données.

Pour plus de détails sur les vérifications préalables exécutées, consultez Référence des vérifications préalables avec description pour Aurora MySQL.

Référence des vérifications préalables avec description pour Aurora MySQL

Les vérifications préalables de mise à niveau pour Aurora MySQL sont décrites ici en détail.

Table des matières

Erreurs

Les vérifications préalables suivantes génèrent des erreurs lorsque la vérification préalable échoue et que la mise à niveau ne peut pas avoir lieu.

Rubriques

- Vérifications préalables de MySQL qui signalent des erreurs
- Vérifications préalables d'Aurora MySQL qui signalent des erreurs

Vérifications préalables de MySQL qui signalent des erreurs

Les vérifications préalables suivantes sont tirées de Community MySQL :

- checkTableMysqlSchema
- circularDirectoryCheck
- columnsWhichCannotHaveDefaultsCheck
- depreciatedSyntaxCheck
- engineMixupCheck
- enumSetElementLengthCheck
- foreignKeyLengthCheck
- getDuplicateTriggers
- getEventsWithNullDefiner
- getMismatchedMetadata
- getTriggersWithNullDefiner
- getValueOfVariablelower_case_table_names
- groupByAscSyntaxCheck
- mysqlEmptyDotTableSyntaxCheck
- mysqlIndexTooLargeCheck
- mysqlInvalid57NamesCheck
- mysqlOrphanedRoutinesCheck
- mysqlSchemaCheck
- nonNativePartitioningCheck
- oldTemporalCheck

- partitionedTablesInSharedTablespaceCheck
- removedFunctionsCheck
- routineSyntaxCheck
- schemaInconsistencyCheck

checkTableMysqlSchema

Niveau de vérification préalable : erreur

Problèmes signalés par la commande **`check table x for upgrade`** pour le schéma **`mysql`**

Avant de démarrer la mise à niveau vers Aurora MySQL version 3, `check table for upgrade` est exécuté sur chaque table du schéma `mysql` de l'instance de base de données. La commande `check table for upgrade` examine les tables pour détecter tout problème potentiel pouvant survenir lors d'une mise à niveau vers une version plus récente de MySQL. L'exécution de cette commande avant de tenter une mise à niveau contribue à identifier et à résoudre les incompatibilités à l'avance, ce qui facilite le processus de mise à niveau réel.

Cette commande effectue différentes vérifications sur chaque table, telles que les suivantes :

- Vérification de la compatibilité de la structure et des métadonnées de la table avec la version MySQL cible
- Identification des fonctionnalités obsolètes ou supprimées qui sont utilisées par la table
- Confirmation de la possibilité de mettre à niveau la table sans perte de données

Pour plus d'informations, consultez Instruction CHECK TABLE dans la documentation MySQL.

Exemple de sortie :

```
{
  "id": "checkTableMysqlSchema",
  "title": "Issues reported by 'check table x for upgrade' command for mysql
 schema.",
  "status": "OK",
  "detectedProblems": []
}
```

Le résultat de cette vérification préalable dépend de l'erreur rencontrée et du moment où elle est détectée, car `check table for upgrade` effectue plusieurs vérifications.

Si vous rencontrez des erreurs lors de cette vérification préalable, ouvrez une demande d'assistance auprès d'AWS Support pour que l'incohérence des métadonnées soit résolue. Vous pouvez également réessayer la mise à niveau en procédant à un vidage logique, puis en effectuant une restauration sur un nouveau cluster de bases de données Aurora MySQL version 3.

circularDirectoryCheck

Niveau de vérification préalable : erreur

Références circulaires à des répertoires dans les chemins de fichiers de données des espaces de table

Depuis MySQL 8.0.17, la clause `CREATE TABLESPACE ... ADD DATAFILE` n'autorise plus les références circulaires à des répertoires. Pour éviter les problèmes de mise à niveau, supprimez toutes les références circulaires à des répertoires dans les chemins de fichiers de données des espaces de table avant de passer à Aurora MySQL version 3.

Exemple de sortie :

```
{
  "id": "circularDirectory",
  "title": "Circular directory references in tablespace data file paths",
  "status": "OK",
  "description": "Error: Following tablespaces contain circular directory references
 (e.g. the reference '/../') in data file paths which as of MySQL 8.0.17 are not
 permitted by the CREATE TABLESPACE ... ADD DATAFILE clause. An exception to the
 restriction exists on Linux, where a circular directory reference is permitted if
 the preceding directory is a symbolic link. To avoid upgrade issues, remove any
 circular directory references from tablespace data file paths before upgrading.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-innodb-changes",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "ts2",
        "description": "circular reference in datafile path: '/home/ec2-user/dbdata/
mysql_5_7_44/../ts2.ibd'",
        "dbObjectType": "Tablespace"
      }
  ]
}
```

Si vous recevez cette erreur, reconstruisez vos tables à l'aide d'un espace de table de fichier par table. Utilisez les chemins de fichier par défaut pour tous les espaces de table et toutes les définitions de tables.

> **Note**
>
> Aurora MySQL ne prend pas en charge les espaces de table généraux ni les commandes `CREATE TABLESPACE`.
> Avant de reconstruire les espaces de table, consultez Online DDL operations dans la documentation MySQL pour comprendre les effets du verrouillage et du déplacement des données sur les transactions de premier plan.

Après la reconstruction, la vérification préalable aboutit, ce qui permet à la mise à niveau d'avoir lieu.

```
{
  "id": "circularDirectoryCheck",
  "title": "Circular directory references in tablespace data file paths",
  "status": "OK",
  "detectedProblems": []
},
```

columnsWhichCannotHaveDefaultsCheck

Niveau de vérification préalable : erreur

Colonnes ne pouvant pas avoir de valeur par défaut

Avant MySQL 8.0.13, les colonnes `BLOB`, `TEXT`, `GEOMETRY` et `JSON` ne pouvaient pas avoir de valeur par défaut. Supprimez toutes les clauses par défaut sur ces colonnes avant de procéder à la mise à niveau vers Aurora MySQL version 3. Pour plus d'informations sur les modifications apportées à la gestion par défaut de ces types de données, consultez Valeurs par défaut des types de données dans la documentation MySQL.

Exemple de sortie :

```
{
  "id": "columnsWhichCannotHaveDefaultsCheck",
  "title": "Columns which cannot have default values",
  "status": "OK",
```

```
  "description": "Error: The following columns are defined as either BLOB, TEXT,
 GEOMETRY or JSON and have a default value set. These data types cannot have default
 values in MySQL versions prior to 8.0.13, while starting with 8.0.13, the default
 value must be specified as an expression. In order to fix this issue, please use
 the ALTER TABLE ... ALTER COLUMN ... DROP DEFAULT statement.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/data-type-
defaults.html#data-type-defaults-explicit",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.test_blob_default.geo_col",
        "description": "geometry"
      }
  ]
},
```

La vérification préalable renvoie une erreur, car la colonne `geo_col` de la table `test.test_blob_default` utilise un type de données `BLOB`, `TEXT`, `GEOMETRY` ou `JSON` avec une valeur par défaut spécifiée.

En regardant la définition de la table, nous pouvons voir que la colonne `geo_col` est définie comme `geo_col geometry NOT NULL default ''`.

```
mysql> show create table test_blob_default\G
*************************** 1. row ***************************
       Table: test_blob_default
Create Table: CREATE TABLE `test_blob_default` (
  `geo_col` geometry NOT NULL DEFAULT ''
) ENGINE=InnoDB DEFAULT CHARSET=latin1
```

Supprimez cette clause par défaut pour permettre à la vérification préalable d'aboutir.

> **Note**
>
> Avant d'exécuter des instructions `ALTER TABLE` ou de reconstruire des espaces de table, consultez Opérations DDL en ligne dans la documentation MySQL pour comprendre les effets du verrouillage et du déplacement des données sur les transactions de premier plan.

```
mysql> ALTER TABLE test_blob_default modify COLUMN geo_col geometry NOT NULL;
```

```
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show create table test_blob_default\G
*************************** 1. row ***************************
       Table: test_blob_default
Create Table: CREATE TABLE `test_blob_default` (
  `geo_col` geometry NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

La vérification préalable aboutit. Vous pouvez donc réessayer la mise à niveau.

```
{
  "id": "columnsWhichCannotHaveDefaultsCheck",
  "title": "Columns which cannot have default values",
  "status": "OK",
  "detectedProblems": []
},
```

depreciatedSyntaxCheck

Niveau de vérification préalable : erreur

Utilisation de mots clés dépréciés dans la définition

MySQL 8.0 a supprimé le cache de requêtes. Par conséquent, certaines syntaxes SQL spécifiques au cache de requêtes ont été supprimées. Si l'un des objets de votre base de données contient les mots clés `QUERY CACHE`, `SQL_CACHE` ou `SQL_NO_CACHE`, une erreur de vérification préalable est renvoyée. Pour résoudre ce problème, recréez ces objets en supprimant les mots clés mentionnés.

Exemple de sortie :

```
{
  "id": "depreciatedSyntaxCheck",
  "title": "Usage of depreciated keywords in definition",
  "status": "OK",
  "description": "Error: The following DB objects contain keywords like 'QUERY
 CACHE', 'SQL_CACHE', 'SQL_NO_CACHE' which are not supported in major version 8.0.
 It is recommended to drop these DB objects or rebuild without any of the above
 keywords before upgrade.",
  "detectedProblems": [
```

```
      {
"level": "Error",
"dbObject": "test.no_query_cache_check",
"description": "PROCEDURE uses depreciated words in definition"
      }
  ]
}
```

La vérification préalable indique que la procédure stockée `test.no_query_cache_check` utilise l'un des mots clés supprimés. En regardant la définition de la procédure, nous pouvons voir qu'elle utilise `SQL_NO_CACHE`.

```
mysql> show create procedure test.no_query_cache_check\G
*************************** 1. row ***************************
           Procedure: no_query_cache_check
            sql_mode:
    Create Procedure: CREATE DEFINER=`reinvent`@`%` PROCEDURE
 `no_query_cache_check`()
BEGIN
    SELECT SQL_NO_CACHE k from sbtest1 where id > 10 and id < 20 group by k asc;
END
character_set_client: utf8mb4
collation_connection: utf8mb4_0900_ai_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Supprimez le mot clé.

```
mysql> drop procedure test.no_query_cache_check;
Query OK, 0 rows affected (0.01 sec)

mysql> delimiter //

mysql> CREATE DEFINER=`reinvent`@`%` PROCEDURE `no_query_cache_check`() BEGIN
 SELECT k from sbtest1 where id > 10 and id < 20 group by k asc; END//
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;
```

Une fois le mot clé supprimé, la vérification préalable aboutit.

```
{
```

```
  "id": "depreciatedSyntaxCheck",
  "title": "Usage of depreciated keywords in definition",
  "status": "OK",
  "detectedProblems": []
}
```

engineMixupCheck

Niveau de vérification préalable : erreur

Tables reconnues par InnoDB et qui appartiennent à un autre moteur

Semblable à schemaInconsistencyCheck, cette vérification préalable s'assure que les métadonnées des tables dans MySQL sont cohérentes avant de procéder à la mise à niveau.

Si vous rencontrez des erreurs lors de cette vérification préalable, ouvrez une demande d'assistance auprès d'AWS Support pour que l'incohérence des métadonnées soit résolue. Vous pouvez également réessayer la mise à niveau en procédant à un vidage logique, puis en effectuant une restauration sur un nouveau cluster de bases de données Aurora MySQL version 3.

Exemple de sortie :

```
{
  "id": "engineMixupCheck",
  "title": "Tables recognized by InnoDB that belong to a different engine",
  "status": "OK",
  "description": "Error: Following tables are recognized by InnoDB engine while the
 SQL layer believes they belong to a different engine. Such situation may happen
 when one removes InnoDB table files manually from the disk and creates e.g. a
 MyISAM table with the same name.\n\nA possible way to solve this situation is to
 e.g. in case of MyISAM table:\n\n1. Rename the MyISAM table to a temporary name
 (RENAME TABLE).\n2. Create some dummy InnoDB table (its definition does not need to
 match), then copy (copy, not move) and rename the dummy .frm and .ibd files to the
 orphan name using OS file commands.\n3. The orphan table can be then dropped (DROP
 TABLE), as well as the dummy table.\n4. Finally the MyISAM table can be renamed
 back to its original name.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.general_log_backup",
        "description": "recognized by the InnoDB engine but belongs to CSV"
      }
```

```
  ]
}
```

enumSetElementLengthCheck

Niveau de vérification préalable : erreur

Définitions de colonnes **`ENUM`** et **`SET`** contenant des éléments de plus de 255 caractères

Les tables et les procédures stockées ne doivent pas comporter d'éléments de colonne `ENUM` ou `SET` de plus de 255 caractères ou 1 020 octets. Avant MySQL 8.0, la longueur combinée maximale était de 64K, mais la version 8.0 limite les éléments individuels à 255 caractères ou 1 020 octets (en prenant en charge les multioctets). En cas d'échec de la vérification préalable pour `enumSetElementLengthCheck`, modifiez tous les éléments dépassant ces nouvelles limites avant de réessayer la mise à niveau.

Exemple de sortie :

```
{
  "id": "enumSetElementLengthCheck",
  "title": "ENUM/SET column definitions containing elements longer than 255
 characters",
  "status": "OK",
  "description": "Error: The following columns are defined as either ENUM or SET and
 contain at least one element longer that 255 characters. They need to be altered so
 that all elements fit into the 255 characters limit.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/string-type-
overview.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.large_set.s",
        "description": "SET contains element longer than 255 characters"
      }
  ]
},
```

La vérification préalable signale une erreur, car la colonne `s` dans la table `test.large_set` contient un élément `SET` de plus de 255 caractères.

Après avoir réduit la taille `SET` de cette colonne, la vérification préalable aboutit, ce qui permet à la mise à niveau d'avoir lieu.

```
{
  "id": "enumSetElementLenghtCheck",
  "title": "ENUM/SET column definitions containing elements longer than 255
 characters",
  "status": "OK",
  "detectedProblems": []
},
```

foreignKeyLengthCheck

Niveau de vérification préalable : erreur

Noms de contrainte de clé étrangère dépassant 64 caractères

Dans MySQL, la longueur des identifiants est limitée à 64 caractères, comme indiqué dans la documentation MySQL. En raison de problèmes identifiés où la longueur des clés étrangères pouvait être égale ou supérieure à cette valeur, entraînant des échecs de mise à niveau, cette vérification préalable a été mise en œuvre. Si vous rencontrez des erreurs lors de cette vérification préalable, vous devez modifier ou renommer votre contrainte afin qu'elle comporte moins de 64 caractères avant de réessayer la mise à niveau.

Exemple de sortie :

```
{
  "id": "foreignKeyLength",
  "title": "Foreign key constraint names longer than 64 characters",
  "status": "OK",
  "detectedProblems": []
}
```

getDuplicateTriggers

Niveau de vérification préalable : erreur

Tous les noms de déclencheurs d'une base de données doivent être uniques.

En raison des modifications apportées à l'implémentation du dictionnaire de données, MySQL 8.0 ne prend pas en charge les déclencheurs sensibles à la casse dans une base de données. Cette vérification préalable s'assure que votre cluster de bases de données ne possède pas de bases de données contenant des déclencheurs dupliqués. Pour plus d'informations, consultez Sensibilité à la casse des identifiants dans la documentation MySQL.

Exemple de sortie :

```
{
  "id": "getDuplicateTriggers",
  "title": "MySQL pre-checks that all trigger names in a database are unique or
 not.",
  "status": "OK",
  "description": "Error: You have one or more database containing duplicate
 triggers. Mysql 8.0 does not support case sensitive triggers within a database
 https://dev.mysql.com/doc/refman/8.0/en/identifier-case-sensitivity.html. To
 upgrade to MySQL 8.0, drop the triggers with case-insensitive duplicate names and
 recreate with distinct names.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test",
        "description": "before_insert_product"
      },
      {
        "level": "Error",
        "dbObject": "test",
        "description": "before_insert_PRODUCT"
      }
  ]
}
```

La vérification préalable signale une erreur indiquant que le cluster de bases de données a deux déclencheurs portant le même nom, mais dans avec une casse différente : `test.before_insert_product` et `test.before_insert_PRODUCT`.

Avant la mise à niveau, renommez les déclencheurs, ou supprimez-les et recréez-les sous un nouveau nom.

Une fois que le nom `test.before_insert_PRODUCT` est remplacé par `test.before_insert_product_2`, la vérification préalable aboutit.

```
{
  "id": "getDuplicateTriggers",
  "title": "MySQL pre-checks that all trigger names in a database are unique or
 not.",
  "status": "OK",
  "detectedProblems": []
```

```
}
```

getEventsWithNullDefiner

Niveau de vérification préalable : erreur

La colonne DEFINER pour **`mysql.event`** ne peut pas être nulle ou vide.

L'attribut `DEFINER` indique le compte MySQL qui possède une définition d'objet stockée, telle qu'un déclencheur, une procédure stockée ou un événement. Cet attribut est particulièrement utile dans les situations où vous souhaitez contrôler le contexte de sécurité dans lequel s'exécute l'objet stocké. Lorsque vous créez un objet stocké, si `DEFINER` n'est pas spécifié, sa valeur par défaut est l'utilisateur qui a créé l'objet.

Lors de la mise à niveau vers MySQL 8.0, vous ne pouvez stocker aucun objet contenant un attribut DEFINER `null` ou vide dans le dictionnaire de données MySQL. Si vous avez des objets stockés de ce type, une erreur de vérification préalable sera générée. Vous devrez la corriger avant de pouvoir procéder à la mise à niveau.

Exemple d'erreur :

```
{
  "id": "getEventsWithNullDefiner",
  "title": "The definer column for mysql.event cannot be null or blank.",
  "status": "OK",
  "description": "Error: Set definer column in mysql.event to a valid non-null
 definer.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.get_version",
        "description": "Set definer for event get_version in Schema test"
      }
  ]
}
```

La vérification préalable renvoie une erreur pour l'événement `test.get_version`, car celui-ci possède un attribut DEFINER `null`.

Pour résoudre ce problème, vous pouvez vérifier la définition de l'événement. Comme vous pouvez le constater, l'attribut DEFINER est `null` ou vide.

```
mysql> select db,name,definer from mysql.event where name='get_version';
+------+-------------+---------+
| db   | name        | definer |
+------+-------------+---------+
| test | get_version |         |
+------+-------------+---------+
1 row in set (0.00 sec)
```

Supprimez ou recréez l'événement avec un attribut DEFINER valide.

> **Note**
>
> Avant de supprimer ou de redéfinir un `DEFINER`, examinez attentivement votre demande et vérifiez les exigences en matière de privilèges. Pour plus d'informations, consultez Stored object access control dans la documentation MySQL.

```
mysql> drop event test.get_version;
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> delimiter $$
mysql> CREATE EVENT get_version
    ->     ON SCHEDULE
    ->       EVERY 1 DAY
    ->     DO
    ->      ///DO SOMETHING //
    -> $$
Query OK, 0 rows affected (0.01 sec)

mysql> DELIMITER ;

mysql> select db,name,definer from mysql.event where name='get_version';
+------+-------------+------------+
| db   | name        | definer    |
+------+-------------+------------+
| test | get_version | reinvent@% |
+------+-------------+------------+
1 row in set (0.00 sec)
```

Maintenant, la vérification préalable aboutit.

```
{
  "id": "getEventsWithNullDefiner",
  "title": "The definer column for mysql.event cannot be null or blank.",
  "status": "OK",
  "detectedProblems": []},
```

getMismatchedMetadata

Niveau de vérification préalable : erreur

Incompatibilité de définition de colonne entre le dictionnaire de données InnoDB et la définition réelle de la table

Semblable à schemaInconsistencyCheck, cette vérification préalable s'assure que les métadonnées des tables dans MySQL sont cohérentes avant de procéder à la mise à niveau. Dans ce cas, la vérification préalable s'assure que les définitions de colonnes correspondent entre le dictionnaire de données InnoDB et la définition de table MySQL. Si une incompatibilité est détectée, la mise à niveau n'a pas lieu.

Si vous rencontrez des erreurs lors de cette vérification préalable, ouvrez une demande d'assistance auprès d'AWS Support pour que l'incohérence des métadonnées soit résolue. Vous pouvez également réessayer la mise à niveau en procédant à un vidage logique, puis en effectuant une restauration sur un nouveau cluster de bases de données Aurora MySQL version 3.

Exemple de sortie :

```
{
  "id": "getMismatchedMetadata",
  "title": "Column definition mismatch between InnoDB Data Dictionary and actual
 table definition.",
  "status": "OK",
  "description": "Error: Your database has mismatched metadata. The upgrade to mysql
 8.0 will not succeed until this is fixed.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.mismatchTable",
        "description": "Table `test/mismatchTable` column names mismatch with InnoDb
 dictionary column names: iD <> id"
      }
```

```
  ]
}
```

La vérification préalable signale une incohérence des métadonnées dans la colonne `id` de la table `test.mismatchTable`. Plus précisément, les métadonnées MySQL utilisent le nom de colonne `iD`, tandis qu'InnoDB utilise le nom de colonne `id`.

getTriggersWithNullDefiner

Niveau de vérification préalable : erreur

La colonne DEFINER pour **`information_schema.triggers`** ne peut pas être **`null`** ou vide.

La vérification préalable s'assure que votre base de données ne possède aucun déclencheur défini avec un attribut DEFINER `null` ou vide. Pour plus d'informations sur les exigences relatives à l'attribut DEFINER pour les objets stockés, consultez getEventsWithNullDefiner.

Exemple de sortie :

```
{
  "id": "getTriggersWithNullDefiner",
  "title": "The definer column for information_schema.triggers cannot be null or
 blank.",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.example_trigger",
        "description": "Set definer for trigger example_trigger in Schema test"
      }
  ]
}
```

La vérification préalable renvoie une erreur, car l'attribut DEFINER du déclencheur `example_trigger` du schéma `test` est `null`. Pour corriger ce problème, corrigez cet attribut en recréant le déclencheur avec un utilisateur valide, ou supprimez le déclencheur. Pour plus d'informations, consultez l'exemple dans getEventsWithNullDefiner.

> **Note**
>
> Avant de supprimer ou de redéfinir un `DEFINER`, examinez attentivement votre demande et vérifiez les exigences en matière de privilèges. Pour plus d'informations, consultez Stored object access control dans la documentation MySQL.

getValueOfVariablelower_case_table_names

Niveau de vérification préalable : erreur

Tous les noms de base de données ou de table doivent être en minuscules lorsque le paramètre **`lower_case_table_names`** est défini sur **`1`**.

Avant MySQL 8.0, les noms de base de données, les noms de table et les autres objets correspondaient à des fichiers du répertoire de données, tels que des métadonnées basées sur des fichiers (.frm). La variable système lower_case_table_names permet aux utilisateurs de contrôler la façon dont le serveur gère la sensibilité à la casse des identifiants pour les objets de base de données et le stockage de ces objets de métadonnées. Ce paramètre pouvait être modifié sur un serveur déjà initialisé après un redémarrage.

Cependant, dans MySQL 8.0, bien que ce paramètre contrôle toujours la façon dont le serveur gère la sensibilité à la casse, il ne peut pas être modifié une fois le dictionnaire de données initialisé. Lors de la mise à niveau ou de la création d'une base de données MySQL 8.0, la valeur définie pour `lower_case_table_names` lors du premier démarrage du dictionnaire de données sur MySQL est utilisée pendant toute la durée de vie de cette base de données. Cette restriction a été mise en place dans le cadre de l'implémentation de l'Atomic Data Dictionary, où les objets de base de données sont migrés à partir de métadonnées basées sur des fichiers vers des tables InnoDB internes du schéma `mysql`.

Pour plus d'informations, consultez Modifications du dictionnaire de données dans la documentation MySQL.

Pour éviter les problèmes lors de la mise à niveau quand les métadonnées basées sur des fichiers sont migrées vers le nouveau dictionnaire Atomic Data Dictionary, cette vérification préalable permet de s'assurer que toutes les tables sont stockées sur le disque en minuscules quand `lower_case_table_names = 1`. Si ce n'est pas le cas, une erreur de vérification préalable est renvoyée et vous devrez corriger les métadonnées avant de procéder à la mise à niveau.

Exemple de sortie :

```
{
  "id": "getValueOfVariablelower_case_table_names",
  "title": "MySQL pre-checks that all database or table names are lowercase when the
 lower_case_table_names parameter is set to 1.",
  "status": "OK",
  "description": "Error: You have one or more databases or tables with uppercase
 letters in the names, but the lower_case_table_names parameter is set to 1. To
 upgrade to MySQL 8.0, either change all database or table names to lowercase, or
 set the parameter to 0.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.TEST",
        "description": "Table test.TEST contains one or more capital letters in name
 while lower_case_table_names = 1"
      }
  ]
}
```

Une erreur est renvoyée, car la table `test.TEST` contient des majuscules alors que `lower_case_table_names` est défini sur 1.

Pour résoudre ce problème, vous pouvez modifier le nom de la table pour qu'il soit en minuscules ou modifier le paramètre `lower_case_table_names` sur le cluster de bases de données avant de démarrer la mise à niveau.

> **Note**
>
> Testez et examinez attentivement la documentation sur la sensibilité à la casse dans MySQL et sur la manière dont ce type de changements pourrait affecter votre application. Consultez également la documentation de MySQL 8.0 pour déterminer les différences liées à la gestion de lower_case_table_names dans MySQL 8.0.

groupByAscSyntaxCheck

Niveau de vérification préalable : erreur

Utilisation de la syntaxe **`GROUP BY ASC/DESC`** supprimée

Depuis MySQL 8.0.13, la syntaxe ASC ou DESC obsolète pour les clauses GROUP BY a été supprimée. Les requêtes basées sur le tri GROUP BY peuvent désormais générer des résultats différents. Pour obtenir un ordre de tri spécifique, utilisez plutôt une clause ORDER BY. Si des objets utilisent cette syntaxe dans votre base de données, vous devrez les recréer à l'aide d'une clause ORDER BY avant de réessayer la mise à niveau. Pour plus d'informations, consultez Modifications SQL dans la documentation MySQL.

Exemple de sortie :

```
{
  "id": "groupbyAscSyntaxCheck",
  "title": "Usage of removed GROUP BY ASC/DESC syntax",
  "status": "OK",
  "description": "Error: The following DB objects use removed GROUP BY ASC/DESC
 syntax. They need to be altered so that ASC/DESC keyword is removed from GROUP BY
 clause and placed in appropriate ORDER BY clause.",
  "documentationLink": "https://dev.mysql.com/doc/relnotes/mysql/8.0/en/
news-8-0-13.html#mysqld-8-0-13-sql-syntax",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.groupbyasc",
        "description": "PROCEDURE uses removed GROUP BY ASC syntax",
        "dbObjectType": "Routine"
      }
  ]
}
```

mysqlEmptyDotTableSyntaxCheck

Niveau de vérification préalable : erreur

Rechercher l'utilisation de la syntaxe **.<table>** obsolète dans les routines.

Dans MySQL 8.0, les routines ne peuvent plus contenir la syntaxe d'identifiant obsolète (\".<table>\"). Si des routines ou des déclencheurs stockés contiennent ce type d'identifiants, la mise à niveau échouera. Par exemple, la référence .dot_table suivante n'est plus autorisée :

```
mysql> show create procedure incorrect_procedure\G
*************************** 1. row ***************************
           Procedure: incorrect_procedure
            sql_mode:
```

```
    Create Procedure: CREATE DEFINER=`reinvent`@`%` PROCEDURE
 `incorrect_procedure`()
BEGIN delete FROM .dot_table; select * from .dot_table where 1=1; END
character_set_client: utf8mb4
collation_connection: utf8mb4_0900_ai_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Une fois que vous avez recréé les routines et les déclencheurs afin d'utiliser la syntaxe d'identifiant et les caractères d'échappement corrects, la vérification préalable aboutit et la mise à niveau peut avoir lieu. Pour plus d'informations, consultez Noms des objets de schéma dans la documentation PostgreSQL.

Exemple de sortie :

```
{
  "id": "mysqlEmptyDotTableSyntaxCheck",
  "title": "Check for deprecated '.<table>' syntax used in routines.",
  "status": "OK",
  "description": "Error: The following routines contain identifiers in deprecated
 identifier syntax (\".<table>\"), and should be corrected before upgrade:\n",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.incorrect_procedure",
        "description": " routine body contains deprecated identifiers."
      }
  ]
}
```

La vérification préalable renvoie une erreur pour la routine `incorrect_procedure` de la base de données `test`, car elle contient une syntaxe obsolète.

Une fois que vous avez corrigé la routine, la vérification préalable aboutit et vous pouvez réessayer la mise à niveau.

mysqlIndexTooLargeCheck

Niveau de vérification préalable : erreur

Rechercher les index qui sont trop volumineux pour fonctionner sur les versions de MySQL supérieures à 5.7

Pour les formats de ligne compacts ou redondants, il ne devrait pas être possible de créer un index avec un préfixe supérieur à 767 octets. Cependant, avant la version 5.7.35 de MySQL, cela était possible. Pour plus d'informations, consultez Notes de mise à jour de MySQL 5.7.35.

Tous les index affectés par ce bogue deviendront inaccessibles après la mise à niveau vers MySQL 8.0. Cette vérification préalable identifie les index incriminés qui devront être reconstruits avant que la mise à niveau ne soit autorisée.

```
{
 "id": "mysqlIndexTooLargeCheck",
 "title": "Check for indexes that are too large to work on higher versions of MySQL
Server than 5.7",
 "status": "OK",
 "description": "Error: The following indexes ware made too large for their format
in an older version of MySQL (older than 5.7.34). Normally those indexes within
tables with compact or redundant row formats shouldn't be larger than 767 bytes. To
fix this problem those indexes should be dropped before upgrading or those tables
will be inaccessible.",
 "detectedProblems": [
     {
       "level": "Error",
       "dbObject": "test.table_with_large_idx",
       "description": "IDX_2"
     }
 ]
}
```

La vérification préalable renvoie une erreur, car la table `test.table_with_large_idx` contient un index sur une table utilisant un format de ligne compact ou redondant de plus de 767 octets. Ces tables deviendraient inaccessibles après la mise à niveau vers MySQL 8.0. Avant de procéder à la mise à niveau, effectuez l'une des actions suivantes :

- Supprimez l'index mentionné dans la vérification préalable.
- Ajoutez un index mentionné dans la vérification préalable.
- Modifiez le format de ligne utilisé par la table.

Dans ce cas, nous reconstruirons la table pour résoudre l'échec de la vérification préalable. Avant de reconstruire la table, assurez-vous qu'innodb_file_format est défini sur `Barracuda` et qu'innodb_default_row_format est défini sur `dynamic`. Ce sont les valeurs par défaut dans MySQL 5.7. Pour plus d'informations, consultez Formats de ligne InnoDB et Gestion des formats de fichier InnoDB dans la documentation MySQL.

> **Note**
>
> Avant de reconstruire les espaces de table, consultez Online DDL operations dans la documentation MySQL pour comprendre les effets du verrouillage et du déplacement des données sur les transactions de premier plan.

```
mysql > select @@innodb_file_format,@@innodb_default_row_format;
+----------------------+-----------------------------+
| @@innodb_file_format | @@innodb_default_row_format |
+----------------------+-----------------------------+
| Barracuda            | dynamic                     |
+----------------------+-----------------------------+
1 row in set (0.00 sec)

mysql> optimize table table_with_large_idx;
+---------------------------+----------+----------
+-------------------------------------------------------------------+
| Table                     | Op       | Msg_type | Msg_text
                            |
+---------------------------+----------+----------
+-------------------------------------------------------------------+
| test.table_with_large_idx | optimize | note     | Table does not support optimize,
 doing recreate + analyze instead |
| test.table_with_large_idx | optimize | status   | OK
                            |
+---------------------------+----------+----------
+-------------------------------------------------------------------+
2 rows in set (0.02 sec)

# Verify FILE_FORMAT and ROW_FORMAT
mysql>  select * from information_schema.innodb_sys_tables where name like 'test/
table_with_large_idx';
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
| TABLE_ID | NAME                      | FLAG | N_COLS | SPACE | FILE_FORMAT |
 ROW_FORMAT | ZIP_PAGE_SIZE | SPACE_TYPE |
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
|       43 | test/table_with_large_idx |   33 |      4 |    26 | Barracuda   |
 Dynamic    |             0 | Single     |
```

```
+----------+---------------------------+------+--------+-------+-------------
+------------+---------------+------------+
1 row in set (0.00 sec)
```

Après avoir reconstruit la table, la vérification préalable aboutit et la mise à niveau peut avoir lieu.

```
{
  "id": "mysqlIndexTooLargeCheck",
  "title": "Check for indexes that are too large to work on higher versions of MySQL
 Server than 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlInvalid57NamesCheck

Niveau de vérification préalable : erreur

Rechercher les noms de table et de schéma non valides utilisés dans MySQL 5.7

Après la migration vers le nouveau dictionnaire de données dans MySQL 8.0, votre instance de base de données ne peut plus contenir de schémas ou de tables avec le préfixe `#mysql50#`. Si de tels objets existent, la mise à niveau échouera. Pour résoudre ce problème, exécutez mysqlcheck sur les schémas et les tables renvoyés.

> **Note**
>
> Assurez-vous d'utiliser une version MySQL 5.7 de `mysqlcheck`, car --fix-db-names et --fix-table-names ont été supprimés de MySQL 8.0.

Exemple de sortie :

```
{
  "id": "mysqlInvalid57NamesCheck",
  "title": "Check for invalid table names and schema names used in 5.7",
  "status": "OK",
  "description": "The following tables and/or schemas have invalid names. In order
 to fix them use the mysqlcheck utility as follows:\n\n  $ mysqlcheck --check-
upgrade --all-databases\n  $ mysqlcheck --fix-db-names --fix-table-names --all-
databases\n\nOR via mysql client, for eg:\n\n  ALTER DATABASE `#mysql50#lost+found`
 UPGRADE DATA DIRECTORY NAME;",
```

```
  "documentationLink": "https://dev.mysql.com/doc/refman/5.7/en/identifier-
mapping.html https://dev.mysql.com/doc/refman/5.7/en/alter-database.html https://
dev.mysql.com/doc/refman/8.0/en/mysql-nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "#mysql50#fix_db_names",
        "description": "Schema name"
      }
  ]
}
```

La vérification préalable indique que le schéma `#mysql50#fix_db_names` n'est pas valide.

Après avoir corrigé le nom du schéma, la vérification préalable aboutit, ce qui permet à la mise à niveau de se poursuivre.

```
{
  "id": "mysqlInvalid57NamesCheck",
  "title": "Check for invalid table names and schema names used in 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlOrphanedRoutinesCheck

Niveau de vérification préalable : erreur

Rechercher les routines orphelines dans la version 5.7

Lors de la migration vers le nouveau dictionnaire de données dans MySQL 8.0, s'il existe des procédures stockées dans la base de données où le schéma n'existe plus, la mise à niveau échoue. Cette vérification préalable s'assure que tous les schémas référencés dans les procédures stockées de votre instance de base de données existent toujours. Pour permettre à la mise à niveau de se poursuivre, supprimez ces procédures stockées.

Exemple de sortie :

```
{
  "id": "mysqlOrphanedRoutinesCheck",
  "title": "Check for orphaned routines in 5.7",
  "status": "OK",
```

```
  "description": "Error: The following routines have been orphaned. Schemas that
 they are referencing no longer exists.\nThey have to be cleaned up or the upgrade
 will fail.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "dropped_db.get_version",
        "description": "is orphaned"
      }
  ]
},
```

La vérification préalable indique que la procédure stockée `get_version` dans la base de données `dropped_db` est orpheline.

Pour nettoyer cette procédure, vous pouvez recréer le schéma manquant.

```
mysql> create database dropped_db;
Query OK, 1 row affected (0.01 sec)
```

Une fois le schéma recréé, vous pouvez supprimer la procédure pour permettre à la mise à niveau de se poursuivre.

```
{
  "id": "mysqlOrphanedRoutinesCheck",
  "title": "Check for orphaned routines in 5.7",
  "status": "OK",
  "detectedProblems": []
},
```

mysqlSchemaCheck

Niveau de vérification préalable : erreur

Les noms des tables dans le schéma **`mysql`** sont en conflit avec les nouvelles tables de MySQL 8.0

Le nouveau dictionnaire Atomic Data Dictionary introduit dans MySQL 8.0 stocke toutes les métadonnées dans un ensemble de tables InnoDB internes du schéma `mysql`. Au cours de la mise à niveau, les nouvelles tables du dictionnaire de données interne sont créées dans le schéma `mysql`. Pour éviter les conflits de noms, qui pourraient entraîner des échecs de mise à niveau, la vérification préalable examine tous les noms de table du schéma `mysql` afin de

s'assurer qu'aucun des nouveaux noms de table n'est déjà utilisé. Si tel est le cas, une erreur est renvoyée et la mise à niveau n'est pas autorisée à se poursuivre.

Exemple de sortie :

```
{
  "id": "mysqlSchema",
  "title": "Table names in the mysql schema conflicting with new tables in the
 latest MySQL.",
  "status": "OK",
  "description": "Error: The following tables in mysql schema have names that will
 conflict with the ones introduced in the latest version. They must be renamed or
 removed before upgrading (use RENAME TABLE command). This may also entail changes
 to applications that use the affected tables.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrade-before-you-
begin.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.tablespaces",
        "description": "Table name used in mysql schema.",
        "dbObjectType": "Table"
      }
  ]
}
```

Une erreur est renvoyée, car une table est nommée `tablespaces` dans le schéma `mysql`. Il s'agit de l'un des nouveaux noms de tables du dictionnaire de données interne de MySQL 8.0. Vous devrez renommer ou supprimer ces tables avant de procéder à la mise à niveau, à l'aide de la commande `RENAME TABLE`.

nonNativePartitioningCheck

Niveau de vérification préalable : erreur

Tables partitionnées à l'aide de moteurs avec partitionnement non natif

Selon la documentation MySQL 8.0, deux moteurs de stockage prennent actuellement en charge le partitionnement natif : InnoDB et NDB. Parmi eux, seul InnoDB est pris en charge dans Aurora MySQL version 3, compatible avec MySQL 8.0. Toute tentative de création de tables partitionnées dans MySQL 8.0 à l'aide d'un autre moteur de stockage échouera. Cette vérification préalable recherche les tables de votre cluster de bases de données qui utilisent un partitionnement non

natif. Le cas échéant, vous devrez supprimer le partitionnement ou remplacer le moteur de stockage par InnoDB.

Exemple de sortie :

```
{
  "id": "nonNativePartitioning",
  "title": "Partitioned tables using engines with non native partitioning",
  "status": "OK",
  "description": "Error: In the latest MySQL storage engine is responsible for
 providing its own partitioning handler, and the MySQL server no longer provides
 generic partitioning support. InnoDB and NDB are the only storage engines that
 provide a native partitioning handler that is supported in the latest MySQL. A
 partitioned table using any other storage engine must be altered—either to convert
 it to InnoDB or NDB, or to remove its partitioning—before upgrading the server,
 else it cannot be used afterwards.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-configuration-changes",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.partMyisamTable",
         "description": "MyISAM engine does not support native partitioning",
         "dbObjectType": "Table"
      }
  ]
}
```

Ici, une table MyISAM utilise le partitionnement. Une action est donc nécessaire avant que la mise à niveau puisse commencer.

oldTemporalCheck

Niveau de vérification préalable : erreur

Utilisation du type « anciens temporels »

Les « anciens temporels » font référence aux colonnes de type temporel (comme `TIMESTAMP` et `DATETIME`) créées dans les versions 5.5 et antérieures de MySQL. Dans MySQL 8.0, la prise en charge de ces types de données « anciens temporels » est supprimée, ce qui signifie que les mises à niveau sur place de MySQL 5.7 à 8.0 ne sont pas possibles si la base de données en contient encore. Pour résoudre ce problème, vous devez reconstruire toutes les tables contenant ces types de données « anciens temporels » avant de procéder à la mise à niveau.

Pour plus d'informations sur la dépréciation des types de données « anciens temporels » dans MySQL 5.7, consultez ce blog. Pour plus d'informations sur la suppression des types de données « anciens temporels » dans MySQL 8.0, consultez ce blog.

> **Note**
>
> Avant de reconstruire les espaces de table, consultez Online DDL operations dans la documentation MySQL pour comprendre les effets du verrouillage et du déplacement des données sur les transactions de premier plan.

Exemple de sortie :

```
{
  "id": "oldTemporalCheck",
  "title": "Usage of old temporal type",
  "status": "OK",
  "description": "Error: Following table columns use a deprecated and no longer
 supported temporal disk storage format. They must be converted to the new format
 before upgrading. It can by done by rebuilding the table using 'ALTER TABLE
 <table_name> FORCE' command",
  "documentationLink": "https://dev.mysql.com/blog-archive/mysql-8-0-removing-
support-for-old-temporal-datatypes/",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.55_temporal_table.timestamp_column",
        "description": "timestamp /* 5.5 binary format */",
        "dbObjectType": "Column"
      }
  ]
},
```

Une erreur est signalée pour la colonne `timestamp_column` de la table `test.55_temporal_table`, car elle utilise un format de stockage sur disque ancien temporel qui n'est plus pris en charge.

Pour résoudre ce problème et permettre la mise à niveau, reconstruisez la table pour convertir le format de stockage sur disque ancien temporel vers le nouveau format introduit dans MySQL 5.6. Pour plus d'informations et pour connaître les conditions préalables, consultez Conversion entre les jeux de caractères Unicode à 3 octets et à 4 octets dans la documentation MySQL.

L'exécution de la commande suivante pour reconstruire les tables mentionnées dans cette vérification préalable convertit les types de données « anciens temporels » au nouveau format avec une précision d'une fraction de seconde.

```
ALTER TABLE ... ENGINE=InnoDB;
```

Pour plus d'informations sur la reconstruction des tables, consultez Instruction ALTER TABLE dans la documentation MySQL.

Après avoir reconstruit la table en question et redémarré la mise à niveau, la vérification de la compatibilité aboutit, ce qui permet à la mise à niveau de se poursuivre.

```
{
  "id": "oldTemporalCheck",
  "title": "Usage of old temporal type",
  "status": "OK",
  "detectedProblems": []
}
```

partitionedTablesInSharedTablespaceCheck

Niveau de vérification préalable : erreur

Utilisation de tables partitionnées dans les espaces de table partagés

Depuis MySQL 8.0.13, la prise en charge du placement des partitions de table dans les espaces de table partagés est supprimée. Avant la mise à niveau, déplacez ces tables des espaces de table partagés vers des espaces de table de fichier par table.

> **Note**
>
> Avant de reconstruire les espaces de table, consultez Partitioning operations dans la documentation MySQL pour comprendre les effets du verrouillage et du déplacement des données sur les transactions de premier plan.

Exemple de sortie :

```
{
  "id": "partitionedTablesInSharedTablespaceCheck",
  "title": "Usage of partitioned tables in shared tablespaces",
```

```
  "status": "OK",
  "description": "Error: The following tables have partitions in shared tablespaces.
 They need to be moved to file-per-table tablespace before upgrading. You can do
 this by running query like 'ALTER TABLE table_name REORGANIZE PARTITION X INTO
 (PARTITION X VALUES LESS THAN (30) TABLESPACE=innodb_file_per_table);'",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.partInnoDBTable",
        "description": "Partition p1 is in shared tablespace innodb",
        "dbObjectType": "Table"
      }
  ]
}
```

La vérification préalable échoue, car la partition p1 de la table `test.partInnoDBTable` se trouve dans l'espace de table du système.

Pour résoudre ce problème, reconstruisez la table `test.partInnodbTable` en plaçant la partition p1 en cause dans un espace de table de fichier par table.

```
mysql > ALTER TABLE partInnodbTable REORGANIZE PARTITION p1
    ->   INTO (PARTITION p1 VALUES LESS THAN ('2014-01-01')
 TABLESPACE=innodb_file_per_table);
Query OK, 0 rows affected, 1 warning (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Après cela, la vérification préalable aboutira.

```
{
  "id": "partitionedTablesInSharedTablespaceCheck",
  "title": "Usage of partitioned tables in shared tablespaces",
  "status": "OK",
  "detectedProblems": []
}
```

removedFunctionsCheck

Niveau de vérification préalable : erreur

Utilisation de fonctions supprimées de la dernière version de MySQL

Dans MySQL 8.0, plusieurs fonctions intégrées ont été supprimées. Cette vérification préalable examine votre base de données à la recherche d'objets susceptibles d'utiliser ces fonctions. Si elle en trouve, une erreur est renvoyée. Vous devrez résoudre ces problèmes avant de réessayer la mise à niveau.

La majorité des fonctions supprimées sont des fonctions spatiales, qui ont été remplacées par des fonctions ST_* équivalentes. Dans ce cas, modifiez les objets de base de données pour qu'ils utilisent le nom de la nouvelle procédure. Pour plus d'informations, consultez Fonctionnalités supprimées dans MySQL 8.0 dans la documentation MySQL.

Exemple de sortie :

```
{
  "id": "removedFunctionsCheck",
  "title": "Usage of removed functions",
  "status": "OK",
  "description": "Error: The following DB objects use functions that were removed
 in the latest MySQL version. Please make sure to update them to use supported
 alternatives before upgrade.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.GetLocationsInPolygon",
        "description": "PROCEDURE uses removed function POLYGONFROMTEXT (consider
 using ST_POLYGONFROMTEXT instead)",
        "dbObjectType": "Routine"
      },
      {
        "level": "Error",
        "dbObject": "test.InsertLocation",
        "description": "PROCEDURE uses removed function POINTFROMTEXT (consider
 using ST_POINTFROMTEXT instead)",
        "dbObjectType": "Routine"
      }
  ]
},
```

La vérification préalable indique que la procédure stockée `test.GetLocationsInPolygon` utilise deux fonctions supprimées : POLYGONFROMTEXT et POINTFROMTEXT. Elle suggère également d'utiliser les nouvelles fonctions ST_POLYGONFROMTEXT et ST_POINTFROMTEXT

à la place. Après avoir recréé la procédure à l'aide des suggestions, la vérification préalable aboutit.

```
{
  "id": "removedFunctionsCheck",
  "title": "Usage of removed functions",
  "status": "OK",
  "detectedProblems": []
},
```

> **Note**
>
> Bien que, dans la plupart des cas, les fonctions obsolètes soient directement remplacées, assurez-vous de tester votre application et de consulter la documentation pour détecter tout changement de comportement résultant de ce changement.

routineSyntaxCheck

Niveau de vérification préalable : erreur

Recherche d'objets de type routine dans la syntaxe MySQL

MySQL 8.0 a introduit des mots clés réservés qui ne l'étaient pas auparavant. La vérification préalable à la mise à niveau évalue l'utilisation des mots clés réservés dans les noms des objets de base de données, dans leurs définitions et dans leur corps. Si elle détecte des mots clés réservés utilisés dans des objets de base de données, tels que des procédures stockées, des fonctions, des événements et des déclencheurs, la mise à niveau échoue et une erreur est publiée dans le fichier `upgrade-prechecks.log`. Pour résoudre ce problème, vous devez mettre à jour ces définitions d'objets et placer ces références entre guillemets simples (') avant de procéder à la mise à niveau. Pour plus d'informations sur l'échappement des mots réservés dans MySQL, consultez Littéraux de chaîne dans la documentation MySQL.

Vous pouvez également remplacer le nom par un autre, ce qui peut impliquer des modifications de l'application.

Exemple de sortie :

```
{
  "id": "routineSyntaxCheck",
  "title": "MySQL syntax check for routine-like objects",
```

```
  "status": "OK",
  "description": "The following objects did not pass a syntax check with the latest
 MySQL grammar. A common reason is that they reference names that conflict with new
 reserved keywords. You must update these routine definitions and `quote` any such
 references before upgrading.",
  "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
  "detectedProblems": [
      {
         "level": "Error",
         "dbObject": "test.select_res_word",
         "description": "at line 2,18: unexpected token 'except'",
         "dbObjectType": "Routine"
      }
  ]
}
```

Pour résoudre ce problème, vérifiez la définition de la routine.

```
SHOW CREATE PROCEDURE test.select_res_word\G

*************************** 1. row ***************************
           Procedure: select_res_word
            sql_mode:
 ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_Z
    Create Procedure: CREATE PROCEDURE 'select_res_word'()
BEGIN
    SELECT * FROM except;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

La procédure utilise une table nommée `except`, qui est un nouveau mot clé dans MySQL 8.0. Recréez la procédure en échappant le littéral de la chaîne.

```
> drop procedure if exists select_res_word;
Query OK, 0 rows affected (0.00 sec)

> DELIMITER $$
 > CREATE PROCEDURE select_res_word()
    -> BEGIN
    ->     SELECT * FROM 'except';
```

```
    -> END$$
Query OK, 0 rows affected (0.00 sec)

 > DELIMITER ;
```

Désormais, la vérification préalable aboutit.

```
{
  "id": "routineSyntaxCheck",
  "title": "MySQL syntax check for routine-like objects",
  "status": "OK",
  "detectedProblems": []
}
```

schemaInconsistencyCheck

Niveau de vérification préalable : erreur

Incohérences de schéma résultant de la suppression ou de l'endommagement de fichiers

Comme décrit précédemment, MySQL 8.0 a introduit l'Atomic Data Dictionary, qui stocke toutes les métadonnées dans un ensemble de tables InnoDB internes du schéma `mysql`. Cette nouvelle architecture fournit une méthode transactionnelle conforme à l'ACID pour gérer les métadonnées des bases de données, résolvant ainsi le problème de « DDL atomique » rencontré par l'ancienne approche basée sur les fichiers. Avant MySQL 8.0, les objets de schéma pouvaient devenir orphelins en cas d'interruption inattendue d'une opération DDL. La migration des métadonnées basées sur des fichiers vers les nouvelles tables de l'Atomic Data Dictionary lors de la mise à niveau garantit l'absence d'objets de schéma orphelins de ce type dans l'instance de base de données. Si des objets orphelins sont détectés, la mise à niveau échoue.

Pour aider à détecter ces objets orphelins avant de lancer la mise à niveau, la vérification préalable `schemaInconsistencyCheck` est exécutée pour s'assurer que tous les objets de métadonnées du dictionnaire de données sont synchronisés. Si des objets de métadonnées orphelins sont détectés, la mise à niveau ne se poursuit pas. Pour procéder à la mise à niveau, nettoyez ces objets de métadonnées orphelins.

Si vous rencontrez des erreurs lors de cette vérification préalable, ouvrez une demande d'assistance auprès d'AWS Support pour que l'incohérence des métadonnées soit résolue. Vous pouvez également réessayer la mise à niveau en procédant à un vidage logique, puis en effectuant une restauration sur un nouveau cluster de bases de données Aurora MySQL version 3.

Exemple de sortie :

```
{
  "id": "schemaInconsistencyCheck",
  "title": "Schema inconsistencies resulting from file removal or corruption",
  "status": "OK",
  "description": "Error: Following tables show signs that either table datadir
 directory or frm file was removed/corrupted. Please check server logs, examine
 datadir to detect the issue and fix it before upgrade",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.schemaInconsistencyCheck_failure",
        "description": "present in INFORMATION_SCHEMA's INNODB_SYS_TABLES table but
 missing from TABLES table"
      }
  ]
}
```

La vérification préalable indique que les métadonnées de la table `test.schemaInconsistencyCheck_failure` ne sont pas cohérentes. Dans ce cas, la table existe dans les métadonnées du moteur de stockage InnoDB (`information_schema.INNODB_SYS_TABLES`), mais pas dans les métadonnées MySQL (`information_schema.TABLES`).

Vérifications préalables d'Aurora MySQL qui signalent des erreurs

Les vérifications préalables suivantes sont spécifiques à Aurora MySQL :

- auroraCheckDDLRecovery
- auroraCheckRdsUpgradePrechecksTable
- auroraFODUpgradeCheck
- auroraGetDanglingFulltextIndex
- auroraUpgradeCheckForDatafilePathInconsistency
- auroraUpgradeCheckForFtsSpaceIdZero
- auroraUpgradeCheckForIncompleteXATransactions
- auroraUpgradeCheckForInstanceLimit
- auroraUpgradeCheckForInternalUsers

- auroraUpgradeCheckForInvalidUtf8mb3CharacterStringInViews
- auroraUpgradeCheckForInvalidUtf8mb3ColumnComments
- auroraUpgradeCheckForInvalidUtf8mb3IndexComments
- auroraUpgradeCheckForInvalidUtf8mb3TableComments
- auroraUpgradeCheckForMasterUser
- auroraUpgradeCheckForPrefixIndexOnGeometryColumns
- auroraUpgradeCheckForSpecialCharactersInProcedures
- auroraUpgradeCheckForSysSchemaObjectTypeMismatch
- auroraUpgradeCheckForViewColumnNameLength
- auroraUpgradeCheckIndexLengthLimitOnTinyColumns
- auroraUpgradeCheckMissingInnodbMetadataForMysqlHostTable

auroraCheckDDLRecovery

Niveau de vérification préalable : erreur

Rechercher la présence d'artefacts liés à la fonctionnalité de restauration DDL d'Aurora

Dans le cadre de l'implémentation de la restauration DDL (Data Definition Language) dans Aurora MySQL, les métadonnées des instructions DDL en transit sont conservées dans les tables `ddl_log_md_table` et `ddl_log_table` du schéma `mysql`. L'implémentation de la restauration DDL par Aurora n'est pas prise en charge à partir de la version 3, car cette fonctionnalité fait partie de l'implémentation du nouvel Atomic Data Dictionary dans MySQL 8.0. Si des instructions DDL sont en cours d'exécution pendant les vérifications de la compatibilité, cette vérification préalable risque d'échouer. Nous vous recommandons d'essayer la mise à niveau quand aucune instruction DDL n'est en cours d'exécution.

Si cette vérification préalable échoue sans qu'aucune instruction DDL ne soit exécutée, ouvrez une demande d'assistance auprès d'AWS Support pour que l'incohérence des métadonnées soit résolue. Vous pouvez également réessayer la mise à niveau en procédant à un vidage logique, puis en effectuant une restauration sur un nouveau cluster de bases de données Aurora MySQL version 3.

Si des instructions DDL sont en cours d'exécution, la sortie de la vérification préalable affiche le message suivant :

```
“There are DDL statements in process. Please allow DDL statements to finish before
 upgrading.”
```

Exemple de sortie :

```
{
  "id": "auroraCheckDDLRecovery",
  "title": "Check for artifacts related to Aurora DDL recovery feature",
  "status": "OK",
  "description": "Aurora implementation of DDL recovery is not supported from 3.x
 onwards. This check verifies that the database do not have artifacts realted to the
 feature",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "mysql.ddl_log_md_table",
        "description": "Table mysql.ddl_log_md_table is not empty. Your database has
 pending DDL recovery operations. Reachout to AWS support for assistance."
      },
      {
        "level": "Error",
        "dbObject": "mysql.ddl_log_table",
        "description": "Table mysql.ddl_log_table is not empty. Your database has
 pending DDL recovery operations. Reachout to AWS support for assistance."
      },
      {
        "level": "Error",
        "dbObject": "information_schema.processlist",
        "description": "There are DDL statements in process. Please allow DDL
 statements to finish before upgrading."
      }
  ]
}
```

La vérification préalable a renvoyé une erreur en raison d’une instruction DDL en transit exécutée en même temps que les vérifications de la compatibilité. Nous vous recommandons de réessayer la mise à niveau sans qu’aucune instruction DDL ne soit active.

auroraCheckRdsUpgradePrechecksTable

Niveau de vérification préalable : erreur

Vérifier l’existence de la table **`mysql.rds_upgrade_prechecks`**

Il s'agit d'une vérification préalable interne, qui est effectuée par le service RDS. Toutes les erreurs seront traitées automatiquement lors de la mise à niveau et peuvent être ignorées sans risque.

Si vous rencontrez des erreurs lors de cette vérification préalable, ouvrez une demande d'assistance auprès d'AWS Support pour que l'incohérence des métadonnées soit résolue. Vous pouvez également réessayer la mise à niveau en procédant à un vidage logique, puis en effectuant une restauration sur un nouveau cluster de bases de données Aurora MySQL version 3.

```
{
  "id": "auroraCheckRdsUpgradePrechecksTable",
  "title": "Check existence of mysql.rds_upgrade_prechecks table",
  "status": "OK",
  "detectedProblems": []
}
```

auroraFODUpgradeCheck

Niveau de vérification préalable : erreur

Rechercher la présence d'artefacts liés à la fonctionnalité DDL rapide d'Aurora

L'optimisation Fast DDL a été introduite en mode Lab sur Aurora MySQL version 2 pour améliorer l'efficacité de certaines opérations DDL. Dans la version 3 d'Aurora MySQL, le mode lab a été supprimé, et l'implémentation de Fast DDL a été remplacée par la fonctionnalité de MySQL 8.0 appelée Instant DDL.

Avant la mise à niveau vers Aurora MySQL version 3, toutes les tables utilisant Fast DDL en mode Lab doivent être reconstruites en exécutant la commande `OPTIMIZE TABLE` ou `ALTER TABLE ... ENGINE=InnoDB` pour assurer la compatibilité avec Aurora MySQL version 3.

Cette vérification préalable renvoie la liste de ces tables. Une fois que les tables renvoyées ont été reconstruites, vous pouvez réessayer la mise à niveau.

Exemple de sortie :

```
{
  "id": "auroraFODUpgradeCheck",
  "title": "Check for artifacts related to Aurora fast DDL feature",
```

```
  "status": "OK",
  "description": "Aurora fast DDL is not supported from 3.x onwards. This check
 verifies that the database does not have artifacts related to the feature",
  "documentationLink": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/AuroraMySQL.Managing.FastDDL.html#AuroraMySQL.Managing.FastDDL-v2",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.test",
        "description": "Your table has pending Aurora fast DDL operations. Run
 'OPTIMIZE TABLE <table name>' for the table to apply all the pending DDL updates.
 Then try the upgrade again."
      }
  ]
}
```

La vérification préalable indique que la table `test.test` contient des opérations Fast DDL en attente.

Pour permettre la mise à niveau, vous pouvez reconstruire la table, puis réessayer la mise à niveau.

> **Note**
>
> Avant de reconstruire les espaces de table, consultez Online DDL operations dans la documentation MySQL pour comprendre les effets du verrouillage et du déplacement des données sur les transactions de premier plan.

```
mysql> optimize table test.test;
+-----------+----------+----------
+-------------------------------------------------------------------+
| Table     | Op       | Msg_type | Msg_text
                  |
+-----------+----------+----------
+-------------------------------------------------------------------+
| test.test | optimize | note     | Table does not support optimize, doing recreate
 + analyze instead |
| test.test | optimize | status   | OK
                  |
+-----------+----------+----------
+-------------------------------------------------------------------+
```

```
2 rows in set (0.04 sec)
```

Après avoir reconstruit la table, la vérification préalable aboutit.

```
{
  "id": "auroraFODUpgradeCheck",
  "title": "Check for artifacts related to Aurora fast DDL feature",
  "status": "OK",
  "detectedProblems": []
}
```

auroraGetDanglingFulltextIndex

Niveau de vérification préalable : erreur

Tables avec référence d'index **FULLTEXT** suspendue

Avant MySQL 5.6.26, il était possible qu'après la suppression d'un index de recherche en texte intégral, les colonnes `FTS_DOC_ID` et `FTS_DOC_ID_INDEX` masquées deviennent orphelines. Pour plus d'informations, consultez le bogue n° 76 012.

Si cela s'est produit dans des tables créées sur des versions antérieures de MySQL, les mises à niveau vers la version 3 d'Aurora MySQL peuvent échouer. Cette vérification préalable s'assure qu'aucun index de texte intégral orphelin ou « suspendu » n'existe sur votre cluster de bases de données avant la mise à niveau vers MySQL 8.0. Si cette vérification préalable échoue, reconstruisez toutes les tables contenant des index de texte intégral suspendus.

Exemple de sortie :

```
{
  "id": "auroraGetDanglingFulltextIndex",
  "title": "Tables with dangling FULLTEXT index reference",
  "status": "OK",
  "description": "Error: The following tables contain dangling FULLTEXT index which
 is not supported. It is recommended to rebuild the table before upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.table_with_fts_index",
        "description": "Table `test.table_with_fts_index` contains dangling FULLTEXT
 index. Kindly recreate the table before upgrade."
```

```
      }
  ]
},
```

La vérification préalable signale une erreur pour la table `test.table_with_fts_index`, car elle contient un index de texte intégral suspendu. Pour permettre la mise à niveau, reconstruisez la table pour nettoyer les tables auxiliaires d'index de texte intégral. Utiliser `OPTIMIZE TABLE test.table_with_fts_index` ou `ALTER TABLE test.table_with_fts_index, ENGINE=INNODB`.

Après avoir reconstruit la table, la vérification préalable aboutit.

```
{
  "id": "auroraGetDanglingFulltextIndex",
  "title": "Tables with dangling FULLTEXT index reference",
  "status": "OK",
  "detectedProblems": []
},
```

auroraUpgradeCheckForDatafilePathInconsistency

Niveau de vérification préalable : erreur

Rechercher les incohérences liées au chemin du fichier **`ibd`**

Cette vérification préalable ne s'applique qu'à Aurora MySQL 3.03.0 ou version antérieure. Si vous rencontrez une erreur lors de cette vérification préalable, procédez à une mise à niveau vers Aurora MySQL 3.04 ou version ultérieure.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForDatafilePathInconsistency",
  "title": "Check for inconsistency related to ibd file path.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForFtsSpaceIdZero

Niveau de vérification préalable : erreur

Rechercher l'index de texte intégral avec un identifiant d'espace égal à zéro

Dans MySQL, lorsque vous ajoutez un index de texte intégral à une table InnoDB, plusieurs espaces de table d'index auxiliaires sont créés. En raison d'un bogue dans les versions antérieures de MySQL, corrigé dans la version 5.6.20, il était possible que ces tables d'index auxiliaires soient créées dans l'espace de table du système, plutôt que dans leur propre espace de table InnoDB.

S'il existe de tels espaces de table auxiliaires, la mise à niveau échouera. Recréez les index de texte intégral mentionnés dans l'erreur générée par la vérification préalable, puis réessayez la mise à niveau.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForFtsSpaceIdZero",
  "title": "Check for fulltext index with space id as zero",
  "status": "OK",
  "description": "The auxiliary tables of FTS indexes on the table are created
 in system table-space. Due to this DDL queries executed on MySQL8.0 shall cause
 database unavailability. To avoid that, drop and recreate all the FTS indexes on
 the table or rebuild the table using ALTER TABLE query before the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.fts_space_zero_check",
        "description": " The auxiliary tables of FTS indexes on the table
 'test.fts_space_zero_check' are created in system table-space due to https://
bugs.mysql.com/bug.php?id=72132. In MySQL8.0, DDL queries executed on this table
 shall cause database unavailability. To avoid that, drop and recreate all the
 FTS indexes on the table or rebuild the table using ALTER TABLE query before the
 upgrade."
      }
  ]
},
```

La vérification préalable signale une erreur pour la table `test.fts_space_zero_check`, car elle contient des tables auxiliaires de recherche de texte intégral (FTS) dans l'espace de table du système.

Une fois que vous avez supprimé et recréé les index FTS associés à cette table, la vérification préalable aboutit.

> **Note**
>
> Avant de reconstruire les espaces de table, consultez Partitioning operations dans la documentation MySQL pour comprendre les effets du verrouillage et du déplacement des données sur les transactions de premier plan.

```
{
 "id": "auroraUpgradeCheckForFtsSpaceIdZero",
 "title": "Check for fulltext index with space id as zero",
 "status": "OK",
 "detectedProblems": []
}
```

auroraUpgradeCheckForIncompleteXATransactions

Niveau de vérification préalable : erreur

Rechercher les transactions XA à l'état préparé

Lors de l'exécution du processus de mise à niveau de la version majeure, il est essentiel que l'instance de base de données Aurora MySQL version 2 soit complètement arrêtée. Cela garantit que toutes les transactions sont validées ou annulées, et qu'InnoDB a purgé tous les enregistrements du journal d'annulation. L'annulation des transactions étant nécessaire, si votre base de données contient des transactions XA à l'état préparé, cela peut empêcher l'arrêt complet d'avoir lieu. Pour cette raison, si des transactions XA préparées sont détectées, la mise à niveau ne pourra pas avoir lieu tant que vous n'aurez pas pris les mesures nécessaires pour les valider ou les annuler.

Pour plus d'informations sur la recherche des transactions XA à l'état préparé à l'aide de `XA RECOVER`, consultez Instructions SQL des transactions XA dans la documentation MySQL. Pour plus d'informations, consultez États des transactions XA dans la documentation MySQL.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForIncompleteXATransactions",
  "title": "Pre-checks for XA Transactions in prepared state.",
  "status": "OK",
```

```
  "description": "Your cluster currently has XA transactions in the prepared state.
 To proceed with the upgrade, commit or rollback these transactions.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Your cluster currently has XA transactions in the prepared
 state. To proceed with the upgrade, commit or rollback these transactions."
      }
  ]
}
```

Cette vérification préalable signale une erreur, car certaines transactions préparées doivent être validées ou annulées.

Une fois connecté à la base de données, vous pouvez consulter la table information_schema.innodb_trx et la sortie `XA RECOVER` pour plus d'informations.

> **Important**
>
> Avant de valider ou d'annuler une transaction, nous vous recommandons de vérifier la documentation MySQL et les exigences de votre application.

```
mysql> select trx_started,
    trx_mysql_thread_id,
    trx_id,trx_state,
    trx_operation_state,
    trx_rows_modified,
    trx_rows_locked
from
    information_schema.innodb_trx;
+---------------------+--------------------+---------+-----------
+---------------------+------------------+-----------------+
| trx_started         | trx_mysql_thread_id | trx_id  | trx_state |
 trx_operation_state | trx_rows_modified | trx_rows_locked |
+---------------------+--------------------+---------+-----------
+---------------------+------------------+-----------------+
| 2024-08-12 01:09:39 |                  0 | 2849470 | RUNNING   | NULL
    |                1 |               0 |
+---------------------+--------------------+---------+-----------
+---------------------+------------------+-----------------+
```

```
1 row in set (0.00 sec)

mysql> xa recover;
+----------+--------------+--------------+--------+
| formatID | gtrid_length | bqual_length | data   |
+----------+--------------+--------------+--------+
|        1 |            6 |            0 | xatest |
+----------+--------------+--------------+--------+
1 row in set (0.00 sec)
```

Dans ce cas, nous annulons la transaction préparée.

```
mysql> XA ROLLBACK 'xatest';
Query OK, 0 rows affected (0.00 sec)
v
mysql> xa recover;
Empty set (0.00 sec)
```

Une fois que la transaction XA est annulée, la vérification préalable aboutit.

```
{
  "id": "auroraUpgradeCheckForIncompleteXATransactions",
  "title": "Pre-checks for XA Transactions in prepared state.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForInstanceLimit

Niveau de vérification préalable : erreur

Vérifier si la mise à niveau est prise en charge sur la classe d'instance actuelle

L'exécution d'une mise à niveau sur place à partir d'Aurora MySQL version 2.12.0 ou 2.12.1, où la classe d'instance de base de données d'enregistreur est db.r6i.32xlarge, n'est actuellement pas prise en charge. Dans ce cas, la vérification préalable renvoie une erreur. Pour autoriser la mise à niveau, vous pouvez remplacer votre classe d'instance de base de données par db.r6i.24xlarge ou par une classe inférieure. Vous pouvez également effectuer une mise à niveau vers Aurora MySQL version 2.12.2 ou supérieure, où la mise à niveau sur place vers Aurora MySQL version 3 est prise en charge sur db.r6i.32xlarge.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForInstanceLimit",
  "title": "Checks if upgrade is supported on the current instance class",
  "status": "OK",
  "description": "Upgrade from Aurora Version 2.12.0 and 2.12.1 may fail for 32.xl
 and above instance class.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Upgrade is not supported on this instance size for Aurora
 MySql Version 2.12.1. Before upgrading to Aurora MySql 3, please consider either:
 1. Changing the instance class to 24.xl or lower. -or- 2. Upgrading to patch
 version 2.12.2 or higher."
      }
  ]
},
```

La vérification préalable renvoie une erreur, car l'instance de base de données d'enregistreur utilise la classe d'instance db.r6i.32xlarge et s'exécute sur Aurora MySQL version 2.12.1.

auroraUpgradeCheckForInternalUsers

Niveau de vérification préalable : erreur

Rechercher les utilisateurs internes de la version 8.0

Cette vérification préalable ne s'applique qu'à Aurora MySQL 3.03.0 ou version antérieure. Si vous rencontrez une erreur lors de cette vérification préalable, procédez à une mise à niveau vers Aurora MySQL 3.04 ou version ultérieure.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForInternalUsers",
  "title": "Check for 8.0 internal users.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForInvalidUtf8mb3CharacterStringInViews

Niveau de vérification préalable : erreur

Rechercher la présence de caractères utf8mb3 non valides dans la définition de la vue

Cette vérification préalable identifie les vues contenant des commentaires dont l'encodage de caractères `utf8mb3` n'est pas valide. Dans MySQL 8.0, une validation plus stricte est appliquée à l'encodage des caractères dans les métadonnées, y compris les commentaires de vue. Si une définition de vue contient des caractères non valides dans le jeu de caractères `utf8mb3`, la mise à niveau échoue.

Pour résoudre ce problème, modifiez la définition de la vue afin de supprimer ou de remplacer tout caractère non BMP avant de tenter la mise à niveau.

Exemple de sortie :

```
{
"id": "auroraUpgradeCheckForInvalidUtf8mb3CharacterStringInViews",
"title": "Check for invalid utf8mb3 character string.",
"status": "OK",
"description": "Definition of following view(s) has/have invalid utf8mb3 character
 string.",
"detectedProblems": [
        {
            "level": "Error",
            "dbObject": "precheck.utf8mb3_invalid_char_view",
            "description": "Definition of view precheck.utf8mb3_invalid_char_view
 contains an invalid utf8mb3 character string. This is due to https://
bugs.mysql.com/bug.php?id=110177. To fix the inconsistency, we recommend you to
 modify the view definition to not use non-BMP characters and try the upgrade
 again."
        }
    ]
},
```

La vérification préalable indique que la définition de la vue `utf8mb3_invalid_char_view` contient des caractères `utf8mb3` non valides dans sa définition.

Pour résoudre ce problème, identifiez la vue contenant les caractères non pris en charge et mettez à jour les commentaires. Tout d'abord, examinez la structure de la vue et identifiez les commentaires.

```
MySQL> SHOW CREATE VIEW precheck.utf8mb3_invalid_char_view\G
*************************** 1. row ***************************
```

```
                View: utf8mb3_invalid_char_view
        Create View: CREATE ALGORITHM=UNDEFINED DEFINER=`admin`@`%` SQL SECURITY
 DEFINER VIEW `utf8mb3_invalid_char_view` AS select 'This row contains a dolphin #'
 AS `message`
character_set_client: utf8
collation_connection: utf8_general_ci
1 row in set, 1 warning (0.00 sec)
```

Une fois que vous avez identifié la vue contenant l'erreur, remplacez-la par l'instruction `CREATE OR REPLACE VIEW`.

```
MySQL> CREATE OR REPLACE VIEW precheck.utf8mb3_invalid_char_view AS select 'This
 view definition to not use non-BMP characters' AS message;
```

Après avoir mis à jour toutes les définitions de vue contenant des caractères non pris en charge, la vérification préalable aboutit et la mise à niveau peut avoir lieu.

```
{
"id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
"title": "Check for invalid utf8mb3 column comments.",
"status": "OK",
"detectedProblems": []
}
```

auroraUpgradeCheckForInvalidUtf8mb3ColumnComments

Niveau de vérification préalable : erreur

Rechercher la présence de commentaires de colonne utf8mb3 non valides

Cette vérification préalable identifie les tables contenant des commentaires de colonne dont l'encodage de caractères `utf8mb3` n'est pas valide. Dans MySQL 8.0, une validation plus stricte est appliquée à l'encodage des caractères dans les métadonnées, y compris les commentaires de colonne. Si des commentaires de colonne contiennent des caractères non valides dans le jeu de caractères utf8mb3, la mise à niveau échouera.

Pour résoudre ce problème, vous devez modifier les commentaires de ces colonnes afin de supprimer ou de remplacer tout caractère non BMP avant de tenter la mise à niveau. Vous pouvez utiliser l'instruction `ALTER TABLE` pour mettre à jour les commentaires des colonnes.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
  "title": "Check for invalid utf8mb3 column comments.",
  "status": "OK",
  "description": "Following table(s) has/have invalid utf8mb3 comments on the
 column/columns.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.t2",
        "description": "Table test.t2 has invalid utf8mb3 comments in it's column/
columns. This is due to non-BMP characters in the comment field. To fix the
 inconsistency, we recommend you to modify comment fields to not use non-BMP
 characters and try the upgrade again."
      }
  ]
}
```

La vérification préalable indique que le tableau `test.t2` contient des caractères `utf8mb3` non valides dans un ou plusieurs commentaires de colonne, notamment en raison de la présence de caractères non BMP.

Pour résoudre ce problème, vous pouvez identifier les colonnes problématiques et mettre à jour leurs commentaires. Tout d'abord, examinez la structure de la table pour identifier les colonnes contenant des commentaires :

```
mysql> SHOW CREATE TABLE test.t2\G
```

Une fois que vous avez identifié les colonnes contenant des commentaires problématiques, mettez-les à jour à l'aide de l'instruction `ALTER TABLE`. Exemples :

```
mysql> ALTER TABLE test.t2 MODIFY COLUMN column_name data_type COMMENT 'Updated
 comment without non-BMP characters';
```

Vous pouvez également supprimer complètement le commentaire :

```
mysql> ALTER TABLE test.t2 MODIFY COLUMN column_name data_type COMMENT '';
```

Après que vous avez mis à jour tous les commentaires de colonne problématiques, la vérification préalable aboutit et la mise à niveau peut avoir lieu :

```
{
  "id": "auroraUpgradeCheckForInvalidUtf8mb3ColumnComments",
  "title": "Check for invalid utf8mb3 column comments.",
  "status": "OK",
  "detectedProblems": []
}
```

> **Note**
>
> Avant de modifier les commentaires de colonne, assurez-vous que tout code d'application ou toute documentation reposant sur ces commentaires est mis à jour en conséquence. Envisagez d'adopter le jeu de caractères utf8mb4 pour une meilleure prise en charge Unicode si votre application nécessite des caractères non BMP.

auroraUpgradeCheckForInvalidUtf8mb3IndexComments

Niveau de vérification préalable : erreur

Rechercher la présence de commentaires d'index utf8mb3 non valides

Cette vérification préalable identifie les tables contenant des commentaires d'index dont l'encodage de caractères `utf8mb3` n'est pas valide. Dans MySQL 8.0, une validation plus stricte est appliquée à l'encodage des caractères dans les métadonnées, y compris les commentaires d'index. Si des commentaires d'index contiennent des caractères non valides dans le jeu de caractères `utf8mb3`, la mise à niveau échoue.

Pour résoudre ce problème, vous devez modifier ces commentaires d'index afin de supprimer ou de remplacer tout caractère non BMP avant de tenter la mise à niveau.

Exemple de sortie :

```
{
    "id": "auroraUpgradeCheckForInvalidUtf8mb3IndexComments",
    "title": "Check for invalid utf8mb3 index comments.",
    "status": "OK",
    "description": "Following table(s) has/have invalid utf8mb3 comments on the
 index.",
    "detectedProblems": [
        {
            "level": "Error",
```

```
            "dbObject": "precheck.utf8mb3_tab_index_comment",
            "description": "Table precheck.utf8mb3_tab_index_comment has invalid
 utf8mb3 comments in it's index. This is due to https://bugs.mysql.com/bug.php?
id=110177. To fix the inconsistency, we recommend you to modify comment fields to
 not use non-BMP characters and try the upgrade again."
        }
    ]
},
```

La vérification préalable indique que le tableau `utf8mb3_tab_index_comment` contient des caractères `utf8mb3` non valides dans un ou plusieurs commentaires de colonne, notamment en raison de la présence de caractères non BMP.

Pour résoudre ce problème, examinez d'abord la structure de la table afin d'identifier l'index contenant des commentaires problématiques.

```
MySQL> SHOW CREATE TABLE precheck.utf8mb3_tab_index_comment\G
*************************** 1. row ***************************
    Table: utf8mb3_tab_index_comment
Create Table: CREATE TABLE `utf8mb3_tab_index_comment` (
`id` int(11) DEFAULT NULL,
`name` varchar(100) DEFAULT NULL,
KEY `idx_name` (`name`) COMMENT 'Name index #'
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.01 sec)
```

Une fois que vous avez identifié l'index contenant des commentaires utilisant des caractères non pris en charge, supprimez-le et recréez-le.

> **Note**
>
> La suppression et la recréation d'un index de table peuvent entraîner une durée d'indisponibilité. Nous vous recommandons de planifier cette opération pendant la maintenance.

```
MySQL> ALTER TABLE precheck.utf8mb3_tab_index_comment DROP INDEX idx_name;
MySQL> ALTER TABLE precheck.utf8mb3_tab_index_comment ADD INDEX idx_name(name);
```

L'exemple suivant présente une autre façon de recréer l'index.

```
MySQL> ALTER TABLE utf8mb3_tab_index_comment DROP INDEX idx_name, ADD INDEX idx_name
 (name) COMMENT 'Updated comment without non-BMP characters';
```

Une fois que vous avez supprimé ou mis à jour tous les commentaires d'index non pris en charge, la vérification préalable aboutit et la mise à niveau peut avoir lieu.

```
{
"id": "auroraUpgradeCheckForInvalidUtf8mb3IndexComments",
"title": "Check for invalid utf8mb3 index comments.",
"status": "OK",
"detectedProblems": []
},
```

auroraUpgradeCheckForInvalidUtf8mb3TableComments

Niveau de vérification préalable : erreur

Rechercher la présence de caractères utf8mb3 non valides dans la définition de la table

Cette vérification préalable identifie les tables contenant des commentaires dont l'encodage de caractères `utf8mb3` n'est pas valide. Dans MySQL 8.0, une validation plus stricte est appliquée à l'encodage des caractères dans les métadonnées, y compris les commentaires des tables. Si des commentaires de table contiennent des caractères non valides dans le jeu de caractères `utf8mb3`, la mise à niveau échoue.

Pour résoudre ce problème, vous devez modifier ces commentaires de table afin de supprimer ou de remplacer tout caractère non BMP avant de tenter la mise à niveau.

Exemple de sortie :

```
{
    "id": "auroraUpgradeCheckForInvalidUtf8mb3TableComments",
    "title": "Check for invalid utf8mb3 table comments.",
    "status": "OK",
    "description": "Following table(s) has/have invalid utf8mb3 comments.",
    "detectedProblems": [
        {
            "level": "Error",
            "dbObject": "precheck.utf8mb3_table_with_comment",
            "description": "Table precheck.utf8mb3_table_with_comment has invalid
 utf8mb3 comments. This is due to https://bugs.mysql.com/bug.php?id=110177. To fix
```

```
 the inconsistency, we recommend you to modify comment fields to not use non-BMP
 characters and try the upgrade again."
       }

    ]
},
```

La vérification préalable signale des commentaires `utf8mb3` non valides définis pour les tables `utf8mb3_table_with_comment` dans la base de données de test.

Pour résoudre ce problème, identifiez la table contenant les caractères non pris en charge et mettez à jour les commentaires. Tout d'abord, examinez la structure de la table et identifiez les commentaires.

```
MySQL> SHOW CREATE TABLE precheck.utf8mb3_table_with_comment\G
*************************** 1. row ***************************
    Table: utf8mb3_table_with_comment
Create Table: CREATE TABLE `utf8mb3_table_with_comment` (
`id` int(11) NOT NULL,
`name` varchar(100) DEFAULT NULL,
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='This table comment contains flag ##'
1 row in set (0.00 sec)
```

Une fois que vous avez identifié les commentaires de table contenant des caractères non pris en charge, mettez-les à jour avec l'instruction `ALTER TABLE`.

```
MySQL> ALTER TABLE precheck.utf8mb3_table_with_comment COMMENT='Updated comment
 without non-BMP characters';
```

Vous pouvez également supprimer le commentaire.

```
MySQL> ALTER TABLE precheck.utf8mb3_table_with_comment COMMENT='';
```

Une fois que vous avez supprimé tous les caractères non pris en charge de tous les commentaires de table, la vérification préalable aboutit.

```
{
"id": "auroraUpgradeCheckForInvalidUtf8mb3TableComments",
"title": "Check for invalid utf8mb3 table comments.",
"status": "OK",
```

```
"detectedProblems": []
},
```

auroraUpgradeCheckForMasterUser

Niveau de vérification préalable : erreur

Vérifier si l'utilisateur principal RDS existe

MySQL 8 a ajouté un nouveau modèle de privilèges prenant en charge les rôles et les privilèges dynamiques afin de rendre la gestion des privilèges plus facile et plus précise. Dans le cadre de cette modification, Aurora MySQL a introduit le nouveau rôle `rds_superuser_role`, qui est automatiquement accordé à l'utilisateur principal de la base de données lors de la mise à niveau d'Aurora MySQL version 2 vers la version 3.

Pour plus d'informations sur les rôles et privilèges attribués à l'utilisateur principal dans Aurora MySQL, consultez Privilèges du compte utilisateur principal. Pour plus d'informations sur le modèle de privilèges basé sur les rôles dans Aurora MySQL version 3, consultez Modèle de privilège basé sur les rôles.

Cette vérification préalable s'assure que l'utilisateur principal se trouve dans la base de données. Si l'utilisateur principal n'y est pas, la vérification préalable échoue. Pour autoriser la mise à niveau, recréez l'utilisateur principal en réinitialisant son mot de passe ou en créant manuellement l'utilisateur. Réessayez ensuite la mise à niveau. Pour plus d'informations sur la réinitialisation du mot de passe de l'utilisateur principal, consultez Modification du mot de passe de l'utilisateur principal de la base de données..

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForMasterUser",
  "title": "Check if master user exists",
  "status": "OK",
  "description": "Throws error if master user has been dropped!",
  "documentationLink": "https://docs.aws.amazon.com/AmazonRDS/latest/
AuroraUserGuide/UsingWithRDS.MasterAccounts.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "all",
        "description": "Your Master User on host '%' has been dropped. To proceed
 with the upgrade, recreate the master user `reinvent` on default host '%'"
```

```
      }
  ]
}
```

Une fois que vous avez réinitialisé le mot de passe de l'utilisateur principal, la vérification préalable aboutit et vous pouvez réessayer la mise à niveau.

L'exemple suivant utilise l'AWS CLI pour réinitialiser le mot de passe. Les modifications de mot de passe sont appliquées immédiatement.

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --master-user-password my-new-password
```

Ensuite, la vérification préalable aboutit.

```
{
  "id": "auroraUpgradeCheckForMasterUser",
  title": "Check if master user exists",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForPrefixIndexOnGeometryColumns

Niveau de vérification préalable : erreur

Recherche la présence de colonnes de géométrie sur les index à préfixe

Depuis MySQL 8.0.12, il n'est plus possible de créer un index avec préfixe sur une colonne avec le type de données GEOMETRY. Pour plus d'informations, consultez WL#11808.

Si de tels index existent, la mise à niveau échouera. Pour résoudre le problème, supprimez les index GEOMETRY à préfixe dans les tables mentionnées lors de l'échec de la vérification préalable.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexs",
  "status": "OK",
```

```
  "description": "Consider dropping the prefix Indexes of geometry columns and
 restart the upgrade.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.geom_index_prefix",
        "description": "Table `test`.`geom_index_prefix` has an index `LatLon` on
 geometry column/s. Mysql 8.0 does not support this type of index on a geometry
 column https://dev.mysql.com/worklog/task/?id=11808. To upgrade to MySQL 8.0, Run
 'DROP INDEX `LatLon` ON `test`.`geom_index_prefix`;"
      }
  ]
}
```

La vérification préalable signale une erreur, car la table `test.geom_index_prefix` contient un index avec un préfixe dans une colonne `GEOMETRY`.

Une fois que vous supprimez cet index, la vérification préalable aboutit.

```
{
  "id": "auroraUpgradeCheckForPrefixIndexOnGeometryColumns",
  "title": "Check for geometry columns on prefix indexs",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForSpecialCharactersInProcedures

Niveau de vérification préalable : erreur

Rechercher les incohérences liées aux caractères spéciaux dans les procédures stockées

Avant MySQL 8.0, les noms de base de données, les noms de tables et les autres objets correspondaient à des fichiers du répertoire de données, c'est-à-dire à des métadonnées basées sur des fichiers. Dans le cadre de la mise à niveau vers MySQL 8.0, tous les objets de base de données sont migrés vers les nouvelles tables du dictionnaire de données internes qui sont stockées dans le schéma `mysql` afin de prendre en charge l'Atomic Data Dictionary récemment implémenté. Dans le cadre de la migration des procédures stockées, la définition et le corps de chaque procédure sont validés lors de leur ingestion dans le nouveau dictionnaire de données.

Avant MySQL 8, dans certains cas, il était possible de créer des routines stockées ou d'insérer directement dans la table `mysql.proc` des procédures contenant des caractères spéciaux.

Par exemple, vous pouviez créer une procédure stockée avec un commentaire contenant un espace incassable non conforme, `\xa0`. Si de telles procédures sont identifiées, la mise à niveau échouera.

Cette vérification préalable s'assure que les corps et les définitions de vos procédures stockées ne contiennent pas ces caractères. Pour permettre la mise à niveau, recréez ces procédures stockées sans aucun caractère masqué ou spécial.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForSpecialCharactersInProcedures",
  "title": "Check for inconsistency related to special characters in stored
 procedures.",
  "status": "OK",
  "description": "Following procedure(s) has special characters inconsistency.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "information_schema.routines",
        "description": "Data Dictionary Metadata is inconsistent for the procedure
 `get_version_proc` in the database `test` due to usage of special characters in
 procedure body. To avoid that, drop and recreate the procedure without any special
 characters before proceeding with the Upgrade."
      }
  ]
}
```

La vérification préalable indique que le cluster de bases de données contient une procédure appelée `get_version_proc` dans la base de données `test` dont le corps de la procédure contient des caractères spéciaux.

Après avoir supprimé et recréé la procédure stockée, la vérification préalable aboutit, permettant ainsi à la mise à niveau de se poursuivre.

```
{
  "id": "auroraUpgradeCheckForSpecialCharactersInProcedures",
  "title": "Check for inconsistency related to special characters in stored
 procedures.",
  "status": "OK",
  "detectedProblems": []
```

```
},
```

auroraUpgradeCheckForSysSchemaObjectTypeMismatch

Niveau de vérification préalable : erreur

Vérifier les incohérences de type d'objet par rapport au schéma **`sys`**

Le schéma sys est un ensemble d'objets et de vues d'une base de données MySQL, qui aident les utilisateurs à dépanner, optimiser et surveiller leurs instances de base de données. Lorsque vous effectuez une mise à niveau d'une version majeure d'Aurora MySQL version 2 vers Aurora MySQL version 3, les vues du schéma `sys` sont recréées et mises à jour selon les nouvelles définitions d'Aurora MySQL version 3.

Dans le cadre de la mise à niveau, si des objets du schéma `sys` sont définis à l'aide de moteurs de stockage (`sys_config/BASE TABLE` dans INFORMATION_SCHEMA.TABLES) au lieu de vues, la mise à niveau échouera. Ces tables se trouvent dans `information_schema.tables`. Ce comportement n'est pas attendu, mais dans certains cas, il peut se produire en raison d'une erreur de l'utilisateur.

Cette vérification préalable valide tous les objets de schéma `sys` pour s'assurer qu'ils utilisent les définitions de table correctes et que les vues ne sont pas définies par erreur comme des tables InnoDB ou MyISAM. Pour résoudre le problème, corrigez manuellement les objets renvoyés en les renommant ou en les supprimant. Réessayez ensuite la mise à niveau.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForSysSchemaObjectTypeMismatch",
  "title": "Check object type mismatch for sys schema.",
  "status": "OK",
  "description": "Database contains objects with type mismatch for sys schema.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "sys.waits_global_by_latency",
        "description": "Your object sys.waits_global_by_latency has a type mismatch.
 To fix the inconsistency we recommend to rename or remove the object before
 upgrading (use RENAME TABLE command). "
      }
  ]
}
```

La vérification préalable indique que la vue sys.waits_global_by_latency du schéma sys présente une incompatibilité de type qui empêche la mise à niveau d'avoir lieu.

Une fois connecté à l'instance de base de données, vous pouvez voir que cet objet est défini comme une table InnoDB, alors qu'il devrait s'agir d'une vue.

```
mysql> show create table sys.waits_global_by_latency\G
*************************** 1. row ***************************
       Table: waits_global_by_latency
Create Table: CREATE TABLE `waits_global_by_latency` (
  `events` varchar(128) DEFAULT NULL,
  `total` bigint(20) unsigned DEFAULT NULL,
  `total_latency` text,
  `avg_latency` text,
  `max_latency` text
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Pour résoudre ce problème, nous pouvons soit supprimer et recréer la vue avec la définition appropriée, soit la renommer. Au cours du processus de mise à niveau, l'objet sera automatiquement créé avec la définition de table appropriée.

```
mysql> RENAME TABLE sys.waits_global_by_latency to sys.waits_global_by_latency_old;
Query OK, 0 rows affected (0.01 sec)
```

Après cela, la vérification préalable aboutira.

```
{
  "id": "auroraUpgradeCheckForSysSchemaObjectTypeMismatch",
  "title": "Check object type mismatch for sys schema.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckForViewColumnNameLength

Niveau de vérification préalable : erreur

Vérifier la limite supérieure du nom de colonne dans la vue

La longueur maximale autorisée d'un nom de colonne dans MySQL est de 64 caractères. Avant MySQL 8.0, dans certains cas, il était possible de créer une vue avec un nom de colonne

supérieur à 64 caractères. Si des vues de ce type existent sur votre instance de base de données, une erreur est générée par la vérification préalable et la mise à niveau échoue. Pour permettre la mise à niveau, vous devez recréer les vues en question, en vous assurant que la longueur de leurs colonnes est inférieure à 64 caractères. Réessayez ensuite la mise à niveau.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForViewColumnNameLength",
  "title": "Check for upperbound limit related to column name in view.",
  "status": "OK",
  "description": "Following view(s) has column(s) with length greater than 64.",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject":
 "test.colname_view_test.col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pa
        "description": "View `test`.`colname_view_test`has column
 `col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad` with
 invalid column name length. To avoid Upgrade errors, view should be altered by
 renaming the column name so that its length is not 0 and doesn't exceed 64."
      }
  ]
}
```

La vérification préalable indique que la vue `test.colname_view_test` contient une colonne `col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad` dont la longueur dépasse la longueur maximale autorisée de 64 caractères.

En examinant la définition de la vue, nous pouvons voir la colonne incriminée.

```
mysql> desc `test`.`colname_view_test`;
+-------------------------------------------------------------------+-------------
+------+-----+---------+-------+
| Field                                                             | Type        |
 Null | Key | Default | Extra |
+-------------------------------------------------------------------+-------------
+------+-----+---------+-------+
| col1                                                              | varchar(20) |
 YES  |     | NULL    |       |
| col2_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad_pad | int(11)     |
 YES  |     | NULL    |       |
```

```
+-------------------------------------------------------------------+-------------
+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Pour permettre la mise à niveau, recréez la vue en vous assurant que la longueur de la colonne ne dépasse pas 64 caractères.

```
mysql> drop view `test`.`colname_view_test`;
Query OK, 0 rows affected (0.01 sec)

mysql> create view `test`.`colname_view_test`(col1, col2_nopad) as select inf,
 fodcol from test;
Query OK, 0 rows affected (0.01 sec)

mysql> desc `test`.`colname_view_test`;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| col1       | varchar(20) | YES  |     | NULL    |       |
| col2_nopad | int(11)     | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Après cela, la vérification préalable aboutira.

```
{
  "id": "auroraUpgradeCheckForViewColumnNameLength",
  "title": "Check for upperbound limit related to column name in view.",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckIndexLengthLimitOnTinyColumns

Niveau de vérification préalable : erreur

Rechercher les tables dont les index sont définis avec une longueur de préfixe supérieure à 255 octets dans les petites colonnes

Lorsque vous créez un index dans une colonne à l'aide d'un type de données binaire dans MySQL, vous devez ajouter une longueur de préfixe dans la définition de l'index. Avant MySQL 8.0, dans certains cas, il était possible de spécifier une longueur de préfixe supérieure à la

taille maximale autorisée pour ces types de données. Prenons l'exemple des colonnes `TINYTEXT` et `TINYBLOB`, où la taille de données maximale autorisée est de 255 octets, mais où des préfixes d'index ont une taille supérieure. Pour plus d'informations, consultez Limites InnoDB dans la documentation MySQL.

Si cette vérification préalable échoue, supprimez l'index incriminé ou réduisez la longueur du préfixe des colonnes `TINYTEXT` et `TINYBLOB` de l'index pour qu'elle ne dépasse pas 255 octets. Réessayez ensuite la mise à niveau.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinyColumns",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny columns",
  "status": "OK",
  "description": "Prefix length of the indexes defined on tiny columns cannot
 exceed 255 bytes. With utf8mb4 char set, this limits the prefix length supported
 upto 63 characters only. A larger prefix length was allowed in MySQL5.7 using
 innodb_large_prefix parameter. This parameter is deprecated in MySQL 8.0.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/innodb-limits.html,
 https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.tintxt_prefixed_index.col1",
        "description": "Index 'PRIMARY' on tinytext/tinyblob column `col1` of table
 `test.tintxt_prefixed_index` is defined with prefix length exceeding 255 bytes.
 Reduce the prefix length to <=255 bytes depending on character set used. For
 utf8mb4, it should be <=63."
      }
  ]
}
```

La vérification préalable signale une erreur pour la table `test.tintxt_prefixed_index`, car elle contient un index `PRIMARY` dont le préfixe est supérieur à 255 octets sur une colonne TINYTEXT ou TINYBLOB.

En examinant la définition de cette table, nous pouvons voir que la clé primaire a le préfixe 65 sur la colonne `TINYTEXT col1`. Comme la table est définie à l'aide du jeu de caractères `utf8mb4`, qui stocke 4 octets par caractère, le préfixe dépasse la limite de 255 octets.

```
mysql> show create table `test`.`tintxt_prefixed_index`\G
*************************** 1. row ***************************
       Table: tintxt_prefixed_index
Create Table: CREATE TABLE `tintxt_prefixed_index` (
  `col1` tinytext NOT NULL,
  `col2` tinytext,
  `col_id` tinytext,
  PRIMARY KEY (`col1`(65))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 ROW_FORMAT=DYNAMIC
1 row in set (0.00 sec)
```

Le remplacement du préfixe d'index par 63 lors de l'utilisation du jeu de caractères `utf8mb4` permettra à la mise à niveau d'avoir lieu.

```
mysql> alter table `test`.`tintxt_prefixed_index` drop PRIMARY KEY, ADD  PRIMARY KEY
 (`col1`(63));
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Après cela, la vérification préalable aboutira.

```
{
  "id": "auroraUpgradeCheckIndexLengthLimitOnTinyColumns",
  "title": "Check for the tables with indexes defined with prefix length greater
 than 255 bytes on tiny columns",
  "status": "OK",
  "detectedProblems": []
}
```

auroraUpgradeCheckMissingInnodbMetadataForMysqlHostTable

Niveau de vérification préalable : erreur

Rechercher toute incohérence des métadonnées InnoDB pour la table **`mysql.host`**

Il s'agit d'une vérification préalable interne, qui est effectuée par le service RDS. Toutes les erreurs seront traitées automatiquement lors de la mise à niveau et peuvent être ignorées sans risque.

Si vous rencontrez des erreurs lors de cette vérification préalable, ouvrez une demande d'assistance auprès d'AWS Support pour que l'incohérence des métadonnées soit résolue. Vous pouvez également réessayer la mise à niveau en procédant à un vidage logique, puis

en effectuant une restauration sur un nouveau cluster de bases de données Aurora MySQL version 3.

Avertissements

Les vérifications préalables suivantes génèrent des avertissements en cas d'échec, mais la mise à niveau peut tout de même avoir lieu.

Rubriques

- Vérifications préalables de MySQL qui signalent des avertissements
- Vérifications préalables d'Aurora MySQL qui signalent des avertissements

Vérifications préalables de MySQL qui signalent des avertissements

Les vérifications préalables suivantes sont tirées de Community MySQL :

- defaultAuthenticationPlugin
- maxdbFlagCheck
- mysqlDollarSignNameCheck
- reservedKeywordsCheck
- utf8mb3Check
- zeroDatesCheck

defaultAuthenticationPlugin

Niveau de vérification préalable : avertissement

Considérations relatives au nouveau plug+in d'authentification par défaut

Dans MySQL 8.0, le plug-in d'authentification `caching_sha2_password` a été introduit afin de fournir un chiffrement des mots de passe plus sécurisé et de meilleures performances que le plug-in `mysql_native_password` obsolète. Pour Aurora MySQL version 3, le plug-in d'authentification par défaut utilisé par les utilisateurs de base de données est le plug-in `mysql_native_password`.

Cette vérification préalable indique que ce plug-in sera supprimé et sera remplacé dans une future version majeure. Pensez à évaluer la compatibilité des clients et des utilisateurs de votre application avant cette modification.

Pour plus d'informations, consultez Problèmes de compatibilité avec caching_sha2_password et solutions dans la documentation MySQL.

Exemple de sortie :


```
{
  "id": "defaultAuthenticationPlugin",
  "title": "New default authentication plugin considerations",
  "description": "Warning: The new default authentication plugin
 'caching_sha2_password' offers more secure password hashing than previously
 used 'mysql_native_password' (and consequent improved client connection
 authentication). However, it also has compatibility implications that
 may affect existing MySQL installations. If your MySQL installation
 must serve pre-8.0 clients and you encounter compatibility issues after
 upgrading, the simplest way to address those issues is to reconfigure
 the server to revert to the previous default authentication plugin
 (mysql_native_password). For example, use these lines in the server option file:
\n\n[mysqld]\ndefault_authentication_plugin=mysql_native_password\n\nHowever, the
 setting should be viewed as temporary, not as a long term or permanent solution,
 because it causes new accounts created with the setting in effect to forego the
 improved authentication security.\nIf you are using replication please take time to
 understand how the authentication plugin changes may impact you.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-
previous-series.html#upgrade-caching-sha2-password-compatibility-issues\nhttps://
dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-caching-
sha2-password-replication"
},
```


maxdbFlagCheck

Niveau de vérification préalable : avertissement

Utilisation d'un indicateur **`MAXDB` `sql_mode`** obsolète

Dans MySQL 8.0, plusieurs options de variables système sql_mode obsolètes ont été supprimées, y compris `MAXDB`. Cette vérification préalable examine toutes les sessions actuellement connectées, ainsi que les routines et les déclencheurs, pour s'assurer que `sql_mode` n'est défini sur aucune combinaison incluant `MAXDB` pour aucune session, aucune routine ni aucun déclencheur.

Exemple de sortie :


```
{
```

```
  "id": "maxdbFlagCheck",
  "title": "Usage of obsolete MAXDB sql_mode flag",
  "status": "OK",
  "description": "Warning: The following DB objects have the obsolete MAXDB
 option persisted for sql_mode, which will be cleared during the upgrade. It can
 potentially change the datatype DATETIME into TIMESTAMP if it is used inside
 object's definition, and this in turn can change the behavior in case of dates
 earlier than 1970 or later than 2037. If this is a concern, please redefine these
 objects so that they do not rely on the MAXDB flag before running the upgrade.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/mysql-
nutshell.html#mysql-nutshell-removals",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.maxdb_stored_routine",
        "description": "PROCEDURE uses obsolete MAXDB sql_mode",
        "dbObjectType": "Routine"
      }
  ]
}
```

La vérification préalable indique que la routine `test.maxdb_stored_routine` contient une option `sql_mode` non prise en charge.

Une fois connecté à la base de données, vous pouvez voir dans la définition de routine que `sql_mode` contient `MAXDB`.

```
 > SHOW CREATE PROCEDURE test.maxdb_stored_routine\G

*************************** 1. row ***************************
           Procedure: maxdb_stored_routine
            sql_mode:
 PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE,MAXDB,NO_KEY_OPTIONS,NO_TABLE_OPTIONS,NO_FIELD_OPT
    Create Procedure: CREATE DEFINER="msandbox"@"localhost" PROCEDURE
 "maxdb_stored_routine"()
BEGIN
    SELECT * FROM test;
END
character_set_client: utf8
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

Pour résoudre le problème, recréez la procédure après avoir défini la configuration `sql_mode` appropriée sur le client.

> **Note**
>
> Selon la documentation MySQL, MySQL stocke le paramètre `sql_mode` en vigueur lorsqu'une routine est créée ou modifiée. Il exécute toujours la routine avec ce paramètre, quel que soit le paramètre `sql_mode` au moment où la routine démarre.
> Avant de modifier `sql_mode`, consultez Modes SQL Server dans la documentation MySQL. Évaluez soigneusement tout impact potentiel de cette action sur votre application.

Recréez la procédure sans l'élément `sql_mode` non pris en charge.

```
mysql > set sql_mode='PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE';
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql > DROP PROCEDURE test.maxdb_stored_routine\G
Query OK, 0 rows affected (0.00 sec)

mysql >
mysql > DELIMITER $$
mysql >
mysql > CREATE PROCEDURE test.maxdb_stored_routine()
    -> SQL SECURITY DEFINER
    -> BEGIN
    ->     SELECT * FROM test;
    -> END$$
Query OK, 0 rows affected (0.00 sec)

mysql >
mysql > DELIMITER ;
mysql > show create procedure test.maxdb_stored_routine\G
*************************** 1. row ***************************
           Procedure: maxdb_stored_routine
            sql_mode: PIPES_AS_CONCAT,ANSI_QUOTES,IGNORE_SPACE
    Create Procedure: CREATE DEFINER="msandbox"@"localhost" PROCEDURE
 "maxdb_stored_routine"()
BEGIN
    SELECT * FROM test;
END
character_set_client: utf8
```

```
collation_connection: utf8_general_ci
  Database Collation: latin1_swedish_ci
1 row in set (0.00 sec)
```

La vérification préalable aboutit.

```
{
  "id": "maxdbFlagCheck",
  "title": "Usage of obsolete MAXDB sql_mode flag",
  "status": "OK",
  "detectedProblems": []
}
```

mysqlDollarSignNameCheck

Niveau de vérification préalable : avertissement

Rechercher l'utilisation obsolète du signe dollar dans les noms d'objets

Depuis MySQL 8.0.32, l'utilisation du signe dollar ($) comme premier caractère d'un identifiant sans guillemets est obsolète. Si vous avez des schémas, des tables, des vues, des colonnes ou des routines utilisant $ comme premier caractère, cette vérification préalable renvoie un avertissement. Bien que cet avertissement n'empêche pas la mise à niveau d'avoir lieu, nous vous recommandons de prendre rapidement des mesures pour résoudre ce problème. À partir de MySQL 8.4, tout identifiant de ce type renverra une erreur de syntaxe plutôt qu'un avertissement.

Exemple de sortie :

```
{
  "id": "mysqlDollarSignNameCheck",
  "title": "Check for deprecated usage of single dollar signs in object names",
  "status": "OK",
  "description": "Warning: The following objects have names with deprecated usage
 of dollar sign ($) at the begining of the identifier. To correct this warning,
 ensure, that names starting with dollar sign, also end with it, similary to quotes
 ($example$). ",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.$deprecated_syntx",
        "description": " name starts with $ sign."
      }
  ]
```

```
},
```

La vérification préalable signale un avertissement, car la table `$deprecated_syntx` du schéma `test` contient un $ comme premier caractère.

reservedKeywordsCheck

Niveau de vérification préalable : avertissement

Utilisation d'objets de base de données dont les noms sont en conflit avec les nouveaux mots clés réservés

Cette vérification est similaire à routineSyntaxCheck, dans la mesure où il recherche l'utilisation d'objets de base de données dont les noms sont en conflit avec les nouveaux mots clés réservés. Bien que les avertissements ne bloquent pas les mises à niveau, vous devez les évaluer attentivement.

Exemple de sortie :

En utilisant l'exemple précédent avec la table nommée `except`, la vérification préalable renvoie un avertissement :

```
{
  "id": "reservedKeywordsCheck",
  "title": "Usage of db objects with names conflicting with new reserved keywords",
  "status": "OK",
  "description": "Warning: The following objects have names that conflict with
 new reserved keywords. Ensure queries sent by your applications use `quotes` when
 referring to them or they will result in errors.",
  "documentationLink": "https://dev.mysql.com/doc/refman/en/keywords.html",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.except",
        "description": "Table name",
        "dbObjectType": "Table"
      }
  ]
}
```

Cet avertissement vous indique que vous pouvez avoir à vérifier certaines requêtes d'application. Si vos requêtes d'application n'échappent pas correctement les littéraux de chaîne, elles risquent de rencontrer des erreurs après la mise à niveau vers MySQL 8.0. Vérifiez vos applications pour

confirmer ce comportement, en les testant par rapport à un clone ou à un instantané de votre cluster de bases de données Aurora MySQL exécuté sur la version 3.

Exemple de requête d'application non échappée qui échouera après la mise à niveau :

```
SELECT * FROM escape;
```

Exemple de requête d'application correctement échappée qui aboutira après la mise à niveau :

```
SELECT * FROM 'escape';
```

utf8mb3Check

Niveau de vérification préalable : avertissement

Utilisation du jeu de caractères **`utf8mb3`**

Dans MySQL 8.0, le jeu de caractères `utf8mb3` est obsolète et sera supprimé dans une future version majeure de MySQL. Cette vérification préalable est mise en œuvre pour émettre un avertissement si des objets de base de données utilisant ce jeu de caractères sont détectés. Cela n'empêchera pas la mise à niveau d'avoir lieu, mais nous vous recommandons vivement de penser à migrer les tables vers le jeu de caractères `utf8mb4`, qui est le jeu de caractères par défaut à partir de MySQL 8.0. Pour plus d'informations sur utf8mb3 et utf8mb4, consultez Conversion entre les jeux de caractères Unicode à 3 octets et à 4 octets dans la documentation MySQL.

Exemple de sortie :

```
{
  "id": "utf8mb3",
  "title": "Usage of utf8mb3 charset",
  "status": "OK",
  "description": "Warning: The following objects use the deprecated utf8mb3
 character set. It is recommended to convert them to use utf8mb4 instead, for
 improved Unicode support. The utf8mb3 character is subject to removal in the
 future.",
  "documentationLink": "https://dev.mysql.com/doc/refman/8.0/en/charset-unicode-
utf8mb3.html",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.t1.col1",
```

```
        "description": "column's default character set: utf8",
        "dbObjectType": "Column"
      },
      {
        "level": "Warning",
        "dbObject": "test.t1.col2",
        "description": "column's default character set: utf8",
        "dbObjectType": "Column"
      }
  ]
}
```

Pour résoudre ce problème, vous devez reconstruire les objets et les tables référencés. Pour plus d'informations et pour connaître les conditions préalables, consultez Conversion entre les jeux de caractères Unicode à 3 octets et à 4 octets dans la documentation MySQL.

zeroDatesCheck

Niveau de vérification préalable : avertissement

Aucune valeur de date, de date/heure ni d'horodatage

MySQL applique désormais des règles plus strictes concernant l'utilisation de valeurs nulles dans les colonnes date, datetime et timestamp. Nous vous recommandons d'utiliser les modes `NO_ZERO_IN_DATE` et `NO_ZERO_DATE SQL` conjointement avec le mode `strict`, car ils seront fusionnés avec le mode `strict` dans une future version de MySQL.

Si le paramètre `sql_mode` de l'une de vos connexions à la base de données, au moment de l'exécution de la vérification préalable, n'inclut pas ces modes, un avertissement sera émis lors de la vérification préalable. Il est possible que les utilisateurs puissent toujours insérer des valeurs nulles pour la date, les date et heure ou l'horodatage. Cependant, nous vous conseillons vivement de remplacer les valeurs nulles par des valeurs valides, car leur comportement pourrait changer à l'avenir et elles pourraient cesser de fonctionner correctement. Comme il s'agit d'un avertissement, la mises à niveau ne sera pas bloquée, mais nous vous recommandons de commencer à planifier d'apporter les modifications nécessaires.

Exemple de sortie :

```
{
  "id": "zeroDatesCheck",
  "title": "Zero Date, Datetime, and Timestamp values",
  "status": "OK",
```

```
 "description": "Warning: By default zero date/datetime/timestamp values are no
longer allowed in MySQL, as of 5.7.8 NO_ZERO_IN_DATE and NO_ZERO_DATE are included
in SQL_MODE by default. These modes should be used with strict mode as they will be
merged with strict mode in a future release. If you do not include these modes in
your SQL_MODE setting, you are able to insert date/datetime/timestamp values that
contain zeros. It is strongly advised to replace zero values with valid ones, as
they may not work correctly in the future.",
 "documentationLink": "https://lefred.be/content/mysql-8-0-and-wrong-dates/",
 "detectedProblems": [
     {
       "level": "Warning",
       "dbObject": "global.sql_mode",
       "description": "does not contain either NO_ZERO_DATE or NO_ZERO_IN_DATE
which allows insertion of zero dates"
     },
     {
       "level": "Warning",
       "dbObject": "session.sql_mode",
       "description": " of 10 session(s) does not contain either NO_ZERO_DATE or
NO_ZERO_IN_DATE which allows insertion of zero dates"
     }
 ]
}
```


Vérifications préalables d'Aurora MySQL qui signalent des avertissements

Les vérifications préalables suivantes sont spécifiques à Aurora MySQL :

- auroraUpgradeCheckForRollbackSegmentHistoryLength
- auroraUpgradeCheckForUncommittedRowModifications

auroraUpgradeCheckForRollbackSegmentHistoryLength

Niveau de vérification préalable : avertissement

Vérifie si la longueur de la liste d'historique des segments de restauration pour le cluster est élevée

Comme indiqué dans auroraUpgradeCheckForIncompleteXATransactions, lors de l'exécution du processus de mise à niveau de version majeure, il est essentiel que l'instance de base de données Aurora MySQL version 2 soit complètement arrêtée. Cela garantit que toutes les

transactions sont validées ou annulées, et qu'InnoDB a purgé tous les enregistrements du journal d'annulation.

Si la longueur de la liste d'historique des segments de restauration de votre cluster de bases de données est élevée, cela peut prolonger le temps nécessaire à InnoDB pour purger les enregistrements du journal d'annulation. Cela peut entraîner une durée d'indisponibilité prolongée pendant le processus de mise à niveau de la version majeure. Si la vérification préalable détecte que la longueur de cette liste sur votre cluster de bases de données est élevé, elle émet un avertissement. Bien que cet avertissement n'empêche pas la mise à niveau d'avoir lieu, nous vous recommandons de surveiller de près la longueur de cette liste sur votre cluster de bases de données. Une longueur de liste d'historique faible réduit la durée d'indisponibilité requise pendant une mise à niveau de version majeure. Pour plus d'informations sur la surveillance de la longueur de la liste d'historique, consultez La longueur de la liste d'historique InnoDB a considérablement augmenté.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForRollbackSegmentHistoryLength",
  "title": "Checks if the rollback segment history length for the cluster is high",
  "status": "OK",
  "description": "Rollback Segment History length is greater than 1M. Upgrade may
 take longer time.",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "information_schema.innodb_metrics",
        "description": "The InnoDB undo history list length('trx_rseg_history_len')
 is 82989114. Upgrade may take longer due to purging of undo information for old row
 versions."
      }
  ]
}
```

La vérification préalable renvoie un avertissement, car elle a détecté que la longueur de la liste d'historique d'annulation d'InnoDB était élevée sur le cluster de bases de données (82989114). Bien que la mise à niveau se poursuive, en fonction du nombre d'annulations à purger, elle peut prolonger la durée d'indisponibilité requise pendant le processus de mise à niveau.

Nous vous recommandons d'examiner les transactions en cours sur votre cluster de bases de données avant d'exécuter la mise à niveau en production, afin de vous assurer que la longueur de votre liste d'historique reste à une taille gérable.

auroraUpgradeCheckForUncommittedRowModifications

Niveau de vérification préalable : avertissement

Vérifie s'il existe de nombreuses modifications de ligne non validées

Comme indiqué dans auroraUpgradeCheckForIncompleteXATransactions, lors de l'exécution du processus de mise à niveau de version majeure, il est essentiel que l'instance de base de données Aurora MySQL version 2 soit complètement arrêtée. Cela garantit que toutes les transactions sont validées ou annulées, et qu'InnoDB a purgé tous les enregistrements du journal d'annulation.

Si votre cluster de bases de données contient des transactions qui ont modifié un grand nombre de lignes, cela peut prolonger le temps nécessaire à InnoDB pour annuler cette transaction dans le cadre du processus d'arrêt complet. Si la vérification préalable détecte des transactions de longue durée comportant un grand nombre de lignes modifiées sur votre cluster de bases de données, un avertissement s'affiche. Bien que cet avertissement n'empêche pas la mise à niveau d'avoir lieu, nous vous recommandons de surveiller de près la taille des transactions actives sur votre cluster de bases de données. Une longueur de liste d'historique faible réduit la durée d'indisponibilité requise pendant une mise à niveau de version majeure.

Exemple de sortie :

```
{
  "id": "auroraUpgradeCheckForUncommittedRowModifications",
  "title": "Checks if there are many uncommitted modifications to rows",
  "status": "OK",
  "description": "Database contains uncommitted row changes greater than 10M.
 Upgrade may take longer time.",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "information_schema.innodb_trx",
        "description": "The database contains 11000000 uncommitted row change(s) in
 1 transaction(s). Upgrade may take longer due to transaction rollback."
      }
  ]
```

```
},
```

La vérification préalable indique que le cluster de bases de données contient une transaction avec 11 000 000 modifications de lignes non validées qui devront être annulées pendant le processus d'arrêt complet. La mise à niveau aura lieu, mais pour réduire la durée d'indisponibilité pendant le processus de mise à niveau, nous vous recommandons de surveiller et d'étudier ce cas avant d'exécuter la mise à niveau sur vos clusters de production.

Pour consulter les transactions actives sur votre instance de base de données d'enregistreur, vous pouvez utiliser la table information_schema.innodb_trx. La requête suivante sur l'instance de base de données d'enregistreur indique les transactions en cours, la durée d'exécution, l'état et les lignes modifiées pour le cluster de bases de données.

```
# Example of uncommitted transaction
mysql> SELECT trx_started,
       TIME_TO_SEC(TIMEDIFF(now(), trx_started)) AS seconds_trx_has_been_running,
       trx_mysql_thread_id AS show_processlist_connection_id,
       trx_id,
       trx_state,
       trx_rows_modified AS rows_modified
FROM information_schema.innodb_trx;
+---------------------+------------------------------
+---------------------------------+----------+-----------+---------------+
| trx_started         | seconds_trx_has_been_running |
 show_processlist_connection_id | trx_id   | trx_state | rows_modified |
+---------------------+------------------------------
+---------------------------------+----------+-----------+---------------+
| 2024-08-12 18:32:52 |                          1592 |
 20041 | 52866130 | RUNNING   |      11000000 |
+---------------------+------------------------------
+---------------------------------+----------+-----------+---------------+
1 row in set (0.01 sec)

# Example of transaction rolling back
mysql> SELECT trx_started,
       TIME_TO_SEC(TIMEDIFF(now(), trx_started)) AS seconds_trx_has_been_running,
       trx_mysql_thread_id AS show_processlist_connection_id,
       trx_id,
       trx_state,
       trx_rows_modified AS rows_modified
FROM information_schema.innodb_trx;
```

```
+--------------------+------------------------------
+--------------------------------+---------+-------------+--------------+
| trx_started         | seconds_trx_has_been_running |
 show_processlist_connection_id | trx_id   | trx_state    | rows_modified |
+--------------------+------------------------------
+--------------------------------+---------+-------------+--------------+
| 2024-08-12 18:32:52 |                          1719 |
 20041 | 52866130 | ROLLING BACK |      10680479 |
+--------------------+------------------------------
+--------------------------------+---------+-------------+--------------+
1 row in set (0.01 sec)
```

Une fois la transaction validée ou annulée, la vérification préalable ne renvoie plus d'avertissement. Consultez la documentation MySQL et faites appel l'équipe chargée de votre application avant d'annuler des transactions importantes, car la restauration peut prendre un certain temps, en fonction de la taille des transactions.

```
{
  "id": "auroraUpgradeCheckForUncommittedRowModifications",
  "title": "Checks if there are many uncommitted modifications to rows",
  "status": "OK",
  "detectedProblems": []
},
```

Pour plus d'informations sur l'optimisation de la gestion des transactions InnoDB et sur l'impact potentiel de l'exécution et de l'annulation de transactions importantes sur les instances de base de données MySQL, consultez Optimisation de la gestion des transactions InnoDB dans la documentation MySQL.

Notifications

La vérification préalable suivante génère une notification en cas d'échec, mais la mise à niveau peut tout de même avoir lieu.

sqlModeFlagCheck

Niveau de vérification préalable : notification

Utilisation d'indicateurs **`sql_mode`** obsolètes

Outre `MAXDB`, plusieurs autres options `sql_mode` ont été supprimées : `DB2`, `MSSQL`, `MYSQL323`, `MYSQL40`, `ORACLE`, `POSTGRESQL`, `NO_FIELD_OPTIONS`, `NO_KEY_OPTIONS` et `NO_TABLE_OPTIONS`. Depuis MySQL 8.0, aucune de ces valeurs ne peut être affectée à la variable système `sql_mode`. Si cette vérification préalable détecte des sessions en cours utilisant ces paramètres `sql_mode`, assurez-vous que les groupes de paramètres de votre instance de base de données et de cluster de bases de données, ainsi que les applications et les configurations clientes, sont mis à jour pour les désactiver. Pour plus d'informations, consultez la documentation MySQL.

Exemple de sortie :

```
{
  "id": "sqlModeFlagCheck",
  "title": "Usage of obsolete sql_mode flags",
  "status": "OK",
  "detectedProblems": []
}
```

Pour résoudre l'un de ces échecs de vérification préalable, consultez maxdbFlagCheck.

Erreurs, avertissements ou notifications

La vérification préalable suivante peut renvoyer une erreur, un avertissement ou une notification en fonction de sa sortie.

checkTableOutput

Niveau de vérification préalable : erreur, avertissement ou notification

Problèmes signalés par la commande **`check table x for upgrade`**

Avant de commencer la mise à niveau vers Aurora MySQL version 3, `check table for upgrade` est exécuté sur chaque table dans les schémas utilisateur de votre cluster de bases de données. Cette vérification préalable n'est la même que checkTableMysqlSchema.

La commande `check table for upgrade` examine les tables pour détecter tout problème potentiel pouvant survenir lors d'une mise à niveau vers une version plus récente de MySQL. L'exécution de cette commande avant de tenter une mise à niveau contribue à identifier et à résoudre les incompatibilités à l'avance, ce qui facilite le processus de mise à niveau réel.

Cette commande effectue différentes vérifications sur chaque table, telles que les suivantes :

- Vérification de la compatibilité de la structure et des métadonnées de la table avec la version MySQL cible
- Identification des fonctionnalités obsolètes ou supprimées qui sont utilisées par la table
- Confirmation de la possibilité de mettre à niveau la table sans perte de données

Contrairement aux autres vérifications préalables, celle-ci peut renvoyer une erreur, un avertissement ou une notification en fonction de la sortie `check table`. Si cette vérification préalable renvoie des tables, examinez-les attentivement, ainsi que le code de retour et le message, avant de lancer la mise à niveau. Pour plus d'informations, consultez Instruction CHECK TABLE dans la documentation MySQL.

Nous fournissons ci-dessous un exemple d'erreur et un exemple d'avertissement.

Exemple d'erreur :

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Error",
        "dbObject": "test.parent",
        "description": "Table 'test.parent' doesn't exist"
      }
  ]
},
```

La vérification préalable signale une erreur indiquant que la table `test.parent` n'existe pas.

Le fichier `mysql-error.log` de l'instance de base de données d'enregistreur indique qu'il existe une erreur de clé étrangère.

```
2024-08-13T15:32:10.676893Z 62 [Warning] InnoDB: Load table `test`.`parent` failed,
 the table has missing foreign key indexes. Turn off 'foreign_key_checks' and try
 again.
2024-08-13T15:32:10.676905Z 62 [Warning] InnoDB: Cannot open table test/parent from
 the internal data dictionary of InnoDB though the .frm file for the table exists.
Please refer to http://dev.mysql.com/doc/refman/5.7/en/innodb-troubleshooting.html
 for how to resolve the issue.
```

Connectez-vous à l'instance de base de données d'enregistreur et exécutez `show engine innodb status\G` pour obtenir plus d'informations sur l'erreur de clé étrangère.

```
mysql> show engine innodb status\G
*************************** 1. row ***************************
  Type: InnoDB
  Name:
Status:
=====================================
2024-08-13 15:33:33 0x14ef7b8a1700 INNODB MONITOR OUTPUT
=====================================
.
.
.
------------------------
LATEST FOREIGN KEY ERROR
------------------------
2024-08-13 15:32:10 0x14ef6dbbb700 Error in foreign key constraint of table test/
child:
there is no index in referenced table which would contain
the columns as the first columns, or the data types in the
referenced table do not match the ones in table. Constraint:
,
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
The index in the foreign key in table is p_name_idx
Please refer to http://dev.mysql.com/doc/refman/5.7/en/innodb-foreign-key-
constraints.html for correct foreign key definition.
.
.
```

Le message `LATEST FOREIGN KEY ERROR` indique que la contrainte de clé étrangère `fk_pname` dans la table `test.child`, qui référence la table `test.parent`, présente un problème d'index manquant ou d'incompatibilité du type de données. La documentation MySQL sur les contraintes liées aux clés étrangères indique que les colonnes référencées dans une clé étrangère doivent avoir un index associé et que les colonnes parent/enfant doivent utiliser le même type de données.

Pour vérifier si le problème est lié à un index manquant ou à une incompatibilité du type de données, connectez-vous à la base de données et vérifiez les définitions de table en désactivant temporairement la variable de session foreign_key_checks. Nous pouvons maintenant voir que la contrainte enfant en question (`fk_pname`) utilise `p_name varchar(20) CHARACTER SET`

`latin1 DEFAULT NULL` pour référencer la table parent `name varchar(20) NOT NULL`. La table parent utilise `DEFAULT CHARSET=utf8`, mais la colonne `p_name` de la table enfant utilise `latin1`, de sorte que l'erreur d'incompatibilité du type de données est renvoyée.

```
mysql> show create table parent\G
ERROR 1146 (42S02): Table 'test.parent' doesn't exist

mysql> show create table child\G
*************************** 1. row ***************************
       Table: child
Create Table: CREATE TABLE `child` (
  `id` int(11) NOT NULL,
  `p_name` varchar(20) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `p_name_idx` (`p_name`),
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

mysql> set foreign_key_checks=0;
Query OK, 0 rows affected (0.00 sec)

mysql> show create table parent\G
*************************** 1. row ***************************
       Table: parent
Create Table: CREATE TABLE `parent` (
  `name` varchar(20) NOT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)

mysql> show create table child\G
*************************** 1. row ***************************
       Table: child
Create Table: CREATE TABLE `child` (
  `id` int(11) NOT NULL,
  `p_name` varchar(20) CHARACTER SET latin1 DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `p_name_idx` (`p_name`),
  CONSTRAINT `fk_pname` FOREIGN KEY (`p_name`) REFERENCES `parent` (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

Pour résoudre ce problème, nous pouvons soit modifier la table enfant pour qu'elle utilise le même jeu de caractères que la table parent, soit modifier la table parent pour qu'elle utilise le même jeu de caractères que la table enfant. Ici, comme la table enfant utilise explicitement `latin1` dans la définition de colonne `p_name`, nous exécutons `ALTER TABLE` pour modifier le jeu de caractères sur `utf8`.

```
mysql> alter table child modify p_name varchar(20) character set utf8 DEFAULT NULL;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> flush tables;
Query OK, 0 rows affected (0.01 sec)
```

Après cela, la vérification préalable aboutira et la mise à niveau pourra avoir lieu.

Exemple d'avertissement :

```
{
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": [
      {
        "level": "Warning",
        "dbObject": "test.orders",
        "description": "Trigger test.orders.delete_audit_trigg does not have CREATED
 attribute."
      }
  ]
}
```

La vérification préalable signale un avertissement concernant le déclencheur `delete_audit_trigg` sur la table `test.orders`, car celui-ci ne possède pas d'attribut `CREATED`. Selon la section Vérification de la compatibilité des versions de la documentation MySQL, ce message est informatif et s'affiche pour les déclencheurs créés avant MySQL 5.7.2.

Comme il s'agit d'un avertissement, il n'empêche pas la mise à niveau d'avoir lieu. Toutefois, si vous souhaitez résoudre le problème, vous pouvez recréer le déclencheur en question pour que la vérification préalable aboutisse sans avertissement.

```
{
```

```
  "id": "checkTableOutput",
  "title": "Issues reported by 'check table x for upgrade' command",
  "status": "OK",
  "detectedProblems": []
},
```

Comment effectuer une mise à niveau sur place

Nous vous conseillons de passer en revue la documentation dans Fonctionnement de la mise à niveau sur place d'une version majeure de Aurora MySQL.

Effectuez la planification et les tests préalables nécessaires à la mise à niveau, comme décrit dans Planification d'une mise à niveau de version majeure d'un cluster Aurora MySQL.

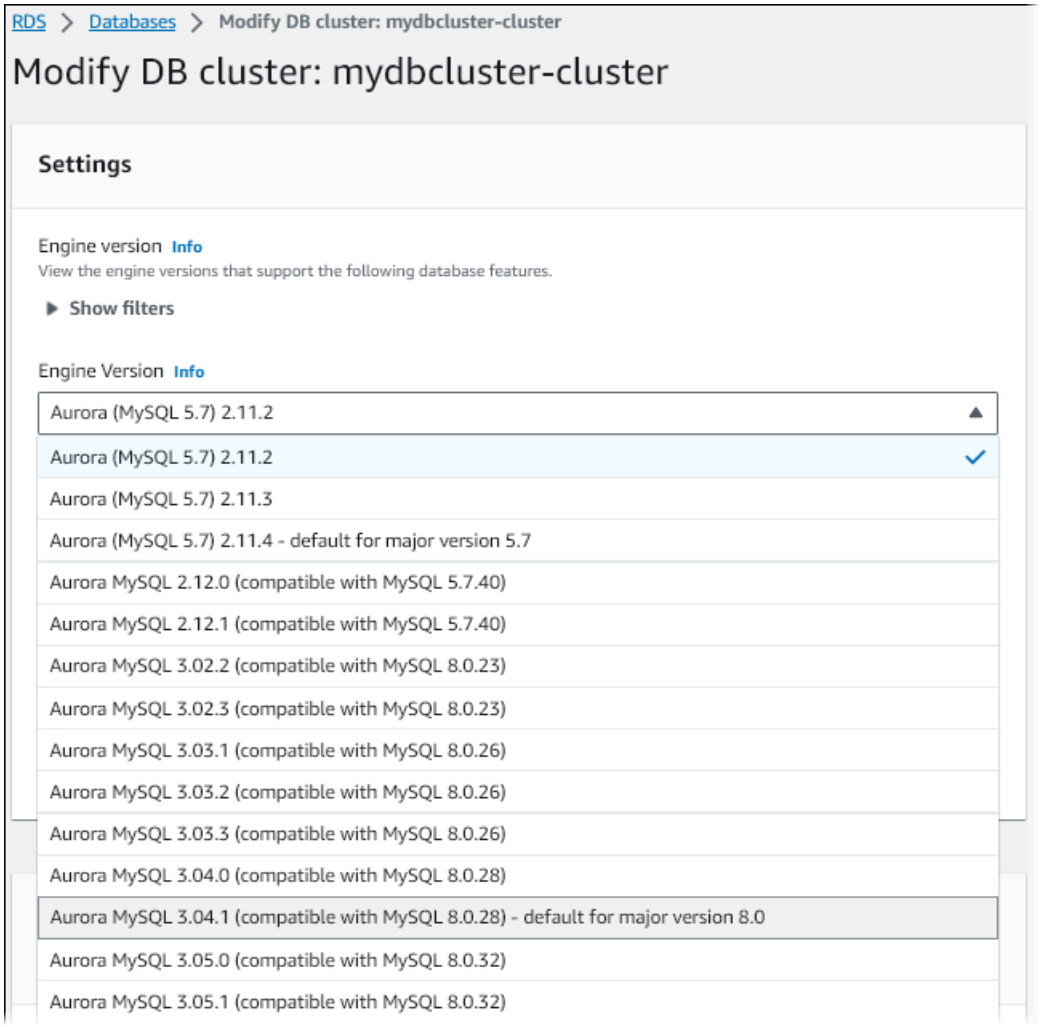
Console

L'exemple suivant met à niveau le cluster de bases de données `mydbcluster-cluster` vers Aurora MySQL version 3.04.1.

Pour mettre à niveau la version majeure d'un cluster de bases de données Aurora MySQL

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Si vous avez utilisé un groupe de paramètres personnalisé avec le cluster de bases de données d'origine, créez un groupe de paramètres correspondant compatible avec la nouvelle version majeure. Apportez les modifications nécessaires aux paramètres de configuration de ce nouveau groupe de paramètres. Pour plus d'informations, consultez Comment les mises à niveau sur place affectent les groupes de paramètres d'un cluster.
3. Dans la panneau de navigation, choisissez Databases (Bases de données).
4. Dans la liste, sélectionnez le cluster de bases de données à modifier.
5. Sélectionnez Modify.
6. Pour Version, sélectionnez une nouvelle version majeure d'Aurora MySQL.

   Nous conseillons généralement d'utiliser la dernière version mineure de la version majeure. Ici, nous choisissons la version par défaut actuelle.



7. Choisissez Continuer.
8. Sur la page suivante, indiquez quand effectuer la mise à niveau. Sélectionnez During the next scheduled maintenance window (Pendant la fenêtre de maintenance planifiée suivante) ou Immediately (Immédiatement).
9. (Facultatif) Examinez régulièrement la page Events (Événements) de la console RDS pendant la mise à niveau pour mieux surveiller la progression de la mise à niveau et identifier les problèmes éventuels. Si la mise à niveau rencontre des problèmes, consultez Dépannage de la mise à niveau sur place d'Aurora MySQL pour connaître les étapes à suivre.
10. Si vous avez créé un nouveau groupe de paramètres au début de cette procédure, associez le groupe de paramètres personnalisé à votre cluster mis à niveau. Pour plus d'informations, consultez Comment les mises à niveau sur place affectent les groupes de paramètres d'un cluster.

> **Note**
>
> Pour effectuer cette étape, vous devez redémarrer le cluster afin d'appliquer le nouveau groupe de paramètres.

11. (Facultatif) Après avoir terminé les tests postérieurs à la mise à niveau, supprimez l'instantané manuel créé par Aurora au début de la mise à niveau.

AWS CLI

Pour mettre à niveau la version majeure d'un cluster de bases de données Aurora MySQL, utilisez la commande de l'AWS CLI modify-db-cluster avec les paramètres requis suivants :

- `--db-cluster-identifier`
- `--engine-version`
- `--allow-major-version-upgrade`
- `--apply-immediately` ou `--no-apply-immediately`

Si votre cluster utilise des groupes de paramètres personnalisés, incluez également l'une des options suivantes ou les deux :

- `--db-cluster-parameter-group-name`, si le cluster utilise un groupe de paramètres de cluster personnalisé
- `--db-instance-parameter-group-name`, si des instances du cluster utilisent un groupe de paramètres de base de données personnalisé

L'exemple suivant met à niveau le cluster de bases de données `sample-cluster` vers Aurora MySQL version 3.04.1. La mise à niveau se produit immédiatement, au lieu d'attendre la fenêtre de maintenance suivante.

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
          --db-cluster-identifier sample-cluster \
          --engine-version 8.0.mysql_aurora.3.04.1 \
```

```
          --allow-major-version-upgrade \
          --apply-immediately
```

Pour Windows :

```
aws rds modify-db-cluster ^
          --db-cluster-identifier sample-cluster ^
          --engine-version 8.0.mysql_aurora.3.04.1 ^
          --allow-major-version-upgrade ^
          --apply-immediately
```

Vous pouvez combiner d'autres commandes de la CLI avec `modify-db-cluster` pour créer un processus automatisé de bout en bout permettant d'effectuer et de vérifier les mises à niveau. Pour plus d'informations et d'exemples, consultez Tutoriel de mise à niveau sur place d'Aurora MySQL.

> **Note**
>
> Si votre cluster fait partie d'une base de données Aurora globale, la procédure de mise à niveau sur place est légèrement différente. Vous appelez l'opération de commande modify-global-cluster au lieu de `modify-db-cluster`. Pour plus d'informations, consultez Mises à niveau majeures sur place des bases de données globales.

API RDS

Pour mettre à niveau la version majeure d'un cluster de bases de données Aurora MySQL, utilisez l'opération d'API RDS ModifyDBCluster avec les paramètres requis suivants :

- `DBClusterIdentifier`
- `Engine`
- `EngineVersion`
- `AllowMajorVersionUpgrade`
- `ApplyImmediately` (défini sur `true` ou `false`).

> **Note**
>
> Si votre cluster fait partie d'une base de données Aurora globale, la procédure de mise à niveau sur place est légèrement différente. Appelez l'opération ModifyGlobalCluster au lieu

de `ModifyDBCluster`. Pour plus d'informations, consultez Mises à niveau majeures sur place des bases de données globales.

Comment les mises à niveau sur place affectent les groupes de paramètres d'un cluster

Les groupes de paramètres Aurora ont différents ensembles de paramètres de configuration pour les clusters compatibles avec MySQL 5.7 ou 8.0. Lorsque vous effectuez une mise à niveau sur place, le cluster mis à niveau et toutes ses instances doivent utiliser les groupes de paramètres de cluster et d'instance correspondants :

Votre cluster et vos instances peuvent utiliser les groupes de paramètres compatibles avec la version 5.7 par défaut. Si tel est le cas, le cluster mis à niveau et l'instance commencent par les groupes de paramètres compatibles avec la version 8.0 par défaut. Si votre cluster et vos instances utilisent des groupes de paramètres personnalisés, assurez-vous de créer des groupes de paramètres correspondants compatibles avec la version 8.0. Assurez-vous également de les spécifier au cours du processus de mise à niveau.

Note

Pour la plupart des paramètres, vous pouvez choisir le groupe de paramètres personnalisé à deux points. C'est lorsque vous créez le cluster ou que vous associez le groupe de paramètres au cluster ultérieurement.
Toutefois, si vous utilisez un autre paramètre que le paramètre par défaut pour `lower_case_table_names`, vous devez configurer le groupe de paramètres personnalisés avec ce paramètre à l'avance. Spécifiez ensuite le groupe de paramètres lorsque vous effectuez la restauration des instantanés pour créer le cluster. Les modifications apportées au paramètre `lower_case_table_names` n'ont aucun effet après la création du cluster.
Nous vous recommandons d'utiliser le même paramètre pour `lower_case_table_names` lorsque vous passez de la version 2 à la version 3 de Aurora MySQL.
Avec une base de données globale Aurora basée sur Aurora MySQL, vous ne pouvez effectuer une mise à niveau sur place d'Aurora MySQL version 2 vers la version 3 que si le paramètre `lower_case_table_names` est défini sur sa valeur par défaut et si vous redémarrez la base de données globale. Pour plus d'informations sur les méthodes que vous pouvez utiliser, consultez Mises à niveau de version majeure..

Modifications apportées aux propriétés du cluster entre les versions d'Aurora MySQL

Lorsque vous effectuez une mise à niveau d'Aurora MySQL version 2 vers la version 3, veillez à vérifier les applications et les scripts que vous utilisez pour configurer ou gérer des clusters et des instances de base de données Aurora MySQL.

En outre, modifiez le code qui manipule les groupes de paramètres afin qu'il tienne compte du fait que les noms de groupes de paramètres par défaut sont différents pour les clusters compatibles avec 5.7 et 8.0. Les noms des groupes de paramètres par défaut pour des clusters Aurora MySQL versions 2 et 3 sont `default.aurora-mysql5.7` et `default.aurora-mysql8.0`, respectivement.

Par exemple, vous pouvez avoir un code semblable au suivant qui s'applique à votre cluster avant une mise à niveau.

```
# Check the default parameter values for MySQL 5.7-compatible clusters.
aws rds describe-db-parameters --db-parameter-group-name default.aurora-mysql5.7 --
region us-east-1
```

Après la mise à niveau de la version majeure du cluster, modifiez ce code comme suit.

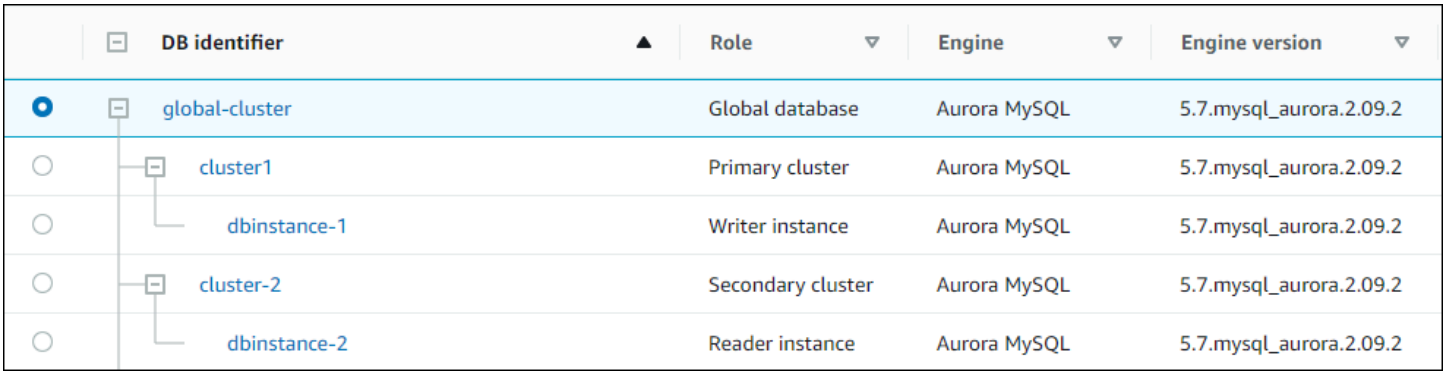
```
# Check the default parameter values for MySQL 8.0-compatible clusters.
aws rds describe-db-parameters --db-parameter-group-name default.aurora-mysql8.0 --
region us-east-1
```

Mises à niveau majeures sur place des bases de données globales

Pour une base de données Aurora globale, mettez à niveau le cluster de bases de données globale. Aurora met automatiquement à niveau tous les clusters en même temps et s'assure qu'ils utilisent tous la même version du moteur. Cette exigence est due au fait que toutes les modifications apportées aux tables système, aux formats de fichiers de données et autres éléments sont automatiquement répliquées sur tous les clusters secondaires.

Suivez les instructions de la section Fonctionnement de la mise à niveau sur place d'une version majeure de Aurora MySQL. Lorsque vous spécifiez ce qui doit être mis à niveau, veillez à choisir le cluster de bases de données global plutôt que l'un des clusters qu'il contient.

Si vous utilisez l'AWS Management Console, choisissez l'élément avec le rôle Global database (Base de données globale).



DB identifier	Role	Engine	Engine version
global-cluster	Global database	Aurora MySQL	5.7.mysql_aurora.2.09.2
cluster1	Primary cluster	Aurora MySQL	5.7.mysql_aurora.2.09.2
dbinstance-1	Writer instance	Aurora MySQL	5.7.mysql_aurora.2.09.2
cluster-2	Secondary cluster	Aurora MySQL	5.7.mysql_aurora.2.09.2
dbinstance-2	Reader instance	Aurora MySQL	5.7.mysql_aurora.2.09.2

Si vous utilisez l'AWS CLI ou l'API RDS, lancez le processus de mise à niveau en appelant la commande modify-global-cluster ou l'opération ModifyGlobalCluster. Vous utilisez l'un d'entre eux au lieu de `modify-db-cluster` ou `ModifyDBCluster`.

> **Note**
>
> Vous ne pouvez pas spécifier un groupe de paramètres personnalisés pour le cluster de bases de données globale pendant que vous effectuez une mise à niveau majeure de la version de cette base de données globale Aurora. Créez vos groupes de paramètres personnalisés dans chaque région du cluster global. Appliquez-les ensuite manuellement aux clusters régionaux après la mise à niveau.

Pour mettre à niveau la version majeure d'un cluster de bases de données Aurora MySQL global à l'aide de l'interface AWS CLI, utilisez la commande modify-global-cluster avec les paramètres requis suivants :

- `--global-cluster-identifier`
- `--engine aurora-mysql`
- `--engine-version`
- `--allow-major-version-upgrade`

L'exemple suivant met à niveau le cluster de bases de données global vers Aurora MySQL version 3.04.2.

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-global-cluster \
          --global-cluster-identifier global_cluster_identifier \
          --engine aurora-mysql \
          --engine-version 8.0.mysql_aurora.3.04.2 \
          --allow-major-version-upgrade
```

Pour Windows :

```
aws rds modify-global-cluster ^
          --global-cluster-identifier global_cluster_identifier ^
          --engine aurora-mysql ^
          --engine-version 8.0.mysql_aurora.3.04.2 ^
          --allow-major-version-upgrade
```

Mises à niveau sur place pour les clusters de bases de données ayant des réplicas en lecture entre régions

Vous pouvez mettre à niveau un cluster de bases de données Aurora ayant un réplica en lecture entre régions à l'aide de la procédure de mise à niveau sur place, mais certains éléments doivent être pris en considération :

- Vous devez d'abord mettre à niveau le cluster de bases de données du réplica en lecture. Si vous essayez d'abord de mettre à niveau le cluster principal, vous recevez un message d'erreur similaire au message suivant :

  Impossible de mettre à niveau le cluster de bases de données test-xr-primary-cluster, car le réplica entre régions d'Aurora associé, test-xr-replica-cluster, n'a pas encore reçu les correctifs requis. Mettez à niveau le réplica entre régions d'Aurora et réessayez.

  Cela signifie que le cluster de bases de données principal ne peut pas avoir une version de moteur de base de données supérieure à celle du cluster de réplication.
- Avant de mettre à niveau le cluster de bases de données principal, arrêtez la charge de travail d'écriture et désactivez toute nouvelle demande de connexion à l'instance de base de données d'enregistreur du cluster principal.
- Lorsque vous mettez à niveau le cluster principal, choisissez un groupe de paramètres de cluster de bases de données personnalisé dont le paramètre `binlog_format` est défini sur une valeur qui prend en charge la réplication des journaux binaires, comme `MIXED`.

Pour plus d'informations sur l'utilisation de la journalisation binaire Aurora, consultez Réplication entre Aurora et MySQL ou entre Aurora et un autre cluster de bases de données Aurora (réplication de journaux binaires). Pour plus d'informations sur la modification des paramètres de configuration de Aurora MySQL, consultez Paramètres de configuration d'Aurora MySQL et Groupes de paramètres pour Amazon Aurora.

- N'attendez pas trop longtemps pour mettre à niveau le cluster de bases de données principal après avoir mis à niveau le cluster du réplica. Nous vous recommandons de ne pas attendre au-delà de la fenêtre de maintenance suivante.
- Après avoir mis à niveau le cluster de bases de données principal, redémarrez son instance de base de données d'enregistreur. Le groupe de paramètres de cluster de bases de données personnalisé qui active la réplication des journaux binaires ne prend effet qu'une fois que l'instance de base de données d'enregistreur est redémarrée.
- Ne relancez pas la charge de travail d'écriture et n'activez pas les connexions à l'instance de base de données d'enregistreur tant que vous n'avez pas confirmé que la réplication entre régions a redémarré et que le délai de réplication de la Région AWS secondaire est de 0.

Tutoriel de mise à niveau sur place d'Aurora MySQL

Les exemples Linux suivants montrent comment effectuer les grandes étapes de la procédure de mise à niveau sur place à l'aide de la AWS CLI.

Ce premier exemple crée un cluster de bases de données Aurora exécutant une version 2.x d'Aurora MySQL. Le cluster comprend une instance de base de données de scripteur et une instance de base de données de lecteur. La commande `wait db-instance-available` se suspend tant que l'instance de base de données du scripteur n'est pas disponible. C'est là que le cluster est prêt à être mis à niveau.

```
aws rds create-db-cluster --db-cluster-identifier mynewdbcluster --engine aurora-mysql
 \
  --db-cluster-version 5.7.mysql_aurora.2.11.2
...
aws rds create-db-instance --db-instance-identifier mynewdbcluster-instance1 \
  --db-cluster-identifier mynewdbcluster --db-instance-class db.t4g.medium --engine
 aurora-mysql
...
aws rds wait db-instance-available --db-instance-identifier mynewdbcluster-instance1
```

Les versions 3.x d'Aurora MySQL vers lesquelles vous pouvez mettre à niveau le cluster dépendent de la version 2.x que le cluster exécute actuellement et de la Région AWS dans laquelle il se trouve. La première commande, avec `--output text`, montre simplement la version cible disponible. La deuxième commande affiche la sortie JSON complète de la réponse. Dans cette réponse, vous pouvez voir des détails tels que la valeur `aurora-mysql` que vous utilisez pour le paramètre `engine`. Vous pouvez également voir le fait que la mise à niveau vers 3.04.0 représente une mise à niveau de version majeure.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].{EngineVersion:EngineVersion}' --output text
5.7.mysql_aurora.2.11.2

aws rds describe-db-engine-versions --engine aurora-mysql --engine-version
 5.7.mysql_aurora.2.11.2 \
  --query '*[].[ValidUpgradeTarget]'
...
{
    "Engine": "aurora-mysql",
    "EngineVersion": "8.0.mysql_aurora.3.04.0",
    "Description": "Aurora MySQL 3.04.0 (compatible with MySQL 8.0.28)",
    "AutoUpgrade": false,
    "IsMajorVersionUpgrade": true,
    "SupportedEngineModes": [
        "provisioned"
    ],
    "SupportsParallelQuery": true,
    "SupportsGlobalDatabases": true,
    "SupportsBabelfish": false,
    "SupportsIntegrations": false
},
...
```

Cet exemple montre pourquoi, si vous saisissez un numéro de version cible qui n'est pas une cible de mise à niveau valide pour le cluster, Aurora n'effectue pas la mise à niveau. Aurora n'effectue pas non plus de mise à niveau de version majeure, sauf si vous incluez le paramètre `--allow-major-version-upgrade`. Ainsi, vous ne pouvez pas effectuer par erreur une mise à niveau susceptible de nécessiter des tests approfondis et des modifications du code de votre application.

```
aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 5.7.mysql_aurora.2.09.2 --apply-immediately
```

```
An error occurred (InvalidParameterCombination) when calling the ModifyDBCluster
 operation: Cannot find upgrade target from 5.7.mysql_aurora.2.11.2 with requested
 version 5.7.mysql_aurora.2.09.2.

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --region us-east-1 --apply-immediately
An error occurred (InvalidParameterCombination) when calling the ModifyDBCluster
 operation: The AllowMajorVersionUpgrade flag must be present when upgrading to a new
 major version.

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --apply-immediately --allow-major-version-
upgrade
{
  "DBClusterIdentifier": "mynewdbcluster",
  "Status": "available",
  "Engine": "aurora-mysql",
  "EngineVersion": "5.7.mysql_aurora.2.11.2"
}
```

Quelques instants sont nécessaires pour que l'état du cluster et des instances de base de données associées passe à `upgrading`. Les numéros de version du cluster et des instances de base de données ne changent que lorsque la mise à niveau est terminée. Encore une fois, vous pouvez utiliser la commande `wait db-instance-available` pour l'instance de base de données du scripteur afin d'attendre la fin de la mise à niveau avant de poursuivre.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].[Status,EngineVersion]' --output text
upgrading 5.7.mysql_aurora.2.11.2

aws rds describe-db-instances --db-instance-identifier mynewdbcluster-instance1 \
  --query '*[].
{DBInstanceIdentifier:DBInstanceIdentifier,DBInstanceStatus:DBInstanceStatus} | [0]'
{
    "DBInstanceIdentifier": "mynewdbcluster-instance1",
    "DBInstanceStatus": "upgrading"
}

aws rds wait db-instance-available --db-instance-identifier mynewdbcluster-instance1
```

À ce stade, le numéro de version du cluster correspond à celui spécifié pour la mise à niveau.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].[EngineVersion]' --output text

8.0.mysql_aurora.3.04.0
```

L'exemple précédent représente une mise à niveau immédiate en spécifiant le paramètre `--apply-immediately`. Pour que la mise à niveau se produise à un moment opportun, lorsque le cluster ne devrait pas être occupé, vous pouvez spécifier le paramètre `--no-apply-immediately`. Il permet de lancer la mise à niveau lors de la prochaine fenêtre de maintenance du cluster. La fenêtre de maintenance définit la période pendant laquelle les opérations de maintenance peuvent commencer. Une opération de longue durée ne doit pas nécessairement se terminer pendant la fenêtre de maintenance. Par conséquent, vous n'avez pas besoin de définir une fenêtre de maintenance plus grande, même si vous pensez que la mise à niveau peut prendre beaucoup de temps.

L'exemple suivant met à niveau un cluster qui exécute initialement Aurora MySQL version 2.11.2. Dans la sortie `describe-db-engine-versions`, les valeurs `False` et `True` représentent la propriété `IsMajorVersionUpgrade`. À partir de la version 2.11.2, vous pouvez effectuer une mise à niveau vers d'autres versions 2.*. Ces mises à niveau ne sont pas considérées comme des mises à niveau de version majeures et ne nécessitent donc pas de mise à niveau sur place. La mise à niveau sur place n'est disponible que pour les mises à niveau vers les versions 3.* répertoriées dans la liste.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster \
  --query '*[].{EngineVersion:EngineVersion}' --output text
5.7.mysql_aurora.2.11.2

aws rds describe-db-engine-versions --engine aurora-mysql --engine-version
 5.7.mysql_aurora.2.10.2 \
  --query '*[].[ValidUpgradeTarget]|[0][0]|[*].[EngineVersion,IsMajorVersionUpgrade]'
 --output text

5.7.mysql_aurora.2.11.3 False
5.7.mysql_aurora.2.11.4 False
5.7.mysql_aurora.2.11.5 False
5.7.mysql_aurora.2.11.6 False
5.7.mysql_aurora.2.12.0 False
5.7.mysql_aurora.2.12.1 False
5.7.mysql_aurora.2.12.2 False
5.7.mysql_aurora.2.12.3 False
8.0.mysql_aurora.3.04.0 True
8.0.mysql_aurora.3.04.1 True
8.0.mysql_aurora.3.04.2 True
```

```
8.0.mysql_aurora.3.04.3 True
8.0.mysql_aurora.3.05.2 True
8.0.mysql_aurora.3.06.0 True
8.0.mysql_aurora.3.06.1 True
8.0.mysql_aurora.3.07.1 True

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster \
  --engine-version 8.0.mysql_aurora.3.04.0 --no-apply-immediately --allow-major-
version-upgrade
...
```

Lorsqu'un cluster est créé sans fenêtre de maintenance spécifiée, Aurora choisit un jour de la semaine de manière aléatoire. Ici, la commande `modify-db-cluster` est soumise un lundi. Nous modifions la fenêtre de maintenance et la définissons sur mardi matin. Toutes les heures sont représentées dans le fuseau horaire UTC. La fenêtre `tue:10:00-tue:10:30` correspond à 2h00-2h30, heure du Pacifique. Les modifications de la fenêtre de maintenance prennent effet immédiatement.

```
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster --query '*[].
[PreferredMaintenanceWindow]'
[
    [
        "sat:08:20-sat:08:50"
    ]
]

aws rds modify-db-cluster --db-cluster-identifier mynewdbcluster --preferred-
maintenance-window tue:10:00-tue:10:30"
aws rds describe-db-clusters --db-cluster-identifier mynewdbcluster --query '*[].
[PreferredMaintenanceWindow]'
[
    [
        "tue:10:00-tue:10:30"
    ]
]
```

L'exemple suivant montre comment obtenir un rapport sur les événements générés par la mise à niveau. L'argument `--duration` représente le nombre de minutes nécessaire pour récupérer les informations d'événement. Cet argument est requis, car par défaut, `describe-events` renvoie uniquement les événements de la dernière heure.

```
aws rds describe-events --source-type db-cluster --source-identifier mynewdbcluster --
duration 20160
{
    "Events": [
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "DB cluster created",
            "EventCategories": [
                "creation"
            ],
            "Date": "2022-11-17T01:24:11.093000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Performing online pre-upgrade checks.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T22:57:08.450000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Performing offline pre-upgrade checks.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T22:57:59.519000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Creating pre-upgrade snapshot [preupgrade-
mynewdbcluster-5-7-mysql-aurora-2-10-2-to-8-0-mysql-aurora-3-02-0-2022-11-18-22-55].",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:00:22.318000+00:00",
```

```
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Cloning volume.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:01:45.428000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Purging undo records for old row versions.
Records remaining: 164",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:02:25.141000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Purging undo records for old row versions.
Records remaining: 164",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:06:23.036000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Upgrade in progress: Upgrading database objects.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:06:48.208000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        },
```

```
        {
            "SourceIdentifier": "mynewdbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster major version has been upgraded",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2022-11-18T23:10:28.999000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mynewdbcluster"
        }
    ]
}
```

Identification de la cause des échecs de mise à niveau de version majeure Aurora MySQL

Dans le didacticiel, la mise à niveau d'Aurora MySQL version 2 vers la version 3 a réussi. Toutefois, si la mise à niveau avait échoué, vous voudriez savoir pourquoi.

Pour ce faire, commencez par utiliser la commande `describe-events` de l'AWS CLI pour examiner les événements du cluster de bases de données. Cet exemple montre les événements liés à `mydbcluster` pendant les 10 dernières heures.

```
aws rds describe-events \
    --source-type db-cluster \
    --source-identifier mydbcluster \
    --duration 600
```

Dans ce cas, la vérification préalable de la mise à niveau a échoué.

```
{
    "Events": [
        {
            "SourceIdentifier": "mydbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster engine version upgrade started.",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2024-04-11T13:23:22.846000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster"
        },
        {
```

```
            "SourceIdentifier": "mydbcluster",
            "SourceType": "db-cluster",
            "Message": "Database cluster is in a state that cannot be upgraded: Upgrade
 prechecks failed. For more details, see the
             upgrade-prechecks.log file. For more information on troubleshooting the
 cause of the upgrade failure, see
             https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
AuroraMySQL.Upgrading.Troubleshooting.html",
            "EventCategories": [
                "maintenance"
            ],
            "Date": "2024-04-11T13:23:24.373000+00:00",
            "SourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster"
        }
    ]
}
```

Pour diagnostiquer la cause exacte du problème, examinez les journaux de base de données de l'instance de base de données d'enregistreur. Lorsqu'une mise à niveau vers Aurora MySQL 3 échoue, l'instance contient un fichier journal portant le nom `upgrade-prechecks.log`. Cet exemple montre comment détecter la présence de ce journal, puis comment le télécharger dans un fichier local pour examen.

```
aws rds describe-db-log-files --db-instance-identifier mydbcluster-instance \
    --query '*[].[LogFileName]' --output text

error/mysql-error-running.log
error/mysql-error-running.log.2024-04-11.20
error/mysql-error-running.log.2024-04-11.21
error/mysql-error.log
external/mysql-external.log
upgrade-prechecks.log

aws rds download-db-log-file-portion --db-instance-identifier mydbcluster-instance \
    --log-file-name upgrade-prechecks.log \
    --starting-token 0 \
    --output text >upgrade_prechecks.log
```

Le fichier `upgrade-prechecks.log` est au format JSON. Nous le téléchargeons à l'aide de l'option `--output text` afin d'éviter de coder la sortie JSON dans un autre encapsuleur JSON. Pour les mises à niveau d'Aurora MySQL version 3, ce journal inclut toujours certains messages d'information

et d'avertissement. Il inclut uniquement des messages d'erreur si la mise à niveau échoue. Si la mise à niveau réussit, le fichier journal n'est pas créé du tout.

Pour résumer toutes les erreurs et afficher les champs d'objet et de description associés, vous pouvez exécuter la commande `grep -A 2 '"level": "Error"'` sur le contenu du fichier `upgrade-prechecks.log`. Cela permet d'afficher chaque ligne d'erreur et les deux lignes qui la suivent. Elles contiennent le nom de l'objet de base de données correspondant et des conseils sur la façon de corriger le problème.

```
$ cat upgrade-prechecks.log | grep -A 2 '"level": "Error"'

"level": "Error",
"dbObject": "problematic_upgrade.dangling_fulltext_index",
"description": "Table `problematic_upgrade.dangling_fulltext_index` contains dangling
 FULLTEXT index. Kindly recreate the table before upgrade."
```

Dans cet exemple, vous pouvez exécuter la commande SQL suivante sur la table en cause pour tenter de résoudre le problème, ou vous pouvez recréer la table sans l'index suspendu.

```
OPTIMIZE TABLE problematic_upgrade.dangling_fulltext_index;
```

Réessayez ensuite la mise à niveau.

Dépannage de la mise à niveau sur place d'Aurora MySQL

Suivez les conseils suivants pour résoudre les problèmes liés aux mises à niveau sur place d'Aurora MySQL. Ces conseils ne s'appliquent pas aux clusters de bases de données Aurora Serverless.

Pourquoi la mise à niveau sur place s'est arrêtée ou est lente	Effet	Solution permettant à la mise à niveau sur place de se terminer dans la fenêtre de maintenance
Le réplica entre régions d'Aurora associé n'a pas encore reçu les correctifs requis	Aurora annule la mise à niveau.	Mettez à niveau le réplica entre régions d'Aurora et réessayez.
Le cluster a des transactions XA à l'état préparé.	Aurora annule	Validez ou restaurez toutes les transactions XA préparées.

Pourquoi la mise à niveau sur place s’est arrêtée ou est lente	Effet	Solution permettant à la mise à niveau sur place de se terminer dans la fenêtre de maintenance
	la mise à niveau.	
Le cluster traite toutes les instructions en langage de définition de données (DDL).	Aurora annule la mise à niveau.	Pensez à attendre et à effectuer la mise à niveau une fois toutes les instructions DDL terminées.
Le cluster a des modificat ions non validées pour de nombreuses lignes.	La mise à niveau pourrait prendre beaucoup de temps.	Le processus de mise à niveau restaure les modifications non validées. L’indicateur de cette condition est la valeur de `TRX_ROWS_ MODIFIED` dans la table `INFORMATI ON_SCHEMA.INNODB_TRX` . Prévoyez d’effectuer la mise à niveau uniquement lorsque toutes les transactions volumineuses ont été validées ou restaurées.

Pourquoi la mise à niveau sur place s’est arrêtée ou est lente	Effet	Solution permettant à la mise à niveau sur place de se terminer dans la fenêtre de maintenance
Le cluster a un nombre élevé d’enregistrements d’annulation.	La mise à niveau pourrait prendre beaucoup de temps.	Même si les transactions non validées affectent peu de lignes, elles peuvent impliquer un grand volume de données. Par exemple, si vous insérez des objets BLOB volumineux, Aurora ne détecte ni ne génère automatiquement d’événement pour ce type d’activité de transaction. L’indicateur de cette condition est la longueur de la liste d’historique (HLL). Le processus de mise à niveau restaure les modifications non validées. Vous pouvez vérifier la liste HLL dans la sortie de la commande SQL `SHOW ENGINE INNODB STATUS` ou directement à l’aide de la requête SQL suivante : `SELECT count FROM information_schema .innodb_metrics WHERE name = 'trx_rseg_history_len';` Vous pouvez également utiliser la métrique `RollbackSegmentHistoryListLength` dans Amazon CloudWatch. Prévoyez d’effectuer la mise à niveau uniquement une fois que la longueur de la liste HLL sera plus petite.

Pourquoi la mise à niveau sur place s’est arrêtée ou est lente	Effet	Solution permettant à la mise à niveau sur place de se terminer dans la fenêtre de maintenance
Le cluster est en train de valider une transaction de journal binaire volumineuse	La mise à niveau pourrait prendre beaucoup de temps.	Le processus de mise à niveau attend que les modifications de journaux binaires soient appliquées. Plus de transactions ou d’instructions DDL pourraient commencer pendant cette période, ce qui ralentirait encore le processus de mise à niveau. Planifiez le processus de mise à niveau lorsque le cluster n’est pas occupé à générer des modifications de réplication de journaux binaires. Aurora ne détecte ni ne génère automatiquement d’événement pour cette condition.

Pourquoi la mise à niveau sur place s'est arrêtée ou est lente	Effet	Solution permettant à la mise à niveau sur place de se terminer dans la fenêtre de maintenance
Incohérences de schéma résultant de la suppression ou de l'endommagement de fichiers	Aurora annule la mise à niveau.	Changez le moteur de stockage par défaut pour les tables temporaires de MyISAM à InnoDB. Procédez comme suit : 1. Modifiez le paramètre de base de données `default_tmp_storage_engine` en spécifiant `InnoDB`. 2. Redémarrez le cluster de bases de données. 3. Après le redémarrage, confirmez que le paramètre de base de données `default_tmp_storage_engine` est défini sur `InnoDB`. Utilisez la commande suivante : `show global variables like 'default_tmp_storage_engine';` 4. Veillez à ne pas créer de tables temporaires utilisant le moteur de stockage MyISAM. Nous vous recommandons de suspendre toute charge de travail de base de données et de ne pas créer de nouvelles connexions de base de données, car vous allez bientôt effectuer une mise à niveau. 5. Réessayez la mise à niveau sur place.

Pourquoi la mise à niveau sur place s'est arrêtée ou est lente	Effet	Solution permettant à la mise à niveau sur place de se terminer dans la fenêtre de maintenance
L'utilisateur principal a été supprimé	Aurora annule la mise à niveau.	**Important** Ne supprimez pas l'utilisateur principal. Toutefois, si, pour une raison ou une autre, vous venez à supprimer l'utilisateur principal , restaurez-le à l'aide des commandes SQL suivantes : `CREATE USER 'master_username '@'%' IDENTIFIED BY ' master_user_password ' REQUIRE NONE PASSWORD EXPIRE DEFAULT ACCOUNT UNLOCK;` `GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES, CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER, LOAD FROM S3, SELECT INTO S3, INVOKE LAMBDA, INVOKE SAGEMAKER , INVOKE COMPREHEND ON *.* TO 'master_username '@'%' WITH GRANT OPTION;`

Pour plus de détails sur la résolution des problèmes à l'origine de l'échec des vérifications préalables de la mise à niveau, consultez les blogs suivants :

- Liste de contrôle de mise à niveau d'Amazon Aurora MySQL version 2 (avec compatibilité MySQL 5.7) vers la version 3 (avec compatibilité MySQL 8.0), partie 1

- Liste de contrôle de mise à niveau d'Amazon Aurora MySQL version 2 (avec compatibilité MySQL 5.7) vers la version 3 (avec compatibilité MySQL 8.0), partie 2

Vous pouvez suivre les étapes suivantes afin d'effectuer vos propres vérifications pour certaines des conditions du tableau précédent. Ainsi, vous pouvez planifier la mise à niveau à un moment où vous savez que la base de données sera dans un état permettant une mise à niveau rapide.

- Vous pouvez vérifier les transactions XA ouvertes en exécutant l'instruction `XA RECOVER`. Vous pouvez ensuite valider ou restaurer les transactions XA avant de lancer la mise à niveau.
- Vous pouvez vérifier les instructions DDL en exécutant une instruction `SHOW PROCESSLIST` et en recherchant les instructions `CREATEDROP`, `ALTER`, `RENAME` et `TRUNCATE` dans la sortie. Laissez toutes les instructions DDL se terminer avant de commencer la mise à niveau.
- Vous pouvez vérifier le nombre total de lignes non validées en interrogeant la table `INFORMATION_SCHEMA.INNODB_TRX`. La table contient une ligne pour chaque transaction. La colonne `TRX_ROWS_MODIFIED` contient le nombre de lignes modifiées ou insérées par la transaction.
- Vous pouvez vérifier la longueur de la liste d'historique InnoDB en exécutant l'instruction `SHOW ENGINE INNODB STATUS SQL` et en recherchant la valeur `History list length` dans la sortie. Vous pouvez également vérifier la valeur directement en exécutant la requête suivante :

  ```
  SELECT count FROM information_schema.innodb_metrics WHERE name =
   'trx_rseg_history_len';
  ```

  La longueur de la liste d'historique correspond à la quantité d'informations d'annulation stockées par la base de données pour implémenter le contrôle de concurrence multi-version (MVCC).

Nettoyage postérieur à la mise à niveau pour Aurora MySQL version 3

Une fois que vous avez terminé de mettre à niveau un cluster Aurora MySQL version 2 vers Aurora MySQL version 3, vous pouvez effectuer les autres actions de nettoyage suivantes :

- Créez de nouvelles versions compatibles avec MySQL 8.0 de tous les groupes de paramètres personnalisés. Appliquez toutes les valeurs de paramètres personnalisés nécessaires aux nouveaux groupes de paramètres.
- Mettez à jour toutes les alarmes CloudWatch, les scripts de configuration, etc. pour utiliser les nouveaux noms pour toutes les métriques dont les noms ont été affectés par des changements

linguistiques inclusifs. Pour connaître la liste des métriques concernées, consultez Changements linguistiques inclusifs pour Aurora MySQL version 3.

- Mettez à jour tous les modèles CloudFormation pour utiliser les nouveaux noms pour tous les paramètres de configuration dont les noms ont été affectés par des modifications linguistiques inclusives. Pour obtenir la liste des paramètres convernés, consultez Changements linguistiques inclusifs pour Aurora MySQL version 3.

Index spatiaux

Après la mise à niveau vers Aurora MySQL version 3, vérifiez si vous devez supprimer ou recréer des objets et des index liés aux index spatiaux. Avant MySQL 8.0, Aurora pouvait optimiser les requêtes spatiales à l'aide d'index qui ne contenaient pas d'identifiant de ressource spatiale (SRID). Aurora MySQL version 3 utilise uniquement des index spatiaux contenant des SRID. Lors d'une mise à niveau, Aurora supprime automatiquement tous les index spatiaux sans SRID et imprime des messages d'avertissement dans le journal de base de données. Si vous observez de tels messages d'avertissement, créez de nouveaux index spatiaux avec des SRID après la mise à niveau. Pour plus d'informations sur les modifications apportées aux fonctions spatiales et les types de données dans MySQL 8.0, consultez Changements dans MySQL 8.0dans le manuel de référence MySQL.

Mises à jour et correctifs du moteur de base de données pour Amazon Aurora MySQL

Vous trouverez les informations suivantes dans Notes de mise à jour d'Amazon Aurora Édition compatible avec MySQL :

- Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 3
- Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 2
- Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 1 (obsolète)
- Bogues MySQL corrigés par les mises à jour du moteur de base de données Aurora MySQL
- Vulnérabilités de sécurité corrigées dans Amazon Aurora MySQL

Utilisation de Amazon Aurora PostgreSQL

Amazon Aurora PostgreSQL est un moteur de base de données relationnelle entièrement géré, compatible avec PostgreSQL et conforme à la norme ACID, qui associe la vitesse, la fiabilité et la facilité de gestion d'Amazon Aurora à la simplicité et à la rentabilité des bases de données open source. Aurora PostgreSQL est une solution alternative pour MySQL, qui vous permet de configurer, gérer et mettre à l'échelle de façon simple et économique vos déploiements PostgreSQL existants et nouveaux, de façon à ce que vous puissiez vous concentrer sur votre activité et vos applications. Pour en savoir plus sur Aurora en général, consultez Qu'est-ce qu'Amazon Aurora ?.

Outre les avantages d'Aurora, Aurora PostgreSQL offre une voie de migration pratique d'Amazon RDS vers Aurora, avec des outils de migration à bouton-poussoir qui convertissent vos applications RDS pour PostgreSQL existantes en Aurora PostgreSQL. Les tâches courantes liées aux bases de données, telles que l'approvisionnement, l'application de correctifs, la sauvegarde, la récupération, la détection des pannes et la réparation, sont également faciles à gérer avec Aurora PostgreSQL.

Aurora PostgreSQL est compatible avec de nombreuses normes du secteur. Par exemple, vous pouvez utiliser les bases de données Aurora PostgreSQL afin de développer des applications conformes HIPAA et de stocker les informations relatives à la santé, y compris les données relatives aux informations de santé protégées (PHI ou Protected Health Information) selon les termes d'un accord BAA (ou Business Associate Agreement) conclu avec AWS.

Aurora PostgreSQL est éligible FedRAMP HIGH. Pour plus de détails sur AWS les efforts de conformité et les efforts de conformité, consultez la section AWS Services concernés par programme de conformité.

> **Important**
>
> Si vous rencontrez un problème avec votre cluster de bases de données Aurora PostgreSQL, AWS votre agent de support peut avoir besoin de plus d'informations sur l'état de vos bases de données. L'objectif est de garantir que le AWS Support obtienne le plus d'informations requises dans les plus brefs délais.
> Vous pouvez utiliser PG Collector pour rassembler de précieuses informations de base de données dans un fichier HTML consolidé. Pour plus d'informations sur PG Collector, comment l'exécuter et comment télécharger le rapport HTML, consultez PG Collector.
> En cas d'exécution réussie et sauf indication contraire, le script renvoie la sortie dans un format HTML lisible. Ce script est conçu pour exclure toutes les données ou informations de sécurité du contenu HTML susceptible de compromettre votre activité. De plus, il n'apporte

aucune modification à la base de données ou à son environnement. Toutefois, si vous trouvez dans le contenu HTML des informations que vous ne souhaitez pas partager, n'hésitez pas à supprimer les informations problématiques avant de charger le contenu HTML. Lorsque le contenu HTML est acceptable, chargez-le dans la section des pièces jointes des détails du cas de votre dossier de support.

Rubriques

- Mises à jour du moteur de base de données pour Amazon Aurora PostgreSQL

Utilisation de l'environnement de version préliminaire de base de données

La communauté PostgreSQL publie chaque année une nouvelle version majeure de PostgreSQL. De même, Amazon Aurora propose des versions majeures de PostgreSQL sous forme de versions préliminaires. Cela vous permet de créer des clusters de bases de données à l'aide de la préversion et de tester ses fonctionnalités dans l'environnement de version préliminaire de base de données.

Les clusters de bases de données Aurora MySQL dans l'environnement de version préliminaire de base de données sont similaires sur le plan fonctionnel à d'autres clusters de bases de données Aurora MySQL. Toutefois, vous ne pouvez pas utiliser une préversion pour la production.

Retenez bien les limites importantes suivantes :

- Toutes les instances de base de données et de clusters de bases de données sont supprimées 60 jours après leur création, en même temps que leurs sauvegardes et leurs instantanés.
- Vous ne pouvez créer une instance de base de données que dans un cloud privé virtuel (VPC) basé sur un service Amazon VPC.
- Vous ne pouvez pas copier un instantané d'instance de base de données dans un environnement de production.

Les options suivantes sont prises en charge par la préversion.

- Vous pouvez créer des instances de base de données à l'aide des types d'instance r5, r6g, r6i, r7g, r7i, r8g, x2g, t3 et t4g uniquement. Pour plus d'informations sur les classes d'instance, consultez Classes d'instance de base de données Amazon Aurora.
- Vous pouvez utiliser à la fois des déploiements mono-AZ et multi-AZ.
- Vous pouvez utiliser les fonctions de vidage et de chargement PostgreSQL standard pour exporter des bases de données depuis ou importer des bases de données vers l'environnement de version préliminaire de base de données.

Types de classes d’instance de bases de données pris en charge

Amazon Aurora PostgreSQL prend en charge les classes d’instance de bases de données suivantes dans la région en version préliminaire :

Classes optimisées pour la mémoire

- db.r5 : classes d’instance à mémoire optimisée
- db.r6g : classes d’instance à mémoire optimisée alimentées par des processeurs AWS Graviton2
- db.r6i : classes d’instance à mémoire optimisée
- db.r7g : classes d’instance à mémoire optimisée alimentées par des processeurs AWS Graviton3
- db.r7i : classes d’instance à mémoire optimisée
- db.r8g : classes d’instance à mémoire optimisée alimentées par des processeurs AWS Graviton4
- db.x2g : classes d’instance à mémoire optimisée alimentées par des processeurs AWS Graviton2

> **Note**
>
> Pour plus d’informations sur la liste des classes d’instance, reportez-vous à Types de classes d’instance de base de données.

Classes pouvant être émises en rafale

- db.t3.medium
- db.t3.large
- db.t4g.medium
- db.t4g.large

Fonctionnalités non prises en charge dans l’environnement en version préliminaire

Les fonctions suivantes ne sont pas disponibles dans l’environnement en préversion :

- Aurora Serverless v1 et v2
- Mises à niveau de version majeure (MVU)

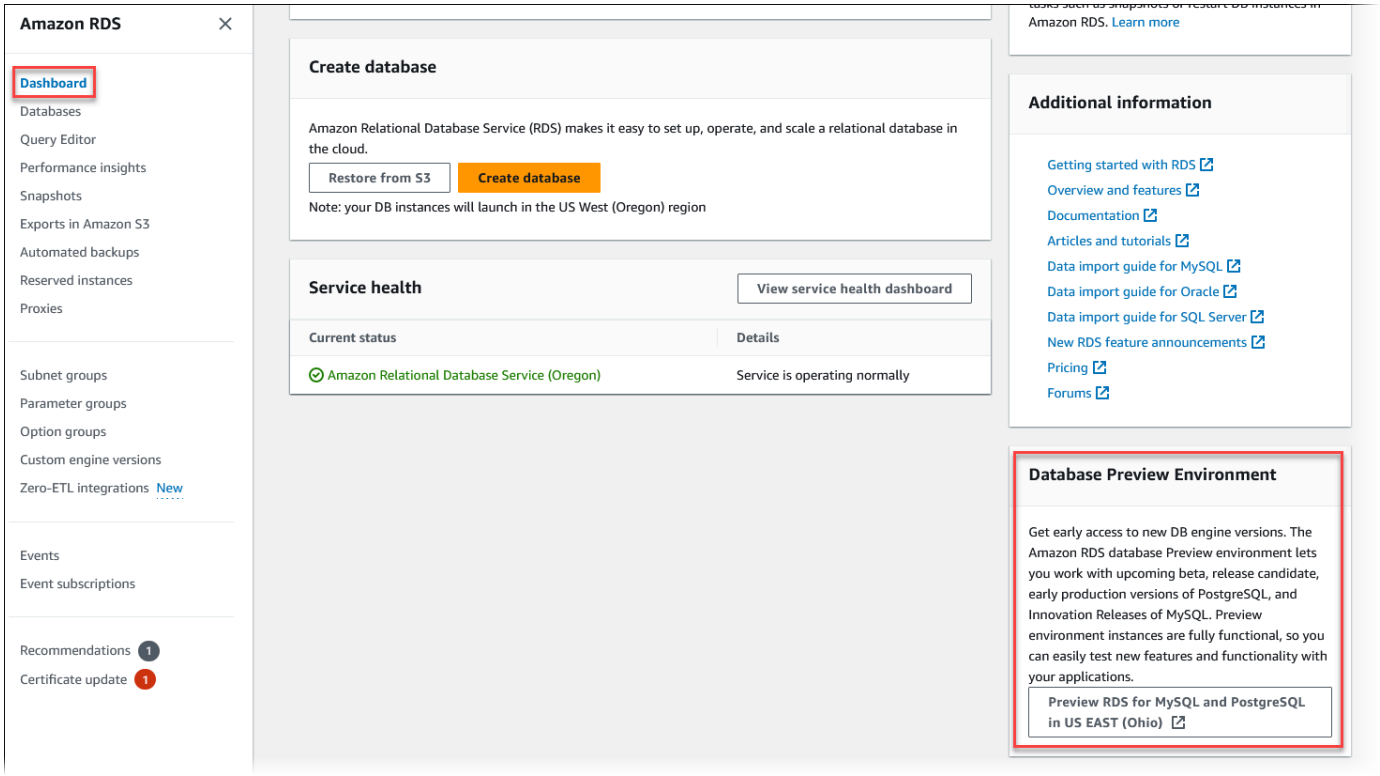
- Aucune nouvelle version mineure ne sera publiée dans la région en version préliminaire
- Réplication entrante de RDS pour PostgreSQL vers Aurora PostgreSQL
- Déploiement bleu/vert Amazon RDS
- Copie d'instantanés entre Régions
- Aurora Global Database
- Flux d'activité de base de données (DAS), RDS Proxy et AWS DMS
- Réplicas en lecture d'autoscaling
- AWS Bedrock
- Exporter au format RDS
- Performance Insights
- Points de terminaison personnalisés
- Copie d'un instantané
- zéro ETL
- Babelfish
- Toutes les extensions

Création d'un nouveau cluster de bases de données dans l'environnement en préversion

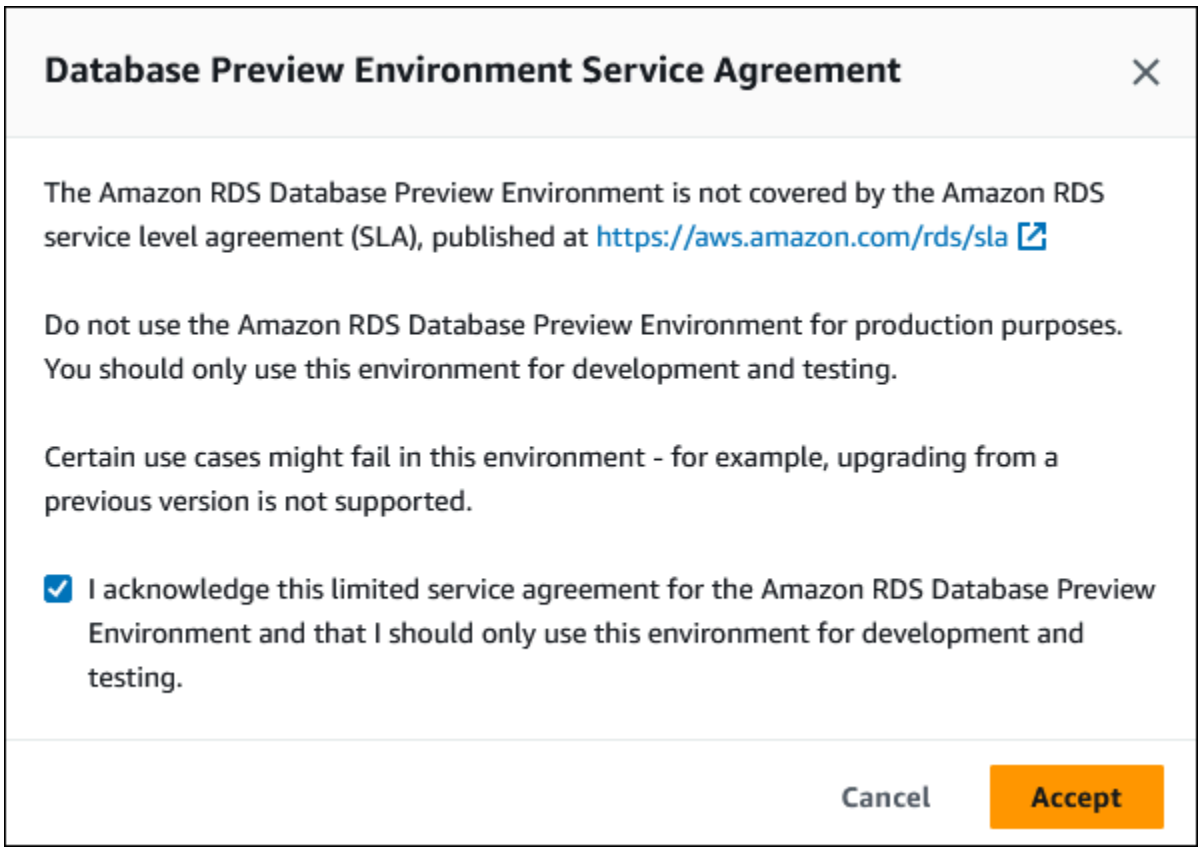
Utilisez la procédure suivante pour créer une instance de base de données dans l'environnement en version préliminaire.

Pour créer un cluster de bases de données dans l'environnement en version préliminaire

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Dashboard (Tableau de bord) dans le panneau de navigation.
3. Sur la page Tableau de bord, recherchez la section Environnement de version préliminaire de base de données, comme illustré dans l'image suivante.



Vous pouvez accéder directement à l'environnement de version préliminaire de base de données. Avant de poursuivre, vous devez reconnaître et accepter les limites.



4. Pour créer le cluster de bases de données Aurora MySQL, suivez le même processus que pour créer n'importe quel cluster de bases de données Aurora. Pour plus d'informations, consultez Création d'un cluster de bases de données Amazon Aurora.

Pour créer une instance dans l'environnement de version préliminaire de base de données via l'API Aurora l'AWS CLI, utilisez le point de terminaison suivant.

```
rds-preview.us-east-2.amazonaws.com
```

PostgreSQL version 17 dans l'environnement de version préliminaire de base de données

⚠ Il s'agit de la documentation en version préliminaire pour Aurora PostgreSQL version 17. Elle est susceptible d'être modifiée.

PostgreSQL version 17.0 est maintenant disponible dans l'environnement de version préliminaire de base de données Amazon RDS. PostgreSQL version 17 contient plusieurs améliorations qui sont décrites dans la documentation PostgreSQL suivante :

- Publication de PostgreSQL

Pour plus d'informations sur l'environnement de version préliminaire de base de données, consultez Utilisation de l'environnement de version préliminaire de base de données. Pour accéder à l'environnement en préversion à partir de la console, sélectionnez https://console.aws.amazon.com/rds-preview/.

Sécurité avec Amazon Aurora PostgreSQL

Pour obtenir une présentation générale de la sécurité Aurora, consultez Sécurité dans Amazon Aurora. Vous pouvez gérer la sécurité d'Amazon Aurora PostgreSQL à différents niveaux :

- Pour contrôler qui peut effectuer des actions de gestion Amazon RDS sur les clusters de bases de données et les instances de base de données Aurora PostgreSQL, Gestion des identités et des accès AWS utilisez (IAM). IAM gère l'authentification de l'identité de l'utilisateur avant que l'utilisateur puisse accéder au service. Il gère également l'autorisation, c'est-à-dire si l'utilisateur est autorisé à faire ce qu'il essaie de faire. L'authentification de base de données IAM est une méthode d'authentification supplémentaire que vous pouvez choisir lorsque vous créez votre cluster de bases de données Aurora PostgreSQL. Pour de plus amples informations, veuillez consulter Identity and Access Management pour Amazon Aurora.

  Si vous utilisez IAM avec votre cluster de base de données Aurora PostgreSQL, connectez-vous d'abord à l'aide de vos informations d'identification IAM, avant d'ouvrir AWS Management Console la console Amazon RDS à l'adresse. https://console.aws.amazon.com/rds/
- Les clusters de bases de données Aurora doivent être créés dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC. Pour contrôler quels appareils et EC2 instances Amazon peuvent ouvrir des connexions au point de terminaison et au port de l'instance de base de données pour les clusters de base de données Aurora dans un VPC, utilisez un groupe de sécurité VPC. Ces connexions au point de terminaison et au port peuvent être effectuées à l'aide du protocole SSL (Secure Socket Layer). En outre, les règles de pare-feu de votre entreprise peuvent contrôler si les appareils en cours d'exécution dans votre entreprise peuvent ouvrir des connexions à une instance de base de données. Pour plus d'informations sur VPCs, voirAmazon VPC et Amazon Aurora.

La location de VPC prise en charge dépend de la classe d'instance de base de données utilisée par vos clusters de bases de données Aurora PostgreSQL. Avec la location de VPC `default`, le cluster de bases de données s'exécute sur du matériel partagé. Avec la location de VPC `dedicated`, le cluster de bases de données s'exécute sur une instance matérielle dédiée. Les classes d'instance de base de données à performances extensibles prennent uniquement en charge la location de VPC par défaut. Les classes d'instance de base de données à performances extensibles incluent les classes d'instance de base de données db.t3 et db.t4g. Toutes les autres classes d'instance de base de données Aurora PostgreSQL prennent en charge à la fois la location de VPC par défaut et dédiée.

Pour plus d'informations sur les classes d'instance, consultez Classes d'instance de base de données Amazon Aurora. Pour plus d'informations sur la location de VPC `default` et `dedicated`, consultez Instances dédiées dans le Guide de l'utilisateur Amazon Elastic Compute Cloud.

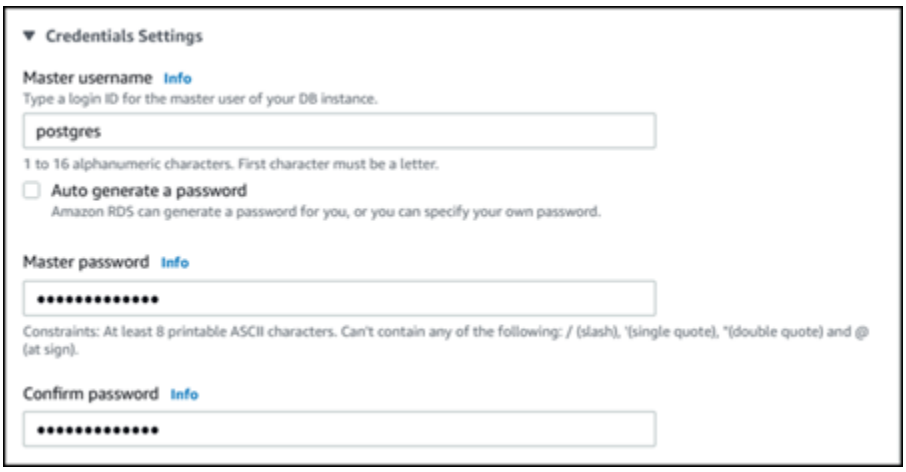
- Pour accorder des autorisations aux bases de données PostgreSQL exécutées sur votre cluster de bases de données Amazon Aurora, vous pouvez adopter la même approche générale que pour les instances autonomes de PostgreSQL. Les commandes telles que `CREATE ROLE`, `ALTER ROLE`, `GRANT` et `REVOKE` fonctionnent de la même façon que dans les bases de données sur site, comme le fait la modification directe des bases de données, schémas et tables.

  PostgreSQL gère les privilèges en utilisant des rôles. Le rôle `rds_superuser` est le rôle disposant du plus haut niveau de privilèges sur un cluster de bases de données Aurora PostgreSQL. Ce rôle est créé automatiquement et il est accordé à l'utilisateur qui crée le cluster de bases de données (le compte utilisateur principal, `postgres` par défaut). Pour en savoir plus, veuillez consulter la section Comprendre les rôles et les autorisations PostgreSQL.

Toutes les versions disponibles d'Aurora PostgreSQL, y compris les versions 10, 11, 12, 13, 14 et supérieures, prennent en charge le mécanisme d'authentification Salted Challenge Response Authentication (SCRAM) pour les mots de passe comme alternative à message digest (). MD5 Nous vous recommandons d'utiliser SCRAM car il est plus sûr que MD5. Pour plus d'informations, notamment sur la façon de migrer les mots de passe des utilisateurs de base de données MD5 vers SCRAM, consultezUtilisation de SCRAM pour le chiffrement de mot de passe PostgreSQL.

Comprendre les rôles et les autorisations PostgreSQL

Lorsque vous créez un cluster de bases de données Aurora PostgreSQL à l'aide d'AWS Management Console, un compte administrateur est créé simultanément. Par défaut, son nom est `postgres`, comme illustré dans la capture d'écran ci-dessous :



Vous pouvez choisir un autre nom plutôt que d'accepter le nom par défaut (`postgres`). Le nom que vous choisissez doit commencer par une lettre et comporter entre 1 et 16 caractères alphanumériques. Par souci de simplicité, nous nous référons à ce compte utilisateur principal par sa valeur par défaut (`postgres`) tout au long de ce manuel.

Si vous utilisez AWS CLI `create-db-cluster` plutôt qu'AWS Management Console, vous créez le nom d'utilisateur en le transmettant avec le paramètre `master-username`. Pour plus d'informations, consultez Étape 2 : Créer un cluster de bases de données Aurora PostgreSQL.

Que vous utilisiez AWS Management Console, AWS CLI ou l'API Amazon RDS, ou que vous utilisiez le nom `postgres` par défaut ou un autre nom, ce premier compte utilisateur de base de données est membre du groupe `rds_superuser` et possède des privilèges `rds_superuser`.

Rubriques

- Comprendre le rôle rds_superuser
- Contrôle de l'accès utilisateur à la base de données PostgreSQL
- Délégation et contrôle de la gestion des mots de passe utilisateur
- Utilisation de SCRAM pour le chiffrement de mot de passe PostgreSQL

Comprendre le rôle rds_superuser

Dans PostgreSQL, un rôle peut définir un utilisateur, un groupe ou un ensemble d'autorisations spécifiques accordées à un groupe ou à un utilisateur pour divers objets de la base de données. Les commandes PostgreSQL `CREATE USER` et `CREATE GROUP` ont été remplacées par la commande `CREATE ROLE` plus générique avec des propriétés spécifiques permettant de distinguer les utilisateurs de la base de données. Un utilisateur de base de données peut être considéré comme un rôle disposant du privilège LOGIN.

> **Note**
>
> Les commandes `CREATE USER` et `CREATE GROUP` peuvent toujours être utilisées. Pour plus d'informations, consultez Database Roles dans la documentation de PostgreSQL.

L'utilisateur `postgres` est l'utilisateur de base de données disposant des privilèges les plus élevés sur votre cluster de bases de données Aurora PostgreSQL. Il présente les caractéristiques définies par l'instruction `CREATE ROLE` suivante.

```
CREATE ROLE postgres WITH LOGIN NOSUPERUSER INHERIT CREATEDB CREATEROLE NOREPLICATION
 VALID UNTIL 'infinity'
```

Sauf indication contraire, les propriétés `NOSUPERUSER`, `NOREPLICATION`, `INHERIT` et `VALID UNTIL 'infinity'` sont les options par défaut de CREATE ROLE.

Par défaut, `postgres` fait en sorte que des privilèges soient octroyés au rôle `rds_superuser` ainsi que des autorisations permettant de créer des rôles et des bases de données. Le rôle `rds_superuser` permet à l'utilisateur `postgres` d'effectuer les opérations suivantes :

- Ajoutez les extensions qu'il est possible d'utiliser avec Aurora PostgreSQL. Pour plus d'informations, consultez Utilisation d'extensions avec encapsuleurs de données externes.
- Créer des rôles pour les utilisateurs et leur accorder des privilèges. Pour plus d'informations, consultez CREATE ROLE et GRANT dans la documentation de PostgreSQL.
- Créer des bases de données. Pour plus d'informations, consultez CREATE DATABASE dans la documentation de PostgreSQL.
- Accorder des privilèges `rds_superuser` aux rôles utilisateur qui ne disposent pas de ces privilèges, et révoquer les privilèges si nécessaire. Nous vous recommandons d'accorder ce rôle

uniquement aux utilisateurs effectuant des tâches de super-utilisateur. En d'autres termes, vous pouvez accorder ce rôle aux administrateurs de base de données (DBA) ou aux administrateurs système.

- Accorder (et révoquer) le rôle `rds_replication` aux utilisateurs de base de données qui ne possèdent pas le rôle `rds_superuser`.
- Accorder (et révoquer) le rôle `rds_password` aux utilisateurs de base de données qui ne possèdent pas le rôle `rds_superuser`.
- Obtenir des informations d'état sur toutes les connexions à la base de données en utilisant la vue `pg_stat_activity`. En cas de besoin, `rds_superuser` peut arrêter toutes les connexions à l'aide de `pg_terminate_backend` ou `pg_cancel_backend`.

Dans l'instruction `CREATE ROLE postgres...`, vous pouvez voir que le rôle utilisateur `postgres` rejette spécifiquement les autorisations `superuser` PostgreSQL. Aurora PostgreSQL étant un service géré, vous ne pouvez ni accéder au système d'exploitation hôte, ni vous connecter à l'aide du compte `superuser` PostgreSQL. La plupart des tâches qui exigent un accès `superuser` sur une instance autonome de PostgreSQL sont gérées automatiquement par Aurora.

Pour plus d'informations sur l'octroi de privilèges, consultez GRANT dans la documentation de PostgreSQL.

Le rôle `rds_superuser` est l'un des nombreux rôles prédéfinis d'un cluster de bases de données Aurora PostgreSQL.

> **Note**
>
> Dans PostgreSQL 13 et versions antérieures, les rôles prédéfinis s'appellent rôles par défaut.

La liste suivante répertorie certains des autres rôles prédéfinis créés automatiquement pour un nouveau cluster de bases de données Aurora PostgreSQL. Les rôles prédéfinis et leurs privilèges ne peuvent pas être modifiés. Vous ne pouvez pas supprimer, renommer ou modifier les privilèges de ces rôles prédéfinis. Toute tentative de ce type génère une erreur.

- rds_password : rôle pouvant modifier les mots de passe et configurer des contraintes de mot de passe pour les utilisateurs de base de données. Le rôle `rds_superuser` se voit accorder ce rôle par défaut et peut accorder le rôle aux utilisateurs de base de données. Pour plus d'informations, consultez Contrôle de l'accès utilisateur à la base de données PostgreSQL.

  - Pour les versions de RDS pour PostgreSQL antérieures à 14, le rôle `rds_password` peut modifier les mots de passe et configurer des contraintes de mot de passe pour les utilisateurs de bases de données et les utilisateurs ayant un rôle `rds_superuser`. À partir de RDS pour PostgreSQL versions 14 et ultérieures, le rôle `rds_password` peut modifier les mots de passe et configurer des contraintes de mot de passe uniquement pour les utilisateurs de bases de données. Seuls les utilisateurs ayant le rôle `rds_superuser` peuvent effectuer ces actions sur d'autres utilisateurs ayant le rôle `rds_superuser`.
- rdsadmin : rôle créé pour gérer la plupart des tâches de gestion que l'administrateur qui utilise les privilèges `superuser` aurait exécutées sur une base de données PostgreSQL autonome. Ce rôle est utilisé en interne par Aurora PostgreSQL pour de nombreuses tâches de gestion.

Affichage des rôles et de leurs privilèges

Vous pouvez afficher les rôles prédéfinis et leurs privilèges dans votre instance de base de données RDS pour PostgreSQL à l'aide de différentes commandes en fonction de votre version de PostgreSQL. Pour voir tous les rôles prédéfinis, vous pouvez vous connecter à votre instance de base de données RDS pour PostgreSQL et exécuter les commandes suivantes à l'aide de `psql`.

Pour **`psql`** 15 et versions antérieures

Connectez-vous à votre instance de base de données RDS pour PostgreSQL et utilisez la commande \du dans psql :

```
postgres=> \du
                                                    List of roles
    Role name    |                         Attributes                          |
                  Member of
-----------------+-----------------------------------------------------------
+-------------------------------------------------
 postgres        | Create role, Create DB                                      +|
 {rds_superuser}
                 | Password valid until infinity                               |
 rds_ad          | Cannot login                                                | {}
 rds_iam         | Cannot login                                                | {}
 rds_password    | Cannot login                                                | {}
 rds_replication | Cannot login                                                | {}
 rds_superuser   | Cannot login                                                |
 {pg_monitor,pg_signal_backend,rds_password,rds_replication}
 rdsadmin        | Superuser, Create role, Create DB, Replication, Bypass RLS+| {}
                 | Password valid until infinity                               |
```

Pour **`psql`** 16 et versions ultérieures

```
postgres=> \drg+
                            List of role grants
   Role name   |          Member of          |       Options       | Grantor
---------------+-----------------------------+---------------------+----------
 postgres      | rds_superuser               | INHERIT, SET        | rdsadmin
 rds_superuser | pg_checkpoint               | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_monitor                  | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_signal_backend           | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | pg_use_reserved_connections | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_password                | ADMIN, INHERIT, SET | rdsadmin
 rds_superuser | rds_replication             | ADMIN, INHERIT, SET | rdsadmin
```

Pour vérifier l'appartenance à un rôle sans dépendance de version, vous pouvez utiliser la requête SQL suivante :

```
SELECT m.rolname AS "Role name", r.rolname AS "Member of"
FROM pg_catalog.pg_roles m
JOIN pg_catalog.pg_auth_members pam ON (pam.member = m.oid)
LEFT JOIN pg_catalog.pg_roles r ON (pam.roleid = r.oid)
LEFT JOIN pg_catalog.pg_roles g ON (pam.grantor = g.oid)
WHERE m.rolname !~ '^pg_'
ORDER BY 1, 2;
```

Dans la sortie, vous pouvez voir que `rds_superuser` n'est pas un rôle utilisateur de base de données (il ne peut pas se connecter), mais qu'il dispose des privilèges de nombreux autres rôles. Vous pouvez également voir que l'utilisateur de base de données `postgres` est membre du rôle `rds_superuser`. Comme mentionné précédemment, `postgres` est la valeur par défaut sur la page Create database (Créer une base de données) de la console Amazon RDS. Si vous avez choisi un autre nom, ce nom apparaît dans la liste des rôles.

> **Note**
>
> Les versions 15.2 et 14.7 d'Aurora PostgreSQL ont introduit un comportement restrictif du rôle `rds_superuser`. Un utilisateur Aurora PostgreSQL doit se voir accorder le privilège `CONNECT` sur la base de données correspondante pour se connecter même si l'utilisateur dispose du rôle `rds_superuser`. Avant les versions 14.7 et 15.2 d'Aurora PostgreSQL, un utilisateur pouvait se connecter à n'importe quelle base de données ou table système s'il

bénéficiait du rôle `rds_superuser`. Ce comportement restrictif s'aligne sur les engagements d'AWS et d'Amazon Aurora en faveur de l'amélioration continue de la sécurité.
Mettez à jour la logique correspondante dans vos applications si l'amélioration ci-dessus a un impact.

Contrôle de l'accès utilisateur à la base de données PostgreSQL

Les nouvelles bases de données de PostgreSQL sont toujours créées avec un ensemble de privilèges par défaut dans le schéma `public` de la base de données, qui permet à tous les utilisateurs et rôles de base de données de créer des objets. Ces privilèges permettent aux utilisateurs de base de données de se connecter à la base de données, par exemple, et de créer des tables temporaires lorsqu'ils sont connectés.

Pour mieux contrôler l'accès des utilisateurs aux instances de base de données que vous créez sur le nœud primaire de votre cluster de bases de données Aurora PostgreSQL, nous vous recommandons de révoquer ces privilèges `public` par défaut. Vous accordez ensuite des privilèges spécifiques aux utilisateurs de base de données de manière plus détaillée, comme indiqué dans la procédure suivante.

Pour configurer des rôles et des privilèges pour une nouvelle instance de base de données

Supposons que vous configuriez une base de données sur un cluster de bases de données Aurora PostgreSQL récemment créé à l'usage de plusieurs chercheurs, qui ont tous besoin d'un accès en lecture/écriture à la base de données.

1. Utilisez `psql` (ou pgAdmin) pour vous connecter à l'instance de base de données principale sur votre cluster de bases de données Aurora PostgreSQL :

   ```
   psql --host=your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

   Lorsque vous y êtes invité, saisissez votre mot de passe. Le client `psql` se connecte à la base de données de connexions administratives par défaut, `postgres=>`, et l'affiche sous forme d'invite.
2. Pour empêcher les utilisateurs de base de données de créer des objets dans le schéma `public`, procédez comme suit :

   ```
   postgres=> REVOKE CREATE ON SCHEMA public FROM PUBLIC;
   ```

```
REVOKE
```

3. Vous créez ensuite une instance de base de données :

```
postgres=> CREATE DATABASE lab_db;
CREATE DATABASE
```

4. Révoquez tous les privilèges du schéma PUBLIC sur cette nouvelle base de données.

```
postgres=> REVOKE ALL ON DATABASE lab_db FROM public;
REVOKE
```

5. Créez un rôle pour les utilisateurs de base de données.

```
postgres=> CREATE ROLE lab_tech;
CREATE ROLE
```

6. Donnez aux utilisateurs de base de données disposant de ce rôle la possibilité de se connecter à la base de données.

```
postgres=> GRANT CONNECT ON DATABASE lab_db TO lab_tech;
GRANT
```

7. Accordez à tous les utilisateurs dotés du rôle `lab_tech` tous les privilèges sur cette base de données.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_db TO lab_tech;
GRANT
```

8. Créez des utilisateurs de base de données, comme suit :

```
postgres=> CREATE ROLE lab_user1 LOGIN PASSWORD 'change_me';
CREATE ROLE
postgres=> CREATE ROLE lab_user2 LOGIN PASSWORD 'change_me';
CREATE ROLE
```

9. Accordez à ces deux utilisateurs les privilèges associés au rôle lab_tech :

```
postgres=> GRANT lab_tech TO lab_user1;
GRANT ROLE
postgres=> GRANT lab_tech TO lab_user2;
```

```
GRANT ROLE
```

À ce stade, `lab_user1` et `lab_user2` peuvent se connecter à la base de données `lab_db`. Cet exemple ne respecte pas les bonnes pratiques pour une utilisation en entreprise, qui peuvent inclure la création de plusieurs instances de base de données, différents schémas et l'octroi d'autorisations limitées. Pour des informations plus complètes et des scénarios supplémentaires, consultez Managing PostgreSQL Users and Roles.

Pour plus d'informations sur les privilèges dans les bases de données PostgreSQL, veuillez consulter la commande GRANT dans la documentation PostgreSQL.

Délégation et contrôle de la gestion des mots de passe utilisateur

En tant qu'administrateur de base de données, vous souhaitez peut-être déléguer la gestion des mots de passe utilisateur. Vous souhaitez peut-être également empêcher les utilisateurs de base de données de modifier leurs mots de passe ou de reconfigurer les contraintes de mot de passe, telles que la durée de vie d'un mot de passe. Pour vous assurer que seuls les utilisateurs de base de données que vous choisissez peuvent modifier les paramètres de mot de passe, vous pouvez activer la fonctionnalité de gestion restreinte des mots de passe. Lorsque vous activez cette fonctionnalité, seuls les utilisateurs de base de données qui ont obtenu le rôle `rds_password` peuvent gérer les mots de passe.

> **Note**
>
> Pour utiliser la gestion restreinte des mots de passe, votre cluster de bases de données Aurora PostgreSQL doit exécuter Amazon Aurora PostgreSQL 10.6 ou versions ultérieures.

Par défaut, cette fonctionnalité est désactivée (`off`), comme illustré ci-dessous :

```
postgres=> SHOW rds.restrict_password_commands;
  rds.restrict_password_commands
--------------------------------
 off
(1 row)
```

Pour l'activer, vous utilisez un groupe de paramètres personnalisé et redéfinissez le paramètre `rds.restrict_password_commands` sur 1. Assurez-vous de redémarrer votre instance de base de données principale Aurora PostgreSQL pour que le réglage prenne effet.

Lorsque cette fonctionnalité est activée, les privilèges `rds_password` sont requis pour les commandes SQL suivantes :

```
CREATE ROLE myrole WITH PASSWORD 'mypassword';
CREATE ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword' VALID UNTIL '2023-01-01';
ALTER ROLE myrole WITH PASSWORD 'mypassword';
ALTER ROLE myrole VALID UNTIL '2023-01-01';
ALTER ROLE myrole RENAME TO myrole2;
```

L'attribution d'un nouveau nom à un rôle (`ALTER ROLE myrole RENAME TO newname`) est également restreint si le mot de passe utilise l'algorithme de hachage MD5.

Lorsque cette fonctionnalité est activée, toute tentative d'exécution de l'une de ces commandes SQL sans les autorisations de rôle `rds_password` génère l'erreur suivante :

```
ERROR: must be a member of rds_password to alter passwords
```

Nous vous recommandons de n'accorder `rds_password` qu'à quelques rôles utilisés exclusivement pour la gestion des mots de passe. Si vous accordez des privilèges `rds_password` aux utilisateurs de base de données qui ne disposent pas de privilèges `rds_superuser`, vous devez également leur accorder l'attribut `CREATEROLE`.

Assurez-vous de vérifier les exigences de mot de passe telles que la date d'expiration et le niveau de complexité requis du côté du client. Si vous utilisez votre propre utilitaire côté client pour les modifications relatives aux mots de passe, l'utilitaire doit être membre de `rds_password` et disposer des privilèges `CREATE ROLE`.

Utilisation de SCRAM pour le chiffrement de mot de passe PostgreSQL

Vous pouvez utiliser le mécanisme d’authentification SCRAM (Salted Challenge Response Authentication Mechanism) au lieu de l’algorithme MD5 par défaut de PostgreSQL pour le chiffrement des mots de passe. Le mécanisme d’authentification SCRAM est considéré comme plus sécurisé que MD5. Pour en savoir plus sur ces deux approches différentes de sécurisation des mots de passe, consultez Password Authentication (Authentification par mot de passe) dans la documentation PostgreSQL.

Nous vous recommandons d’utiliser SCRAM plutôt que MD5 comme schéma de chiffrement de mot de passe pour votre cluster de bases de données Aurora PostgreSQL. SCRAM est pris en

charge dans Aurora PostgreSQL version 10, ainsi que dans toutes les versions majeures et mineures supérieures. Il s'agit d'un mécanisme stimulation/réponse cryptographique qui utilise l'algorithme scram-sha-256 pour l'authentification par mot de passe et le chiffrement de mot de passe.

Vous devrez peut-être mettre à jour les bibliothèques pour vos applications clientes de sorte qu'elles prennent en charge SCRAM. Par exemple, les versions JDBC antérieures à 42.2.0 ne prennent pas en charge SCRAM. Pour plus d'informations, consultez PostgreSQL JDBC Driver (Pilote JDBC PostgreSQL) dans la documentation du pilote JDBC PostgreSQL. Pour obtenir la liste des autres pilotes PostgreSQL prenant en charge SCRAM, consultez la liste des pilotes dans la documentation PostgreSQL.

Aurora PostgreSQL versions 14 et ultérieures prennent en charge scram-sha-256 pour le chiffrement de mot de passe par défaut pour les nouveaux clusters de bases de données. Pour ces versions, la valeur `password_encryption` du groupe de paramètres du cluster de bases de données par défaut (`default.aurora-postgresql14`) est définie sur scram-sha-256. SCRAM n'est pas pris en charge sur Aurora Serverless v1.

Configuration de votre cluster de bases de données Aurora PostgreSQL de sorte à requérir SCRAM

Pour Aurora PostgreSQL 14.3 et les versions ultérieures, vous pouvez exiger que le cluster de bases de données Aurora PostgreSQL n'accepte que les mots de passe qui utilisent l'algorithme scram-sha-256.

> **Important**
>
> Pour les proxys RDS existants avec des bases de données PostgreSQL, si vous modifiez l'authentification de base de données pour utiliser uniquement `SCRAM`, le proxy devient indisponible pendant 60 secondes au maximum. Pour éviter ce problème, effectuez l'une des actions suivantes :
>
> - Veillez à ce que la base de données permette à la fois l'authentification `SCRAM` et `MD5`.
> - Pour utiliser uniquement l'authentification `SCRAM`, créez un nouveau proxy, migrez le trafic de votre application vers ce nouveau proxy, puis supprimez le proxy précédemment associé à la base de données.

Avant d'apporter des modifications à votre système, assurez-vous de bien comprendre le processus complet, comme suit :

- Obtenez des informations sur tous les rôles et sur le chiffrement des mots de passe pour tous les utilisateurs de base de données.
- Revérifiez les paramètres de votre cluster de bases de données Aurora PostgreSQL qui contrôlent le chiffrement des mots de passe.
- Si votre cluster de bases de données Aurora PostgreSQL utilise un groupe de paramètres par défaut, vous devez créer un groupe de paramètres de cluster de bases de données personnalisé et l'appliquer à votre cluster de bases de données Aurora PostgreSQL de sorte à pouvoir modifier les paramètres si nécessaire. Si votre cluster de bases de données Aurora PostgreSQL utilise un groupe de paramètres personnalisé, vous pouvez modifier ultérieurement les paramètres nécessaires dans le processus, selon vos besoins.
- Remplacez le paramètre `password_encryption` par `scram-sha-256`.
- Informez tous les utilisateurs de la base de données qu'ils doivent mettre à jour leurs mots de passe. Faites de même pour votre compte `postgres`. Les nouveaux mots de passe sont chiffrés et stockés à l'aide de l'algorithme scram-sha-256.
- Vérifiez que tous les mots de passe utilisent le même type de chiffrement.
- Si tous les mots de passe utilisent scram-sha-256, vous pouvez modifier le paramètre `rds.accepted_password_auth_method` de `md5+scram` à `scram-sha-256`.

> **Warning**
>
> Après avoir changé `rds.accepted_password_auth_method` pour scram-sha-256 uniquement, tous les utilisateurs (rôles) avec des mots de passe chiffrés par `md5` ne peuvent pas se connecter.

Se préparer à exiger SCRAM pour votre cluster de bases de données Aurora PostgreSQL

Avant d'apporter des modifications à votre cluster de bases de données Aurora PostgreSQL, vérifiez tous les comptes utilisateurs de base de données existants. Vérifiez également le type de chiffrement utilisé pour les mots de passe. Pour ce faire, utilisez l'extension `rds_tools`. Pour savoir quelles versions de PostgreSQL prennent en charge `rds_tools`, consultez Versions d'extension pour Amazon RDS pour PostgreSQL.

Pour obtenir la liste des utilisateurs de base de données (rôles) et des méthodes de chiffrement des mots de passe

1. Utilisez `psql` pour vous connecter à l'instance principale de votre cluster de bases de données Aurora PostgreSQL , comme suit.

```
psql --host=cluster-name-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password
```

2. Installez l'extension `rds_tools`.

```
postgres=> CREATE EXTENSION rds_tools;
CREATE EXTENSION
```

3. Obtenez la liste des rôles et des méthodes de chiffrement.

```
postgres=> SELECT * FROM
      rds_tools.role_password_encryption_type();
```

Vous voyez des résultats similaires à ce qui suit.

```
       rolname        | encryption_type
----------------------+-----------------
 pg_monitor           |
 pg_read_all_settings |
 pg_read_all_stats    |
 pg_stat_scan_tables  |
 pg_signal_backend    |
 lab_tester           | md5
 user_465             | md5
 postgres             | md5
(8 rows)
```

Création d'un groupe de paramètres de cluster de bases de données personnalisé

> **Note**
>
> Si votre cluster de bases de données Aurora PostgreSQL utilise déjà un groupe de paramètres personnalisé, vous n'avez pas besoin d'en créer un.

Pour obtenir un aperçu des groupes de paramètres pour Aurora, consultez Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

Le type de chiffrement utilisé pour les mots de passe est défini dans un paramètre, `password_encryption`. Le chiffrement autorisé par le cluster de bases de données Aurora PostgreSQL est défini dans un autre paramètre, `rds.accepted_password_auth_method`. Le remplacement de l'un de ces paramètres par une valeur autre que celle par défaut requiert de créer un groupe de paramètres de cluster de bases de données personnalisé et de l'appliquer à votre cluster.

Vous pouvez également utiliser AWS Management Console ou l'API RDS pour créer un groupe de paramètres de cluster de bases de données personnalisé. Pour plus d'informations, consultez Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

Vous pouvez maintenant employer le groupe de paramètres personnalisés avec votre instance de base de données.

Pour créer un groupe de paramètres de cluster de bases de données personnalisé

1. Utilisez la commande CLI `create-db-cluster-parameter-group` pour créer le groupe de paramètres personnalisé pour le cluster. L'exemple suivant utilise `aurora-postgresql13` comme source pour ce groupe de paramètres personnalisé.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-db-cluster-parameter-group --db-cluster-parameter-group-name 'docs-lab-scram-passwords' \
     --db-parameter-group-family aurora-postgresql13  --description 'Custom DB cluster parameter group for SCRAM'
   ```

   Pour Windows :

   ```
   aws rds create-db-cluster-parameter-group --db-cluster-parameter-group-name "docs-lab-scram-passwords" ^
     --db-parameter-group-family aurora-postgresql13  --description "Custom DB cluster parameter group for SCRAM"
   ```

   Ensuite, vous pouvez associer le groupe de paramètres personnalisé à votre cluster.
2. Utilisez la commande CLI `modify-db-cluster` pour appliquer ce groupe de paramètres personnalisé à votre cluster de bases de données Aurora PostgreSQL.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster --db-cluster-identifier 'your-instance-name' \
        --db-cluster-parameter-group-name "docs-lab-scram-passwords
```

Pour Windows :

```
aws rds modify-db-cluster --db-cluster-identifier "your-instance-name" ^
        --db-cluster-parameter-group-name "docs-lab-scram-passwords
```

Pour resynchroniser votre cluster de bases de données Aurora PostgreSQL avec votre groupe de paramètres de cluster de bases de données personnalisé, redémarrez l'instance principale et toutes les autres instances du cluster.

Configuration du chiffrement des mots de passe pour utiliser SCRAM

Le mécanisme de chiffrement du mot de passe utilisé par un cluster de bases de données Aurora PostgreSQL est défini(e) dans le groupe de paramètres du cluster de bases de données dans le paramètre `password_encryption`. Les valeurs autorisées incluent une valeur non définie, `md5` ou `scram-sha-256`. La valeur par défaut dépend de la version d'Aurora PostgreSQL , comme suit :

- Aurora PostgreSQL 14 : la valeur par défaut est `scram-sha-256`
- Aurora PostgreSQL 13 : la valeur par défaut est `md5`

En attachant un groupe de paramètres de cluster de bases de données personnalisé à votre cluster de bases de données Aurora PostgreSQL, vous pouvez modifier les valeurs du paramètre de chiffrement des mots de passe.

Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

Pour remplacer le paramètre de chiffrement des mots de passe par scram-sha-256

- Remplacez la valeur du chiffrement des mots de passe par scram-sha-256, comme indiqué ci-après. Cette modification peut être appliquée immédiatement, car le paramètre est dynamique. Aucun redémarrage n'est donc nécessaire pour que la modification soit appliquée.

  Pour Linux, macOS ou Unix :

  ```
  aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name \
    'docs-lab-scram-passwords' --parameters
   'ParameterName=password_encryption,ParameterValue=scram-
  sha-256,ApplyMethod=immediate'
  ```

  Pour Windows :

  ```
  aws rds modify-db-parameter-group --db-parameter-group-name ^
    "docs-lab-scram-passwords" --parameters
   "ParameterName=password_encryption,ParameterValue=scram-
  sha-256,ApplyMethod=immediate"
  ```

Migration des mots de passe des rôles utilisateur vers SCRAM

Vous pouvez migrer les mots de passe pour les rôles d'utilisateur vers SCRAM comme décrit ci-dessous.

Pour migrer les mots de passe des utilisateurs de base de données (rôles) de MD5 vers SCRAM

1. Connectez-vous en tant qu'utilisateur administrateur (nom d'utilisateur par défaut, `postgres`) comme suit.

   ```
   psql --host=cluster-name-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

2. Vérifiez la valeur du paramètre `password_encryption` sur votre instance de base de données RDS pour PostgreSQL à l'aide de la commande suivante.

   ```
   postgres=> SHOW password_encryption;
    password_encryption
   ---------------------
    md5
   ```

```
 (1 row)
```

3. Remplacez la valeur de ce paramètre par scram-sha-256. Pour plus d'informations, consultez Configuration du chiffrement des mots de passe pour utiliser SCRAM.
4. Vérifiez à nouveau la valeur pour vous assurer qu'elle est maintenant réglée sur `scram-sha-256`, comme suit.

```
postgres=> SHOW password_encryption;
 password_encryption
---------------------
 scram-sha-256
 (1 row)
```

5. Demandez à tous les utilisateurs de base de données de modifier leurs mots de passe. Veillez également à modifier votre propre mot de passe pour le compte `postgres` (utilisateur de base de données avec privilèges `rds_superuser`).

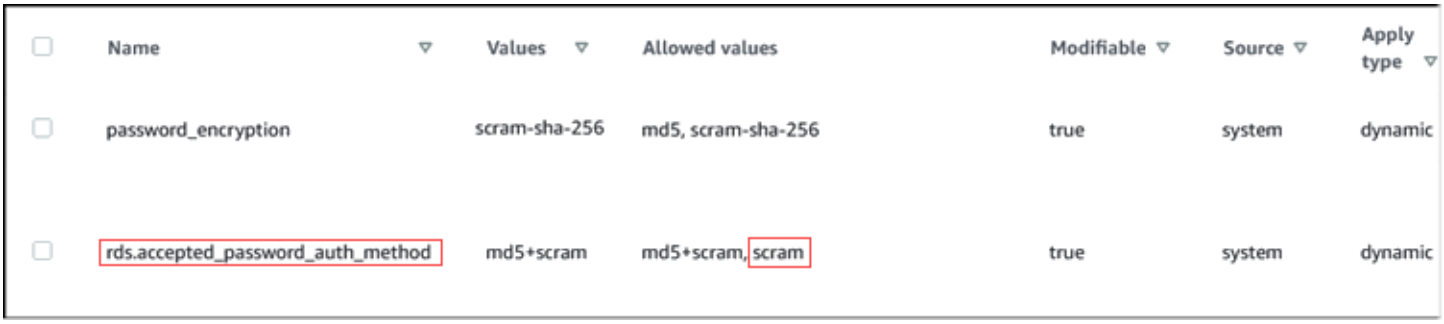
```
labdb=> ALTER ROLE postgres WITH LOGIN PASSWORD 'change_me';
ALTER ROLE
```

6. Répétez l'opération pour toutes les bases de données de votre cluster de bases de données Aurora PostgreSQL.

Modification du paramètre de sorte à utiliser SCRAM

Il s'agit de la dernière étape du processus. Une fois que vous avez effectué la modification de la procédure suivante, tous les comptes utilisateurs (rôles) qui utilisent toujours le chiffrement `md5` pour les mots de passe ne pourront pas se connecter au cluster de bases de données Aurora PostgreSQL.

Le paramètre `rds.accepted_password_auth_method` spécifie la méthode de chiffrement acceptée par le cluster de bases de données Aurora PostgreSQL pour un mot de passe utilisateur pendant le processus de connexion. La valeur par défaut est `md5+scram`, ce qui signifie que l'une des méthodes est acceptée. L'image suivante indique la valeur par défaut de ce paramètre.



Name	Values	Allowed values	Modifiable	Source	Apply type
password_encryption	scram-sha-256	md5, scram-sha-256	true	system	dynamic
rds.accepted_password_auth_method	md5+scram	md5+scram, scram	true	system	dynamic

Les valeurs autorisées pour ce paramètre sont `md5+scram` ou `scram`. Si la valeur de ce paramètre est remplacée par `scram`, le paramètre devient obligatoire.

Pour modifier la valeur du paramètre afin d'exiger l'authentification SCRAM pour les mots de passe

1. Vérifiez que tous les mots de passe utilisateur de toutes les bases de données de votre cluster de bases de données Aurora PostgreSQL utilisent `scram-sha-256` pour le chiffrement des mots de passe. Pour ce faire, interrogez `rds_tools` pour obtenir le rôle (utilisateur) et le type de chiffrement, comme suit.

```
postgres=> SELECT * FROM rds_tools.role_password_encryption_type();
  rolname        | encryption_type
  ---------------------+-----------------
  pg_monitor           |
  pg_read_all_settings |
  pg_read_all_stats    |
  pg_stat_scan_tables  |
  pg_signal_backend    |
  lab_tester           | scram-sha-256
  user_465             | scram-sha-256
  postgres             | scram-sha-256
  ( rows)
```

2. Répétez la requête sur toutes les instances de base de données de votre cluster de bases de données Aurora PostgreSQL.

   Si tous les mots de passe utilisent scram-sha-256, vous pouvez continuer.

3. Remplacez la valeur de l'authentification par mot de passe acceptée par scram-sha-256, comme suit.

   Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name 'docs-
lab-scram-passwords' \
  --parameters
 'ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name "docs-
lab-scram-passwords" ^
  --parameters
 "ParameterName=rds.accepted_password_auth_method,ParameterValue=scram,ApplyMethod=immediat
```

Sécurisation des données Aurora PostgreSQL avec SSL/TLS

Amazon RDS prend en charge le chiffrement SSL (Secure Socket Layer) et TLS (Transport Layer Security) pour les clusters de bases de données Aurora PostgreSQL. Utilisation de SSL/TLS, you can encrypt a connection between your applications and your Aurora PostgreSQL DB clusters. You can also force all connections to your Aurora PostgreSQL DB cluster to use SSL/TLS. Amazon Aurora PostgreSQL prend en charge le protocole TLS (Transport Layer Security) versions 1.1 et 1.2. Nous vous recommandons d'utiliser TLS 1.2 pour les connexions chiffrées. Nous avons ajouté la prise en charge de la version TLSv1 3.3 à partir des versions suivantes d'Aurora PostgreSQL :

- Version 15.3 et toutes les versions ultérieures
- 14.8 et versions 14 ultérieures
- 13.11 et versions 13 ultérieures
- 12.15 et versions 12 ultérieures
- 11.20 et versions 11 ultérieures

Pour des informations générales sur le SSL/TLS support et les bases de données PostgreSQL, consultez la section Support SSL dans la documentation PostgreSQL. Pour plus d'informations sur l'utilisation d'une SSL/TLS connexion via JDBC, consultez la section Configuration du client dans la documentation de PostgreSQL.

Rubriques

- Exigence d'une connexion SSL/TLS à un cluster de bases de données Aurora PostgreSQL

- Détermination du statut de la connexion SSL/TLS
- Configuration de suites de chiffrement pour les connexions aux clusters de bases de données Aurora PostgreSQL

Le support SSL/TLS est disponible dans toutes les régions AWS pour Aurora PostgreSQL. Amazon RDS crée un SSL/TLS certificat pour votre cluster de base de données Aurora PostgreSQL lors de sa création. Si vous activez la vérification du SSL/TLS certificat, le SSL/TLS certificat inclut le point de terminaison du cluster de base de données en tant que nom commun (CN) du SSL/TLS certificat afin de se protéger contre les attaques par usurpation d'identité.

Pour se connecter à un cluster de bases de données Aurora PostgreSQL via SSL/TLS

1. Téléchargez le certificat.

   Pour plus d'informations sur le téléchargement de certificats, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.
2. Importez le certificat dans votre système d'exploitation.
3. Connectez-vous à votre cluster de bases de données Aurora PostgreSQL via SSL/TLS.

   Lors de la connexion avec SSL/TLS, votre client peut choisir de vérifier ou pas la chaîne du certificat. Si vos paramètres de connexion spécifient `sslmode=verify-ca` ou `sslmode=verify-full`, votre client nécessite que les certificats de l'autorité de certification RDS soient dans leur magasin d'approbations ou référencés dans l'URL de connexion. L'exigence nécessite de vérifier la chaîne du certificat qui signe le certificat de votre base de données.

   Lorsqu'un client, tel que psql ou JDBC, est configuré avec le SSL/TLS support, le client essaie d'abord de se connecter à la base de données par SSL/TLS défaut. Si le client ne parvient pas à se connecter àSSL/TLS, it reverts to connecting without SSL/TLS. Par défaut, l'option `sslmode` est définie sur `prefer` pour les clients JDBC et libpq.

   Utilisez le paramètre `sslrootcert` pour référencer le certificat, par exemple, `sslrootcert=rds-ssl-ca-cert.pem`.

Voici un exemple d'utilisation de psql pour se connecter à un cluster de bases de données Aurora PostgreSQL :

```
$ psql -h testpg.cdhmuqifdpib.us-east-1.rds.amazonaws.com -p 5432 \
    "dbname=testpg user=testuser sslrootcert=rds-ca-2015-root.pem sslmode=verify-full"
```

Exigence d'une connexion SSL/TLS à un cluster de bases de données Aurora PostgreSQL

Pour exiger des SSL/TLS connexions à votre cluster de base de données Aurora PostgreSQL, utilisez le paramètre. `rds.force_ssl`

- Pour exiger SSL/TLS des connexions, définissez la valeur du `rds.force_ssl` paramètre sur 1 (activé).
- Pour désactiver SSL/TLS les connexions requises, définissez la valeur du `rds.force_ssl` paramètre sur 0 (désactivé).

La valeur par défaut de ce paramètre dépend de la version d'Aurora PostgreSQL :

- Pour Aurora PostgreSQL 17 et les versions ultérieures : la valeur par défaut est 1 (activé).
- Pour Aurora PostgreSQL 16 et les versions antérieures : la valeur par défaut est 0 (désactivé).

> **Note**
>
> Lorsque vous effectuez une mise à niveau de version majeure d'Aurora PostgreSQL 16 ou version antérieure vers la version 17 ou une version ultérieure, la valeur par défaut du paramètre passe de 0 (désactivé) à 1 (activé). Ce changement peut entraîner des défaillances de connectivité pour les applications qui ne sont pas configurées pour le protocole SSL. Pour revenir au comportement par défaut précédent, définissez ce paramètre sur 0 (désactivé).

Pour plus d'informations sur la gestion des paramètres, consultez Groupes de paramètres pour Amazon Aurora.

La mise à jour du `rds.force_ssl` paramètre définit également le paramètre `ssl` PostgreSQL sur 1 (activé) et modifie le fichier de votre cluster de bases de données pour prendre en charge `pg_hba.conf` la nouvelle configuration. SSL/TLS

Lorsque le `rds.force_ssl` paramètre est défini sur 1 pour un cluster de base de données, vous obtenez une sortie similaire à la suivante lorsque vous vous connectez, ce qui indique que SSL/TLS c'est désormais obligatoire :

```
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql (9.3.12, server 9.4.4)
WARNING: psql major version 9.3, server major version 9.4.
Some psql features might not work.
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

Détermination du statut de la connexion SSL/TLS

Le statut chiffré de votre connexion est affiché dans la page d'accueil d'ouverture de session lorsque vous vous connectez au cluster de bases de données.

```
Password for user master:
psql (9.3.12)
SSL connection (cipher: DHE-RSA-AES256-SHA, bits: 256)
Type "help" for help.

postgres=>
```

Vous pouvez également charger l'`sslinfo`extension, puis appeler la `ssl_is_used()` fonction pour déterminer si elle SSL/TLS est utilisée. La fonction renvoie `t` si la connexion utilise SSL/TLS ; sinon, elle renvoie `f`.

```
postgres=> create extension sslinfo;
CREATE EXTENSION

postgres=> select ssl_is_used();
 ssl_is_used
---------
t
(1 row)
```

Vous pouvez utiliser la commande `select ssl_cipher()` pour déterminer le chiffrement SSL/TLS :

```
postgres=> select ssl_cipher();
ssl_cipher
--------------------
DHE-RSA-AES256-SHA
(1 row)
```

Si vous activez `set rds.force_ssl` et redémarrez le cluster de bases de données, les connexions non-SSL sont refusées avec le message suivant :

```
$ export PGSSLMODE=disable
$ psql postgres -h SOMEHOST.amazonaws.com -p 8192 -U someuser
psql: FATAL: no pg_hba.conf entry for host "host.ip", user "someuser", database
 "postgres", SSL off
$
```

Pour plus d'informations sur l'option `sslmode`, consultez Database Connection Control Functions dans la documentation PostgreSQL.

Configuration de suites de chiffrement pour les connexions aux clusters de bases de données Aurora PostgreSQL

L'utilisation de suites de chiffrement configurables vous permet d'avoir plus de contrôle sur la sécurité des connexions de vos bases de données. Vous pouvez spécifier une liste de suites de chiffrement que vous souhaitez autoriser pour sécuriser les SSL/TLS connexions des clients à votre base de données. Avec les suites de chiffrement configurables, vous pouvez contrôler le chiffrement de connexion accepté par votre serveur de base de données. Cela permet d'éviter l'utilisation de chiffrements non sécurisés ou obsolètes.

Les suites de chiffrement configurables sont prises en charge dans Aurora PostgreSQL 11.8 et versions ultérieures.

Pour spécifier la liste des chiffrements autorisés pour le chiffrement des connexions, modifiez le paramètre de cluster `ssl_ciphers`. Définissez le `ssl_ciphers` paramètre sur une chaîne de valeurs chiffrées séparées par des virgules dans un groupe de paramètres de cluster à l'aide de l'API

AWS Management Console, de ou de l' AWS CLI API RDS. Pour définir des paramètres de cluster, consultez Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

La table suivante présente les chiffrements pris en charge pour les versions valides du moteur Aurora PostgreSQL.

Versions du moteur Aurora PostgreSQL	Chiffrements pris en charge	TLS 1.1	TLS 1.2	TLS 1.3
9.6, 10.20 et antérieures, 11.15 et antérieures, 12.10 et antérieures, 13.6 et antérieures	DHE-RSA- -SHA AES128	Oui	Non	Non
	DHE-RSA- - AES128 SHA256	Non	Oui	Non
	DHE-RSA- -GCM- AES128 SHA256	Non	Oui	Non
	DHE-RSA- -SHA AES256	Non	Oui	Non
	DHE-RSA- - AES256 SHA256	Non	Oui	Non
	DHE-RSA- -GCM- AES256 SHA384	Non	Oui	Non
	ECDHE-ECDSA- - SHA AES256	Non	Oui	Non
	ECDHE-ECDSA- -GCM- AES256 SHA384	Non	Oui	Non
	ECDHE-RSA- - AES256 SHA384	Non	Oui	Non
	ECDHE-RSA- -SHA AES128	Non	Oui	Non

Versions du moteur Aurora PostgreSQL	Chiffrements pris en charge	TLS 1.1	TLS 1.2	TLS 1.3
	ECDHE-RSA- - AES128 SHA256 ECDHE-RSA- -GCM- AES128 SHA256 ECDHE-RSA- -SHA AES256 ECDHE-RSA- -GCM- AES256 SHA384			

Versions du moteur Aurora PostgreSQL	Chiffrements pris en charge	TLS 1.1	TLS 1.2	TLS 1.3
10.21, 11.16, 12.11, 13.7, 14.3 et 14.4	ECDHE-RSA- -SHATLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Oui	Oui	Non
	AES128	Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Oui	Oui	Non
		Non	Oui	Non
		Oui	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Non	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Oui	Oui	Non
		Non	Oui	Non
		Oui	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256			
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA			
	TLS_ECDHE_RSA_WITH			

Versions du moteur Aurora PostgreSQL	Chiffrements pris en charge	TLS 1.1	TLS 1.2	TLS 1.3
	_AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_W ITH_AES_128_GCM_ SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_AVEC_ 0_ 05_ CHACHA2 POLY13 SHA256			

Versions du moteur Aurora PostgreSQL	Chiffrements pris en charge	TLS 1.1	TLS 1.2	TLS 1.3
10.22, 11.17, 12.12, 13.8, 14.5 et 15.2	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Oui	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	Non	Oui	Non
		Oui	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Oui	Oui	Non
		Non	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Oui	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Non	Oui	Non
		Oui	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Oui	Oui	Non
		Oui	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256			

Versions du moteur Aurora PostgreSQL	Chiffrements pris en charge	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_ECDHE _RSA_WITH _AES_256_CBC_SHA TLS_ECDHE _RSA_WITH _AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_W ITH_AES_128_GCM_ SHA256 TLS_RSA_W ITH_AES_128_CBC_ SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_AVEC_ 0_ 05_ CHACHA2 POLY13 SHA256			

Versions du moteur Aurora PostgreSQL	Chiffrements pris en charge	TLS 1.1	TLS 1.2	TLS 1.3
11.20, 12.15, 13.11, 14.8, 15.3, 16.1 et versions ultérieures	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Oui	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	Non	Oui	Non
		Oui	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256	Oui	Oui	Non
		Non	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA	Oui	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384	Non	Oui	Non
		Oui	Oui	Non
		Non	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA	Oui	Oui	Non
		Oui	Oui	Non
	TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256	Non	Oui	Non
		Non	Non	Oui
		Non	Non	Oui
	TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256			

Versions du moteur Aurora PostgreSQL	Chiffrements pris en charge	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_ECDHE _RSA_WITH _AES_256_CBC_SHA TLS_ECDHE _RSA_WITH _AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_256_GCM_ SHA384 TLS_RSA_W ITH_AES_2 56_CBC_SHA TLS_RSA_W ITH_AES_128_GCM_ SHA256 TLS_RSA_W ITH_AES_128_CBC_ SHA256 TLS_RSA_W ITH_AES_1 28_CBC_SHA TLS_ECDHE _RSA_AVEC_ 0_ 05_ CHACHA2 POLY13 SHA256 TLS_AES_128_GCM_ SHA256			

Versions du moteur Aurora PostgreSQL	Chiffrements pris en charge	TLS 1.1	TLS 1.2	TLS 1.3
	TLS_AES_256_GCM_SHA384			

Vous pouvez également utiliser la commande CLI describe-engine-default-cluster-parameters pour déterminer quelles suites de chiffrement sont actuellement prises en charge pour une famille de groupes de paramètres spécifique. L'exemple suivant montre comment obtenir les valeurs autorisées pour le paramètre de cluster `ssl_cipher` pour Aurora PostgreSQL 11.

```
aws rds describe-engine-default-cluster-parameters --db-parameter-group-family aurora-
postgresql11

    ...some output truncated...
 {
  "ParameterName": "ssl_ciphers",
  "Description": "Sets the list of allowed TLS ciphers to be used on secure
 connections.",
  "Source": "engine-default",
  "ApplyType": "dynamic",
  "DataType": "list",
  "AllowedValues": "DHE-RSA-AES128-SHA,DHE-RSA-AES128-SHA256,DHE-RSA-AES128-GCM-
SHA256,DHE-RSA-AES256-SHA,DHE-RSA-AES256-SHA256,DHE-RSA-AES256-GCM-SHA384,
  ECDHE-RSA-AES128-SHA,ECDHE-RSA-AES128-SHA256,ECDHE-RSA-AES128-GCM-
SHA256,ECDHE-RSA-AES256-SHA,ECDHE-RSA-AES256-SHA384,ECDHE-RSA-AES256-GCM-
SHA384,TLS_RSA_WITH_AES_256_GCM_SHA384,

 TLS_RSA_WITH_AES_256_CBC_SHA,TLS_RSA_WITH_AES_128_GCM_SHA256,TLS_RSA_WITH_AES_128_CBC_SHA256,T

 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_AE
  "IsModifiable": true,
  "MinimumEngineVersion": "11.8",
  "SupportedEngineModes": [
   "provisioned"
  ]
 },
    ...some output truncated...
```

Le paramètre `ssl_ciphers` prend par défaut toutes les suites de chiffrement autorisées. Pour plus d'informations sur les chiffrements, consultez la variable ssl_ciphers dans la documentation PostgreSQL.

Utilisation du masquage dynamique avec Aurora PostgreSQL

Le masquage dynamique des données est une fonctionnalité de sécurité qui protège les données sensibles des bases de données Aurora PostgreSQL en contrôlant la façon dont les données apparaissent aux utilisateurs au moment de la requête. Aurora l'implémente via l'`pg_columnmask`extension. `pg_columnmask`fournit une protection des données au niveau des colonnes qui complète les mécanismes natifs de sécurité au niveau des lignes et de contrôle d'accès granulaire de PostgreSQL.

Avec`pg_columnmask`, vous créez des politiques de masquage qui déterminent la visibilité des données en fonction des rôles des utilisateurs. Lorsque les utilisateurs interrogent des tables avec des politiques de masquage, Aurora PostgreSQL applique la fonction de masquage appropriée au moment de la requête en fonction du rôle de l'utilisateur et du poids des politiques. Les données sous-jacentes restent inchangées pendant le stockage.

`pg_columnmask`prend en charge les fonctionnalités suivantes :

- Fonctions de masquage intégrées et personnalisées : utilisez des fonctions prédéfinies pour les modèles courants tels que le masquage des e-mails et des textes, ou créez vos propres fonctions personnalisées pour protéger les données sensibles (PII) grâce à des politiques de masquage basées sur SQL.
- Stratégies de masquage multiples : masquez complètement les informations, remplacez les valeurs partielles par des caractères génériques ou définissez des approches de masquage personnalisées.
- Priorisation des politiques : définissez plusieurs politiques pour une seule colonne. Utilisez des pondérations pour déterminer quelle politique de masquage doit être utilisée lorsque plusieurs politiques s'appliquent à une colonne. Aurora PostgreSQL applique des politiques basées sur le poids et l'appartenance aux rôles des utilisateurs.

`pg_columnmask`est disponible sur Aurora PostgreSQL version 16.10 et versions ultérieures, et sur les versions 17.6 et supérieures. Il est disponible sans frais supplémentaires.

Commencer à utiliser le masquage dynamique

Pour masquer les données de manière dynamique, vous devez installer l'`pg_columnmask`extension dans votre base de données et créer des politiques de masquage pour vos tables. Le processus de configuration implique la vérification des conditions préalables, l'installation de l'extension, la configuration des rôles, la création de politiques et les tests de validation.

Installation et configuration des extensions

Connectez-vous à votre cluster Aurora PostgreSQL à l'aide de l'éditeur de requêtes de la console RDS ou d'un client PostgreSQL tel que psql avec les informations d'identification rds_superuser (utilisateur principal).

Exécutez la commande de création d'extension pour activer `pg_columnmask` les fonctionnalités :

```
CREATE EXTENSION pg_columnmask;
```

Cette commande installe l'`pg_columnmask`extension, crée les tables de catalogue nécessaires et enregistre les fonctions de masquage intégrées. L'installation de l'extension est spécifique à la base de données, ce qui signifie que vous devez l'installer séparément dans chaque base de données où la fonctionnalité est requise.

> **Note**
>
> Les connexions effectuées avant l'installation de cette extension afficheront toujours des données non masquées. Fermez et reconnectez-vous pour résoudre ce problème.

Vérifiez l'installation de l'extension en vérifiant les fonctions de masquage disponibles :

```
SELECT proname FROM pg_proc
    WHERE pronamespace = 'pgcolumnmask'::regnamespace AND proname LIKE 'mask_%';
    proname
--------Output --------
 mask_email
 mask_text
 mask_timestamp
(3 rows)
```

Procédures de gestion des politiques de masquage des données

Vous pouvez gérer les politiques de masquage à l'aide des procédures fournies par l'`pg_columnmask`extension. Pour créer, modifier ou supprimer des politiques de masquage, vous devez disposer de l'un des privilèges suivants :

- Propriétaire de la table sur laquelle vous créez la `pg_columnmask` politique.
- Membre de`rds_superuser`.
- Membre du rôle de responsable des `pg_columnmask` politiques défini par le `pgcolumnmask.policy_admin_rolname` paramètre.

La commande suivante crée une table qui sera utilisée dans les sections suivantes :

```
CREATE TABLE public.customers (
    id SERIAL PRIMARY KEY,
    name TEXT,
    phone TEXT,
    address TEXT,
    email TEXT
);
```

CRÉER UNE POLITIQUE DE MASQUAGE

La procédure suivante crée une nouvelle politique de masquage pour une table utilisateur :

Syntaxe

```
create_masking_policy(
    policy_name,
    table_name,
    masking_expressions,
    roles,
    weight)
```

Arguments

Paramètre	Datatype	Description
`policy_name`	NAME	Nom de la politique de masquage. Doit être unique par table.
`table_name`	REGCLASS	Le qualified/unqualified nom ou l'identifiant de la table à laquelle appliquer la politique de masquage.
`masking_expressions`	JSONB	Objet JSON contenant le nom de colonne et les paires de fonctions de masquage. Chaque clé est un nom de colonne et sa valeur est l'expression de masquage à appliquer à cette colonne.
`roles`	NOM []	Les rôles auxquels s'applique cette politique de masquage. La valeur par défaut est PUBLIC.
`weight`	INT	Le poids de la politique de masquage. Lorsque plusieurs politiques sont applicables à la requête d'un utilisateur donné, la politique ayant le poids le plus élevé (nombre entier le plus élevé) sera appliquée à chaque colonne masquée. La valeur par défaut est 0. Il n'y a pas deux politiques de masquage proposées sur

Paramètre	Datatype	Description
		la table qui peuvent avoir le même poids.

Type de retour

Aucune

Example de la création d'une politique de masquage qui masque la colonne d'e-mail associée au **`test_user`** rôle :

```
CALL pgcolumnmask.create_masking_policy(
    'customer_mask',
    'public.customers',
    JSON_OBJECT('{
        "email", "pgcolumnmask.mask_email(email)"
    }')::JSONB,
    ARRAY['test_user'],
    100
);
```

ALTER_MASKING_POLICY

Cette procédure modifie une politique de masquage existante. `ALTER_MASKING_POLICY`peut modifier les expressions de masquage de la politique, l'ensemble des rôles auxquels la politique s'applique et le poids de la politique de masquage. Lorsque l'un de ces paramètres est omis, la partie correspondante de la politique reste inchangée.

Syntaxe

```
alter_masking_policy(
    policy_name,
    table_name,
    masking_expressions,
    roles,
    weight)
```

Arguments

Paramètre	Datatype	Description
`policy_name`	NAME	Nom existant de la politique de masquage.
`table_name`	REGCLASS	qualified/unqualified Nom oid de la table contenant la politique de masquage.
`masking_expressions`	JSONB	Nouvel objet JSON contenant le nom de colonne et les paires de fonctions de masquage ou NULL dans le cas contraire.
`roles`	NOM []	La liste des nouveaux rôles auxquels s'applique cette politique de masquage ou NULL dans le cas contraire.
`weight`	INT	Nouveau poids pour la politique de masquage ou NULL dans le cas contraire.

Type de retour

Aucune

Example d'ajouter le rôle d'analyste à une politique de masquage existante sans modifier les autres attributs de la stratégie.

```
CALL pgcolumnmask.alter_masking_policy(
    'customer_mask',
    'public.customers',
    NULL,
    ARRAY['test_user', 'analyst'],
    NULL
);
```

```
-- Alter the weight of the policy without altering other details
CALL pgcolumnmask.alter_masking_policy(
    'customer_mask',
    'customers',
    NULL,
    NULL,
    4
);
```

DROP_MASKING_POLICY

Cette procédure supprime une politique de masquage existante.

Syntaxe

```
drop_masking_policy(
        policy_name,
        table_name)
```

Arguments

Paramètre	Datatype	Description
`policy_name`	NAME	Nom existant de la politique de masquage.
`table_name`	REGCLASS	qualified/unqualified Nom oid de la table contenant la politique de masquage.

Type de retour

Aucune

Example de supprimer la politique de masquage customer_mask

```
-- Drop a masking policy
    CALL pgcolumnmask.drop_masking_policy(
        'customer_mask',
        'public.customers',
    );
```

Identifiants échappés dans la procédure DDL de politique de masquage

Lors de la création de politiques de masquage de données avec des identifiants entre guillemets, un échappement approprié est nécessaire pour garantir l'exactitude des références aux objets et l'application des politiques. Pour utiliser des identifiants entre guillemets dans les procédures de gestion des politiques de `pg_columnmask` masquage :

- Nom de la politique — Doit être placé entre guillemets.
- Nom de la table : le nom du schéma et le nom de la table doivent être placés individuellement entre guillemets lorsque cela est nécessaire.
- Expressions de masquage : les noms des colonnes et des fonctions dans les expressions de masquage doivent être placés entre guillemets et les guillemets eux-mêmes doivent être évités à l'aide d'une barre oblique inverse.
- Rôles — Le tableau des noms de rôles est automatiquement cité entre guillemets. Le nom du rôle doit correspondre exactement au nom indiqué en tenant compte de la distinction `pg_roles` majuscules/minuscules.

Example de syntaxe d'échappement et de citation

Cet exemple montre la syntaxe d'échappement et de citation appropriée lors de la création de politiques de masquage pour les tables, les colonnes, les fonctions et les rôles qui utilisent des noms mixtes ou nécessitent des identifiants entre guillemets dans Aurora PostgreSQL.

```
-- Create a table and columns with mixed case name
CREATE TABLE public."Employees" (
    "Name" TEXT,
    "Email" TEXT,
    ssn VARCHAR(20)
);

-- Create a role with mixed case name
CREATE ROLE "Masked_user";

-- Create a function with mixed case name
CREATE OR REPLACE FUNCTION public."MaskEmail"(text)
    RETURNS character varying
    LANGUAGE plpgsql
    IMMUTABLE PARALLEL SAFE
    AS $$ BEGIN
        RETURN 'XXXXXXXX'::text;
```

```
    END $$;

-- Now use these objects with mixed case names in
-- masking policy management procedures
CALL pgcolumnmask.create_masking_policy(
    '"Policy1"',  -- policy name should be surrounded with double quotes for quoting
    'public."Employees"', -- table and schema name should be individually
                          -- surrounded with double quotes for quoting
    JSON_OBJECT('{
        "\"Email\"", "\"MaskEmail\"(\"Email\")"
    }')::JSONB, -- masking expression should have double quotes around function names
                -- and columns names etc when needed. Also the double quotes itself
                -- should be escaped using \ (backslash) since this is a JSON string
    ARRAY['Masked_user'], -- Rolename do not need quoting
                          -- (this behaviour may change in future release)
    100
);

SELECT * FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'Employees';
-[ RECORD 1 ]-----+-----------------------------------
schemaname        | public
tablename         | Employees
policyname        | Policy1
roles             | {Masked_user}
masked_columns    | {Email}
masking_functions | {"(\"MaskEmail\"(\"Email\"))::text"}
weight            | 100
```

Vues administratives

Vous pouvez consulter l'ensemble de la `pg_columnmask` politique à l'aide de la vue `pgcolumnmask.pg_columnmask_policies` administrative accessible au public. Les informations suivantes sont disponibles via cette vue. La vue renvoie uniquement les politiques de masquage détenues par l'utilisateur actuel.

Nom de colonne	Type de données	Description
schemaname	NAME	Schéma de la relation à laquelle la politique est attachée

Nom de colonne	Type de données	Description
tablename	NAME	Nom de la relation à laquelle la politique est attachée
nom de la politique	NAME	Nom de la politique de masquage, toutes les politiques de masquage ont des noms uniques
roles	TEXTE []	Rôle auquel la politique s'applique.
colonnes_masquées	TEXTE []	Colonnes masquées
fonctions_masquage	TEXTE []	Fonctions de masquage
weight	INT	Poids de la politique ci-jointe

Fonctions de masquage de données prédéfinies

`pg_columnmask`L'extension fournit des fonctions utilitaires intégrées écrites en langage C (pour une exécution plus rapide) qui peuvent être utilisées comme expression de masquage pour les `pg_columnmask` politiques.

texte_masque

Fonction permettant de masquer les données textuelles avec des options de visibilité configurables.

Arguments

Paramètre	Datatype	Description
`input`	TEXT	La chaîne de texte d'origine à masquer
`mask_char`	CHAISE (1)	Caractère utilisé pour le masquage (par défaut : « X »)

Paramètre	Datatype	Description
`visible_prefix`	INT	Nombre de caractères au début du texte saisi qui resteront démasqués (par défaut : 0)
`visible_suffix`	INT	Nombre de caractères à la fin du texte saisi qui resteront démasqués (par défaut : 0)
`use_hash_mask`	BOOLEAN	Si VRAI, utilise un masquage basé sur le hachage au lieu de mask_char (par défaut : FALSE)

Example d'utiliser différentes options de masquage

Masquer l'intégralité de la chaîne d'entrée avec le caractère « X » par défaut

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World');
  mask_text
-------------
 XXXXXXXXXXX
```

Utilisez l'`mask_char`argument pour masquer la saisie de texte à l'aide d'un caractère différent

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*');
  mask_text
-------------
 ***********
```

Utilisation `visible_prefix` et `visible_suffix` paramètres pour contrôler le nombre de caractères non masqués au début et à la fin du texte

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*', 5, 1);
  mask_text
-------------
 Hello*****d
```

Lorsque `use_hash_mask` c'est vrai, la chaîne d'entrée est masquée à l'aide de caractères aléatoires, l'`mask_char`argument est ignoré mais `visible_prefix` est `visible_suffix` toujours respecté.

```
postgres=> SELECT pgcolumnmask.mask_text('Hello World', '*', 2, 2, true);
  mask_text
-------------
 Hex36dOHild
```

mask_timestamp

Paramètre	Datatype	Description
`ts_to_mask`	TIMESTAMP	L'horodatage d'origine à masquer
`mask_part`	TEXT	Spécifie la partie de l'horodatage à masquer (par défaut : « toutes ») Valeurs valides : « année », « mois », « jour », « heure », « minute », « seconde », « tout »
`mask_value`	TIMESTAMP	La valeur d'horodatage à utiliser pour le masquage (par défaut : '1900-01-01 00:00:00')

Example d'utilisation de **`mask_timestamps`**

Ces exemples illustrent le masquage complet de l'horodatage à une valeur par défaut, le masquage partiel de composants d'horodatage spécifiques (année uniquement) et le masquage avec une valeur de remplacement personnalisée.

Masquer complètement la valeur d'entrée à l'horodatage par défaut

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00');
   mask_timestamp
---------------------
 1900-01-01 00:00:00
```

Pour masquer une seule partie de l'horodatage (par exemple, uniquement l'année)

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00', 'year');
   mask_timestamp
---------------------
 1900-06-15 14:30:00
```

Pour modifier la valeur masquée de l'horodatage, utilisez l'argument `mask_value`

```
postgres=> SELECT pgcolumnmask.mask_timestamp('2023-06-15 14:30:00', 'all', '2012-12-12
 12:12:12');
   mask_timestamp
---------------------
 2012-12-12 12:12:12
```

mask_timestamp

Fonction permettant de masquer les adresses e-mail tout en préservant la structure des e-mails.

Paramètre	Datatype	Description
`input`	TEXT	L'adresse e-mail d'origine à masquer
`mask_char`	CHAISE (1)	Caractère utilisé pour le masquage (par défaut : « X »)
`mask_local`	BOOLEAN	Si VRAI, masque la partie locale de l'e-mail (avant @) (par défaut : TRUE)
`mask_domain`	BOOLEAN	Si VRAI, masque la partie domaine de l'e-mail (après @) (par défaut : TRUE)

Example d'utilisation de **`mask_email`**

Ces exemples illustrent le masquage complet des e-mails, les caractères de masque personnalisés et le masquage sélectif de la partie locale ou de la partie domaine de l'adresse e-mail.

Masquage complet

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com');
    mask_email
------------------
 XXXX@XXXXXXX.com
```

`mask_char`À utiliser pour modifier le caractère utilisé pour le masquage

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*');
    mask_email
------------------
 ****@*******.com
```

Utiliser `mask_local` et `mask_domain` contrôler le masquage en local et sur le domaine

```
postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*', true, false);
    mask_email
------------------
 ****@example.com

postgres=> SELECT pgcolumnmask.mask_email('user@example.com', '*', false, true);
    mask_email
------------------
 user@*******.com
```

Implémentation de pg_columnmask dans un flux de travail end-to-end

Cette section décrit une implémentation complète de `pg_columnmask` l'utilisation d'un exemple de table d'employés contenant des données sensibles. Vous apprendrez à créer des fonctions de masquage personnalisées, à définir plusieurs politiques de masquage avec différents niveaux de pondération pour différents rôles (stagiaire, support, analyste) et à observer comment les utilisateurs ayant une appartenance à un ou plusieurs rôles voient différents niveaux de données masquées. Les exemples couvrent également le comportement de masquage dans les instructions DML avec des clauses RETURNING, les déclencheurs sur les tables par rapport aux vues, et les opérations de gestion des politiques, notamment le changement de nom, la modification des poids et le nettoyage.

1. Créez un exemple de table contenant des données sensibles :

   ```
   CREATE SCHEMA hr;
   ```

```
CREATE TABLE hr.employees (
    id INT PRIMARY KEY,
    name TEXT NOT NULL,
    email TEXT,
    ssn TEXT,
    salary NUMERIC(10,2)
 );

INSERT INTO hr.employees VALUES
    (1, 'John Doe', 'john.doe@example.com', '123-45-6789', 50000.00),
    (2, 'Jane Smith', 'jane.smith@example.com', '987-65-4321', 60000.00);
```

2. Créez des fonctions de masquage personnalisées :

```
CREATE OR REPLACE FUNCTION public.mask_ssn(ssn TEXT)
    RETURNS TEXT AS $$
    BEGIN
        RETURN 'XXX-XX-' || RIGHT(ssn, 4);
    END;
    $$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION public.mask_salary(salary NUMERIC, multiplier NUMERIC
 DEFAULT 0.0)
    RETURNS NUMERIC AS $$
    BEGIN
        RETURN salary * multiplier;
    END;
    $$ LANGUAGE plpgsql;
```

3. Créez plusieurs politiques avec différents niveaux de masquage en fonction des rôles des utilisateurs :

```
-- Create different roles
CREATE ROLE analyst_role;
CREATE ROLE support_role;
CREATE ROLE intern_role;

GRANT USAGE ON SCHEMA hr TO analyst_role, support_role, intern_role;
GRANT SELECT ON hr.employees TO analyst_role, support_role, intern_role;
---------------------------------------------------------------------

-- Low-Weight Policy (Intern)
```

```
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_strict',
    'hr.employees',
    JSON_BUILD_OBJECT('name', 'pgcolumnmask.mask_text(name, ''*'')',
                      'email', 'pgcolumnmask.mask_email(email)',
                      'ssn', 'pgcolumnmask.mask_text(ssn, ''*'')',
                      'salary', 'public.mask_salary(salary)')::JSONB,
    ARRAY['intern_role'],
    10  -- Lowest weight
);

-----------------------------------------------------------------------
-- Medium-Weight Policy (Support)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_moderate',
    'hr.employees',
    JSON_BUILD_OBJECT('email', 'pgcolumnmask.mask_email(email, ''#'')',
                      'ssn', 'public.mask_ssn(ssn)',
                      'salary', 'public.mask_salary(salary)')::JSONB,
    ARRAY['support_role'],
    50   -- Medium weight
);

-----------------------------------------------------------------------
-- High-Weight Policy (Analyst)
CALL pgcolumnmask.create_masking_policy(
    'employee_mask_light',
    'hr.employees',
    JSON_BUILD_OBJECT('ssn', 'public.mask_ssn(ssn)',
                      'salary', 'public.mask_salary(salary, 0.9)')::JSONB,
    ARRAY['analyst_role'],
    100   -- Highest weight
);
```

4. Les exemples suivants montrent comment les différents utilisateurs voient les données en fonction de leur appartenance aux rôles et de leur pondération des politiques.

```
-- Create users
CREATE USER sarah_intern;
GRANT intern_role TO sarah_intern;

CREATE USER lisa_support;
GRANT support_role TO lisa_support;
```

```
CREATE USER mike_analyst;
GRANT analyst_role TO mike_analyst;

CREATE USER ethan_support_intern;
GRANT support_role, intern_role TO ethan_support_intern;

CREATE USER john_analyst_intern;
GRANT analyst_role, intern_role TO john_analyst_intern;
```

En tant que stagiaire (masquage le plus strict) :

```
SET ROLE sarah_intern;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | ********   | XXXXXXXX@XXXXXXX.com   | ********** |   0.00
  2 | ********** | XXXXXXXXXX@XXXXXXX.com | ********** |   0.00
```

En tant qu'utilisateur du support (masquage modéré) :

```
SET ROLE lisa_support;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | John Doe   | ########@#######.com   | XXX-XX-6789 |   0.00
  2 | Jane Smith | ##########@#######.com | XXX-XX-4321 |   0.00
```

En tant qu'analyste (masquage le plus léger) :

```
SET ROLE mike_analyst;

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
  1 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
  2 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
```

En tant qu'utilisateur ethan_support_intern qui est à la fois stagiaire et utilisateur de support :

```
SET ROLE ethan_support_intern;

-- masking policies appliable to this user: employee_mask_strict and
 employee_mask_moderate
-- id : unmasked because no masking policy appliable on ethan_support_intern
--            masks these columns
-- name : masked because of employee_mask_strict policy
-- email, ssn, salary : both employee_mask_strict and employee_mask_moderate mask
 these columns
--                      but employee_mask_moderate will be use because of higher
 weight

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     | salary
----+------------+------------------------+-------------+--------
  1 | ********   | ########@#######.com   | XXX-XX-6789 |   0.00
  2 | ********** | ##########@#######.com | XXX-XX-4321 |   0.00
```

En tant que john_analyst_intern qui est à la fois stagiaire et analyste :

```
SET ROLE john_analyst_intern;

-- masking policies appliable to this user: employee_mask_strict and
 employee_mask_light
-- id : unmasked because no masking policy appliable on john_analyst_intern
--            masks these columns
-- name, email : masked because of employee_mask_strict
-- ssn, salary : both employee_mask_strict and employee_mask_light mask these
 columns
--               but employee_mask_light will be use because of higher weight

SELECT * FROM hr.employees;
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
  1 | ********   | XXXXXXXX@XXXXXXX.com   | XXX-XX-6789 | 45000.00
  2 | ********** | XXXXXXXXXX@XXXXXXX.com | XXX-XX-4321 | 54000.00
```

Comprendre le comportement de masquage dans les opérations DML

`pg_columnmasks`'applique de manière cohérente à toutes les opérations DML, y compris les instructions INSERT, UPDATE, DELETE et MERGE. Lorsque vous exécutez ces opérations, Aurora PostgreSQL masque les données selon un principe fondamental : toutes les données lues depuis le stockage sont masquées conformément aux politiques applicables de l'utilisateur actuel.

Le masquage affecte certains des composants de requête suivants, tels que :

- clauses WHERE
- Conditions d'adhésion
- Sous-requêtes
- Clauses de retour

Tous ces composants fonctionnent sur des valeurs masquées, et non sur les données d'origine. Lorsque les données sont enregistrées dans le stockage sans être masquées, les utilisateurs ne voient leur vue masquée que lorsqu'ils les relisent.

Aurora PostgreSQL applique toutes les contraintes de base de données (NOT NULL, UNIQUE, CHECK, FOREIGN KEY) aux valeurs stockées réelles, et non aux valeurs masquées. Cela peut parfois créer des incohérences apparentes si les fonctions de masquage ne sont pas conçues avec soin.

Le masquage fonctionne parallèlement aux autorisations au niveau des colonnes :

- Les utilisateurs sans privilèges SELECT ne peuvent pas lire les colonnes
- Les utilisateurs dotés de privilèges SELECT voient les valeurs masquées conformément à leurs politiques applicables

Comprendre le comportement de masquage dans les fonctions de déclenchement

Lorsque `pg_columnmask` des politiques sont appliquées à des tables, il est important de comprendre comment le masquage interagit avec les fonctions de déclenchement. Les déclencheurs sont des fonctions de base de données qui s'exécutent automatiquement en réponse à certains événements sur une table, tels que les opérations INSERT, UPDATE ou DELETE.

Par défaut, DDM applique différentes règles de masquage en fonction du type de déclencheur :

déclencheurs de table

Les tables de transition sont démasquées : les fonctions de déclenchement des tables ont accès aux données non masquées de leurs tables de transition, tant pour les anciennes que pour les nouvelles versions de lignes

Les propriétaires de tables créent des déclencheurs et sont propriétaires des données, de sorte qu'ils disposent d'un accès complet pour gérer efficacement leurs tables

Afficher les déclencheurs (AU LIEU DES déclencheurs)

Les tables de transition sont masquées : les fonctions de déclenchement des vues voient les données masquées en fonction des autorisations de l'utilisateur actuel

Les propriétaires des vues peuvent être différents des propriétaires des tables de base et doivent respecter les politiques de masquage des tables sous-jacentes

Deux paramètres de configuration au niveau du serveur contrôlent le comportement des déclencheurs avec des tables masquées. Ils ne peuvent être définis que par `rds_superuser` :

- Restreindre les déclencheurs sur les tables masquées : empêche l'exécution de déclencheurs lorsqu'un utilisateur masqué effectue des opérations DML sur des tables avec des politiques de masquage applicables.
- Restreindre les déclencheurs sur les vues contenant des tables masquées : — Empêche l'exécution des déclencheurs sur les vues lorsque la définition de la vue inclut des tables avec des politiques de masquage applicables à l'utilisateur actuel.

Example des différences entre l'application de la fonction à la table et à la vue

L'exemple suivant crée une fonction de déclenchement qui imprime les anciennes et les nouvelles valeurs de ligne, puis montre comment la même fonction se comporte différemment lorsqu'elle est attachée à une table par rapport à une vue.

```
-- Create trigger function
CREATE OR REPLACE FUNCTION print_changes()
    RETURNS TRIGGER AS
    $$
        BEGIN
        RAISE NOTICE 'Old row: name=%, email=%, ssn=%, salary=%',
            OLD.name, OLD.email, OLD.ssn, OLD.salary;
```

```
        RAISE NOTICE 'New row: name=%, email=%, ssn=%, salary=%',
            NEW.name, NEW.email, NEW.ssn, NEW.salary;

        RETURN NEW;
        END;
    $$ LANGUAGE plpgsql;

-- Create trigger
CREATE TRIGGER print_changes_trigger
    BEFORE UPDATE ON hr.employees
    FOR EACH ROW
    EXECUTE FUNCTION print_changes();

-- Grant update to analyst role
GRANT UPDATE ON hr.employees TO analyst_role;

-- Unmasked data must be seen inside trigger even for masked user for the OLD and NEW
-- row passed to trigger function
BEGIN;
SET ROLE mike_analyst;
UPDATE hr.employees SET id = id + 10 RETURNING *;
NOTICE:  Old row: name=John Doe, email=john.doe@example.com, ssn=123-45-6789,
 salary=50000.00
NOTICE:  New row: name=John Doe, email=john.doe@example.com, ssn=123-45-6789,
 salary=50000.00
NOTICE:  Old row: name=Jane Smith, email=jane.smith@example.com, ssn=987-65-4321,
 salary=60000.00
NOTICE:  New row: name=Jane Smith, email=jane.smith@example.com, ssn=987-65-4321,
 salary=60000.00
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
 11 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
 12 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
(2 rows)

ROLLBACK;


-- Triggers on views (which are supposed to see masked data for new/old row)
CREATE VIEW hr.view_over_employees AS SELECT * FROM hr.employees;
GRANT UPDATE, SELECT ON hr.view_over_employees TO analyst_role;

-- Create trigger for this view
```

```
CREATE TRIGGER print_changes_trigger
    INSTEAD OF UPDATE ON hr.view_over_employees
    FOR EACH ROW
    EXECUTE FUNCTION print_changes();

-- Masked new and old rows should be passed to trigger if trigger is on view
BEGIN;
SET ROLE mike_analyst;
UPDATE hr.view_over_employees SET id = id + 10 RETURNING *;
NOTICE:  Old row: name=John Doe, email=john.doe@example.com, ssn=XXX-XX-6789,
 salary=45000.00
NOTICE:  New row: name=John Doe, email=john.doe@example.com, ssn=XXX-XX-6789,
 salary=45000.00
NOTICE:  Old row: name=Jane Smith, email=jane.smith@example.com, ssn=XXX-XX-4321,
 salary=54000.00
NOTICE:  New row: name=Jane Smith, email=jane.smith@example.com, ssn=XXX-XX-4321,
 salary=54000.00
 id |    name    |         email          |     ssn     |  salary
----+------------+------------------------+-------------+----------
 11 | John Doe   | john.doe@example.com   | XXX-XX-6789 | 45000.00
 12 | Jane Smith | jane.smith@example.com | XXX-XX-4321 | 54000.00
(2 rows)
ROLLBACK;
```

Nous vous recommandons de revoir le comportement des déclencheurs avant de les implémenter sur des tables masquées. Les déclencheurs de table ont accès aux données non masquées des tables de transition, tandis que les déclencheurs de vue voient les données masquées.

Example de renommer la politique de masquage

L'exemple suivant montre comment renommer les politiques existantes à l'aide de la `rename_masking_policy` procédure.

```
-- Rename the strict policy
CALL pgcolumnmask.rename_masking_policy(
    'employee_mask_strict',
    'hr.employees',
    'intern_protection_policy'
);

-- Verify the rename
SELECT policyname, roles, weight
    FROM pgcolumnmask.pg_columnmask_policies
```

```
    WHERE tablename = 'employees'
    ORDER BY weight DESC;

        policyname        |     roles      | weight
--------------------------+----------------+--------
 employee_mask_light      | {analyst_role} |    100
 employee_mask_moderate   | {support_role} |     50
 intern_protection_policy | {intern_role}  |     10
```

Example de la modification du poids des politiques

L'exemple suivant montre comment modifier les pondérations des politiques pour modifier leur pondération.

```
-- Change weight of moderate policy
CALL pgcolumnmask.alter_masking_policy(
    'employee_mask_moderate'::NAME,
    'hr.employees'::REGCLASS,
    NULL,    -- Keep existing masking expressions
    NULL,    -- Keep existing roles
    75       -- New weight
);

-- Verify the changes
SELECT policyname, roles, weight
    FROM pgcolumnmask.pg_columnmask_policies
    WHERE tablename = 'employees'
    ORDER BY weight DESC;
        policyname        |     roles      | weight
--------------------------+----------------+--------
 employee_mask_light      | {analyst_role} |    100
 employee_mask_moderate   | {support_role} |     75
 intern_protection_policy | {intern_role}  |     10
```

Example de nettoyage

L'exemple suivant montre comment supprimer toutes les politiques, tables et utilisateurs.

```
-- Drop policies
CALL pgcolumnmask.drop_masking_policy(
    'intern_protection_policy',
    'hr.employees'
);
```

```
CALL pgcolumnmask.drop_masking_policy(
    'employee_mask_moderate',
    'hr.employees'
);

CALL pgcolumnmask.drop_masking_policy(
    'employee_mask_light',
    'hr.employees'
);

-- Drop table and functions
DROP VIEW IF EXISTS hr.view_over_employees;
DROP TABLE IF EXISTS hr.employees;
DROP SCHEMA IF EXISTS hr;
DROP FUNCTION IF EXISTS public.mask_ssn(text);
DROP FUNCTION IF EXISTS public.mask_salary(numeric, numeric);

-- Drop users
DROP USER sarah_intern, lisa_support, mike_analyst,
    ethan_support_intern, john_analyst_intern;
DROP ROLE intern_role, support_role, analyst_role;
```

Configuration du rôle de gestion des politiques de masquage

L'extension `pg_columnmask` de masquage de colonnes PostgreSQL vous permet de déléguer la gestion des politiques de masquage à un rôle spécifique, plutôt que `rds_superuser` d'exiger ou de détenir des privilèges de propriétaire de table. Cela permet de contrôler de manière plus précise les personnes habilitées à créer, modifier et supprimer les politiques de masquage.

Pour configurer le rôle qui bénéficiera des privilèges de gestion des politiques de masquage, procédez comme suit :

1. Création du rôle d'administrateur des politiques — En tant que tel`rds_superuser`, créez un nouveau rôle chargé de gérer les politiques de masquage :

   ```
   CREATE ROLE mask_admin NOLOGIN;
   ```

2. Configurer le paramètre PostgreSQL : dans votre groupe de paramètres de cluster de base de données personnalisé, définissez `pgcolumnmask.policy_admin_rolname` le paramètre de configuration du moteur sur le nom du rôle que vous avez créé :

```
pgcolumnmask.policy_admin_rolname = mask_admin
```

Les paramètres de configuration de ce moteur peuvent être définis dans un groupe de paramètres de cluster de base de données et ne nécessitent pas de redémarrage de l'instance. Pour plus de détails sur la mise à jour des paramètres, consultezModification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

3. Accordez le rôle aux utilisateurs En généralrds_superuser, accordez le mask_admin rôle aux utilisateurs qui devraient être en mesure de gérer les politiques de masquage :

```
CREATE USER alice LOGIN;
CREATE USER bob LOGIN;
GRANT mask_admin TO alice, bob;
```

En outre, assurez-vous que les utilisateurs disposent du privilège USAGE sur les schémas dans lesquels ils géreront les politiques de masquage :

```
GRANT USAGE ON SCHEMA hr TO alice, bob;
```

Désormais, lorsque alice les utilisateurs bob se connectent à la base de données, ils peuvent utiliser les fonctions d'pg_columnmaskextension standard pour créer, modifier et supprimer des politiques de masquage sur toutes les tables de tous les schémas pour lesquels ils ont des USAGE privilèges sur le schéma.

Meilleures pratiques pour une implémentation sécurisée de pg_columnmask

La section suivante présente les meilleures pratiques de sécurité pg_columnmask à mettre en œuvre dans votre environnement Aurora PostgreSQL. Suivez ces recommandations pour :

- Mettre en place une architecture de contrôle d'accès sécurisée basée sur les rôles
- Développez des fonctions de masquage qui préviennent les failles de sécurité
- Comprenez et contrôlez le comportement des déclencheurs à l'aide de données masquées

Architecture de sécurité basée sur les rôles

Définissez une hiérarchie des rôles pour implémenter des contrôles d'accès dans votre base de données. Aurora `pg_columnmask` PostgreSQL renforce ces contrôles en fournissant une couche supplémentaire pour un masquage précis des données au sein de ces rôles.

Créez des rôles dédiés qui correspondent aux fonctions de l'organisation plutôt que d'accorder des autorisations à des utilisateurs individuels. Cette approche améliore l'auditabilité et simplifie la gestion des autorisations au fur et à mesure de l'évolution de votre structure organisationnelle.

Example de créer une hiérarchie des rôles organisationnels

L'exemple suivant crée une hiérarchie des rôles organisationnels avec des rôles dédiés à différentes fonctions, puis affecte des utilisateurs individuels aux rôles appropriés. Dans cet exemple, les rôles organisationnels (analyst_role, support_role) sont créés en premier, puis les utilisateurs individuels obtiennent l'adhésion à ces rôles. Cette structure vous permet de gérer les autorisations au niveau du rôle plutôt que pour chaque utilisateur individuel.

```
-- Create organizational role hierarchy
CREATE ROLE data_admin_role;
CREATE ROLE security_admin_role;
CREATE ROLE analyst_role;
CREATE ROLE support_role;
CREATE ROLE developer_role;

-- Specify security_admin_role as masking policy manager in the DB cluster parameter
-- group pgcolumnmask.policy_admin_rolname = 'security_admin_role'

-- Create specific users and assign to appropriate roles
CREATE USER security_manager;
CREATE USER data_analyst1, data_analyst2;
CREATE USER support_agent1, support_agent2;

GRANT security_admin_role TO security_manager;
GRANT analyst_role TO data_analyst1, data_analyst2;
GRANT support_role TO support_agent1, support_agent2;
```

Mettez en œuvre le principe du moindre privilège en n'accordant que les autorisations minimales nécessaires pour chaque rôle. Évitez d'accorder des autorisations étendues qui pourraient être exploitées si les informations d'identification sont compromises.

```
-- Grant specific table permissions rather than schema-wide access
```

```
GRANT SELECT ON sensitive_data.customers TO analyst_role;
GRANT SELECT ON sensitive_data.transactions TO analyst_role;
-- Do not grant: GRANT ALL ON SCHEMA sensitive_data TO analyst_role;
```

Les administrateurs de politiques ont besoin de USAGE privilèges sur les schémas dans lesquels ils gèrent les politiques de masquage. Accordez ces privilèges de manière sélective, conformément au principe du moindre privilège. Révisez régulièrement les autorisations d'accès aux schémas pour vous assurer que seul le personnel autorisé dispose des capacités de gestion des politiques.

La configuration des paramètres du rôle d'administrateur des politiques est réservée aux administrateurs de base de données uniquement. Ce paramètre ne peut pas être modifié au niveau de la base de données ou de la session, ce qui empêche les utilisateurs non privilégiés de remplacer les attributions des administrateurs de politiques. Cette restriction garantit que le contrôle des politiques de masquage reste centralisé et sécurisé.

Attribuez le rôle d'administrateur des politiques à des personnes spécifiques plutôt qu'à des groupes. Cette approche ciblée garantit un accès sélectif à la gestion des politiques de masquage, car les administrateurs de politiques ont la possibilité de masquer toutes les tables de la base de données.

Développement de fonctions de masquage sécurisées

Développez des fonctions de masquage à l'aide d'une sémantique de liaison précoce afin de garantir un suivi correct des dépendances et de prévenir les vulnérabilités liées aux liaisons tardives, telles que la modification du chemin de recherche pendant l'exécution. Il est recommandé d'utiliser la `BEGIN ATOMIC` syntaxe des fonctions SQL afin de permettre la validation au moment de la compilation (c'est-à-dire la liaison anticipée) et la gestion des dépendances.

```
-- Example - Secure masking function with early binding
CREATE OR REPLACE FUNCTION secure_mask_ssn(input_ssn TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
        SELECT CASE
            WHEN input_ssn IS NULL THEN NULL
            WHEN length(input_ssn) < 4 THEN repeat('X', length(input_ssn))
            ELSE repeat('X', length(input_ssn) - 4) || right(input_ssn, 4)
        END;
    END;
```

Vous pouvez également créer des fonctions immunisées contre les modifications du chemin de recherche en qualifiant explicitement toutes les références d'objets selon le schéma, afin de garantir un comportement cohérent entre les différentes sessions utilisateur.

```
-- Function immune to search path changes
CREATE OR REPLACE FUNCTION data_masking.secure_phone_mask(phone_number TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    AS $$
    SELECT CASE
        WHEN phone_number IS NULL THEN NULL
        WHEN public.length(public.regexp_replace(phone_number, '[^0-9]', '', 'g')) < 10
 THEN 'XXX-XXX-XXXX'
        ELSE public.regexp_replace(
            phone_number,
            '([0-9]{3})[0-9]{3}([0-9]{4})',
            public.concat('\1-XXX-\2')
        )
    END;
    $$;
```

Implémentez la validation des entrées dans les fonctions de masquage afin de gérer les cas extrêmes et d'éviter les comportements inattendus. Incluez toujours la gestion des valeurs NULL et validez les formats d'entrée pour garantir un comportement de masquage cohérent.

```
-- Robust masking function with comprehensive input validation
CREATE OR REPLACE FUNCTION secure_mask_phone(phone_number TEXT)
    RETURNS TEXT
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
        SELECT CASE
            WHEN phone_number IS NULL THEN NULL
            WHEN length(trim(phone_number)) = 0 THEN phone_number
            WHEN length(regexp_replace(phone_number, '[^0-9]', '', 'g')) < 10 THEN
 'XXX-XXX-XXXX'
            ELSE regexp_replace(phone_number, '([0-9]{3})[0-9]{3}([0-9]{4})', '\1-XXX-
\2')
        END;
    END;
```

Comportement des déclencheurs DML avec pg_columnmask

Pour les déclencheurs de tables, les tables de transition seront entièrement démasquées. Pour les déclencheurs de vue (IOT), les tables de transition seront masquées en fonction des autorisations de consultation de l'utilisateur actuel.

Déclencheurs de table avec pg_columnmask

Les déclencheurs reçoivent une table de transition contenant l'ancienne et la nouvelle version des lignes modifiées par la requête DML de lancement. Selon le moment où le déclencheur est déclenché, Aurora PostgreSQL remplit les anciennes et les nouvelles lignes. Par exemple, un `BEFORE INSERT` déclencheur contient uniquement les nouvelles versions des lignes et les anciennes versions vides, car il n'existe aucune ancienne version à référencer.

`pg_columnmask`ne masque pas les tables de transition dans les déclencheurs des tables. Les déclencheurs peuvent utiliser des colonnes masquées à l'intérieur de leur corps et celui-ci voit les données non masquées. Le créateur du déclencheur doit s'assurer de la manière dont le déclencheur est exécuté pour un utilisateur. L'exemple suivant fonctionne correctement dans ce cas.

```
-- Example for table trigger uses masked column in its definition
-- Create a table and insert some rows
CREATE TABLE public.credit_card_table (
    name TEXT,
    credit_card_no VARCHAR(16),
    is_fraud BOOL
);

INSERT INTO public.credit_card_table (name, credit_card_no, is_fraud)
    VALUES
    ('John Doe', '4532015112830366', false),
    ('Jane Smith', '5410000000000000', true),
    ('Brad Smith', '1234567891234567', true);

-- Create a role which will see masked data and grant it privileges
CREATE ROLE intern_user;
GRANT SELECT, DELETE ON public.credit_card_table TO intern_user;

-- Trigger which will silenty skip delete of non fraudelent credit cards
CREATE OR REPLACE FUNCTION prevent_non_fraud_delete()
    RETURNS TRIGGER AS
    $$
```

```
    BEGIN
        IF OLD.is_fraud = false THEN
            RETURN NULL;
        END IF;
        RETURN OLD;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER prevent_non_fraud_delete
    BEFORE DELETE ON credit_card_table
    FOR EACH ROW
    EXECUTE FUNCTION prevent_non_fraud_delete();

CREATE OR REPLACE FUNCTION public.return_false()
    RETURNS BOOLEAN
    LANGUAGE SQL
    IMMUTABLE PARALLEL SAFE STRICT
    BEGIN ATOMIC
      SELECT false;
    END;

-- A masking policy that masks both credit card number and is_fraud column.
-- If we apply masking inside trigger then prevent_non_fraud_delete trigger will
-- allow deleting more rows to masked user (even non fraud ones).
CALL pgcolumnmask.create_masking_policy(
    'mask_credit_card_no_&_is_fraud'::NAME,
    'public.credit_card_table'::REGCLASS,
    JSON_BUILD_OBJECT('credit_card_no', 'pgcolumnmask.mask_text(credit_card_no)',
                      'is_fraud', 'public.return_false()')::JSONB,
    ARRAY['intern_user']::NAME[],
    10::INT
);

-- Test trigger behaviour using intern_user
BEGIN;
SET ROLE intern_user;
-- credit card number & is_fraud is completely masked from intern_user
SELECT * FROM public.credit_card_table;
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 John Doe   | XXXXXXXXXXXXXXXX | f
 Jane Smith | XXXXXXXXXXXXXXXX | f
 Brad Smith | XXXXXXXXXXXXXXXX | f
(3 rows)
```

```
-- The delete trigger lets the intern user delete rows for Jane and Brad even though
-- intern_user sees their is_fraud = false, but the table trigger works with
 original
-- unmasked value
DELETE FROM public.credit_card_table RETURNING *;
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 Jane Smith | XXXXXXXXXXXXXXXX | f
 Brad Smith | XXXXXXXXXXXXXXXX | f
(2 rows)

COMMIT;
```

Le créateur du déclencheur divulgue des données démasquées à l'utilisateur s'il ne fait pas attention aux instructions qu'il utilise dans le corps de son déclencheur. Par exemple, l'utilisation d'un `RAISE NOTICE '%', masked_column;` imprime la colonne à l'utilisateur actuel.

```
-- Example showing table trigger leaking column value to current user
CREATE OR REPLACE FUNCTION leaky_trigger_func()
    RETURNS TRIGGER AS
    $$
    BEGIN
        RAISE NOTICE 'Old credit card number was: %', OLD.credit_card_no;
        RAISE NOTICE 'New credit card number is %', NEW.credit_card_no;
        RETURN NEW;
    END;
    $$ LANGUAGE plpgsql;

CREATE TRIGGER leaky_trigger
    AFTER UPDATE ON public.credit_card_table
    FOR EACH ROW
    EXECUTE FUNCTION leaky_trigger_func();

-- Grant update on column is_fraud to auditor role
-- auditor will NOT HAVE PERMISSION TO READ DATA
CREATE ROLE auditor;
GRANT UPDATE (is_fraud) ON public.credit_card_table TO auditor;

-- Also add auditor role to existing masking policy on credit card table
CALL pgcolumnmask.alter_masking_policy(
    'mask_credit_card_no_&_is_fraud'::NAME,
```

```
    'public.credit_card_table'::REGCLASS,
    NULL::JSONB,
    ARRAY['intern_user', 'auditor']::NAME[],
    NULL::INT
);

-- Log in as auditor
-- [auditor]
-- Update will fail if trying to read data from the table
UPDATE public.credit_card_table
    SET is_fraud = true
    WHERE credit_card_no = '4532015112830366';
ERROR:  permission denied for table cc_table

-- [auditor]
-- But leaky update trigger will still print the entire row even though
-- current user does not have permission to select from public.credit_card_table
UPDATE public.credit_card_table SET is_fraud = true;
NOTICE:  Old credit_card_no was: 4532015112830366
NOTICE:  New credit_card_no is 4532015112830366
```

Déclencheurs sur les vues avec pg_columnmask (au lieu de déclencheurs)

Les déclencheurs ne peuvent être créés que sur des vues dans PostgreSQL. Ils sont utilisés pour exécuter des instructions DML sur des vues qui ne sont pas modifiables. Les tables de transit sont toujours masquées à l'intérieur au lieu d'un déclencheur (IOT), car la vue et les tables de base utilisées dans la requête de vue peuvent avoir des propriétaires différents. Dans ce cas, les tables de base peuvent avoir certaines politiques de masquage applicables au propriétaire de la vue et le propriétaire de la vue doit toujours voir les données masquées des tables de base dans ses déclencheurs. Cela est différent des déclencheurs sur les tables car dans ce cas, le créateur du déclencheur et les données contenues dans les tables appartiennent au même utilisateur, ce qui n'est pas le cas ici.

```
-- Create a view over credit card table
CREATE OR REPLACE VIEW public.credit_card_view
    AS
    SELECT * FROM public.credit_card_table;

-- Truncate credit card table and insert fresh data
TRUNCATE TABLE public.credit_card_table;
INSERT INTO public.credit_card_table (name, credit_card_no, is_fraud)
    VALUES
```

```
    ('John Doe', '4532015112830366', false),
    ('Jane Smith', '5410000000000000', true),
    ('Brad Smith', '1234567891234567', true);

CREATE OR REPLACE FUNCTION public.print_changes()
    RETURNS TRIGGER AS
    $$
    BEGIN
        RAISE NOTICE 'Old row: name=%, credit card number=%, is fraud=%',
            OLD.name, OLD.credit_card_no, OLD.is_fraud;

        RAISE NOTICE 'New row: name=%, credit card number=%, is fraud=%',
            NEW.name, NEW.credit_card_no, NEW.is_fraud;

    RETURN NEW;
   END;
   $$ LANGUAGE plpgsql;

CREATE TRIGGER print_changes_trigger
    INSTEAD OF UPDATE ON public.credit_card_view
    FOR EACH ROW
    EXECUTE FUNCTION public.print_changes();

GRANT SELECT, UPDATE ON public.credit_card_view TO auditor;

-- [auditor]
-- Login as auditor role
BEGIN;

-- Any data coming out from the table will be masked in instead of triggers
-- according to masking policies applicable to current user
UPDATE public.credit_card_view
    SET name = CONCAT(name, '_new_name')
    RETURNING *;
NOTICE:  Old row: name=John Doe, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=John Doe_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
NOTICE:  Old row: name=Jane Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Jane Smith_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
NOTICE:  Old row: name=Brad Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Brad Smith_new_name, credit card number=XXXXXXXXXXXXXXXX, is
 fraud=f
        name         |  credit_card_no  | is_fraud
```

```
--------------------+------------------+----------
 John Doe_new_name   | XXXXXXXXXXXXXXXX | f
 Jane Smith_new_name | XXXXXXXXXXXXXXXX | f
 Brad Smith_new_name | XXXXXXXXXXXXXXXX | f

 -- Any new data going into the table using INSERT or UPDATE command will be
 unmasked
 UPDATE public.credit_card_view
    SET credit_card_no = '9876987698769876'
    RETURNING *;
NOTICE:  Old row: name=John Doe, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=John Doe, credit card number=9876987698769876, is fraud=f
NOTICE:  Old row: name=Jane Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Jane Smith, credit card number=9876987698769876, is fraud=f
NOTICE:  Old row: name=Brad Smith, credit card number=XXXXXXXXXXXXXXXX, is fraud=f
NOTICE:  New row: name=Brad Smith, credit card number=9876987698769876, is fraud=f
    name    |  credit_card_no  | is_fraud
------------+------------------+----------
 John Doe   | 9876987698769876 | f
 Jane Smith | 9876987698769876 | f
 Brad Smith | 9876987698769876 | f

 COMMIT;
```

Niveau de base de données/utilisateur GuCs pour contrôler le comportement des déclencheurs

Deux paramètres de configuration contrôlent le comportement d'exécution des déclencheurs pour les utilisateurs dotés de politiques de masquage applicables. Utilisez ces paramètres pour empêcher les déclencheurs de s'exécuter sur des tables ou des vues masquées lorsque des restrictions de sécurité supplémentaires sont requises. Les deux paramètres sont désactivés par défaut, ce qui permet aux déclencheurs de s'exécuter normalement.

Premier GUC : restriction de déclenchement sur les tables masquées

Spécifications :

- Nom : `pgcolumnmask.restrict_dml_triggers_for_masked_users`
- Type : `boolean`
- Par défaut : `false` (les déclencheurs sont autorisés à être exécutés)

Empêche l'exécution du déclencheur sur les tables masquées pour les utilisateurs masqués lorsque ce paramètre est défini sur TRUE. `pg_columnmask`corrige l'erreur.

Deuxième GUC : restriction du déclenchement du déclenchement sur les vues avec des tables masquées

Spécifications :

- Nom : `pgcolumnmask.restrict_iot_triggers_for_masked_users`
- Type : `boolean`
- Par défaut : `false` (les déclencheurs sont autorisés à être exécutés)

Empêche l'exécution de déclencheurs sur les vues qui incluent des tables masquées dans leur définition pour les utilisateurs masqués lorsqu'elles sont définies sur TRUE.

Ces paramètres fonctionnent indépendamment et sont configurables comme les paramètres de configuration de base de données standard.

Scénarios de déplacement de données pg_columnmask dans Aurora PostgreSQL

`pg_columnmask`le comportement varie selon les différentes opérations de déplacement de données selon que l'opération a lieu au niveau de la couche de stockage, de la logique ou de l'application. Les opérations au niveau du stockage (telles que le clonage) se comportent différemment des opérations logiques (telles que`pg_dump`) et des opérations au niveau des applications (telles que les requêtes FDW). Cette section décrit le comportement de masquage pour les scénarios courants, notamment la réplication, les sauvegardes, les exportations et les migrations, et explique les implications de sécurité de chacun d'entre eux.

Rubriques

- Base de données globale Aurora et répliques de lecture
- Clone de base de données et restauration de snapshots
- Réplication logique
- Déploiements bleu/vert
- Streams Zero-ETL et CDC
- AWS Database Migration Service
- Exportations de données
- Vues et vues matérialisées
- Déchargement et restauration de données
- Enveloppeur de données étrangères

Base de données globale Aurora et répliques de lecture

Les `pg_columnmask` politiques Aurora sont stockées dans les tables du système de base de données au sein du volume du cluster. Toutes les répliques ont accès aux mêmes politiques et renvoient des résultats systématiquement masqués. Pour les déploiements de la base de données globale Aurora, les `pg_columnmask` politiques sont répliquées vers les tables secondaires Régions AWS ainsi que vers les autres tables du système de base de données, garantissant ainsi une protection des données cohérente dans toutes les régions. Pendant les scénarios de basculement, toutes les `pg_columnmask` politiques restent intactes et fonctionnelles.

Clone de base de données et restauration de snapshots

Les opérations de clonage rapide et de restauration de snapshots d'Aurora préservent toutes les `pg_columnmask` politiques, tous les rôles et toutes les configurations dans les tables du système de base de données. La base de données clonée ou restaurée hérite de toutes les politiques existantes du cluster source. Après le clonage ou la restauration, chaque cluster de bases de données applique des `pg_columnmask` politiques indépendantes.

Réplication logique

Lors de la synchronisation initiale, la réplication logique utilise des opérations SQL COPY standard, et `pg_columnmask` les politiques sont appliquées en fonction des autorisations de l'utilisateur de réplication. Pendant le CDC (capture des données de modification) en cours, les politiques de masquage ne sont pas appliquées et les données démasquées sont répliquées via des enregistrements WAL. Les utilisateurs dotés de `pg_create_subscription` privilèges peuvent potentiellement exfiltrer des données non masquées en configurant la réplication vers un système qu'ils contrôlent.

Déploiements bleu/vert

Lors de la restauration des instantanés, les `pg_columnmask` politiques sont automatiquement incluses. L'environnement vert commence par une copie identique de toutes les politiques de l'environnement bleu. Lors de la réplication du bleu au vert, les données ne sont pas masquées. Les modifications ultérieures de la politique de masquage (commandes DDL) sur le cluster bleu ne sont pas répliquées sur le cluster vert et invalident les déploiements RDS. blue/green

Streams Zero-ETL et CDC

La réplication des données n'est pas affectée par `pg_columnmask` les politiques. Zero-ETL prend en charge la réplication DDL mais ne réplique `pg_columnmask` pas les politiques RLS. Aucune politique de masquage n'est appliquée aux données répliquées dans Zero-ETL.

AWS Database Migration Service

La synchronisation initiale des données est masquée ou démasquée en fonction de l'utilisateur sélectionné pour la tâche DMS. Les données du CDC sont toujours démasquées. Bien que les politiques RLS internes `pg_columnmask` associées puissent être migrées, elles ne fonctionneront pas sur les cibles non activées par pg_columnmask.

Exportations de données

`pg_columnmask`traite les exportations comme toute autre opération de requête : le masquage est appliqué en fonction des autorisations de l'utilisateur exécutant. Cela s'applique aux commandes SQL telles que COPY, SELECT INTO, CREATE TABLE AS et à la fonctionnalité d'exportation S3 d'Aurora PostgreSQL.

> **Note**
>
> Lorsque des utilisateurs masqués exportent des données, les fichiers qui en résultent contiennent des valeurs masquées susceptibles de violer les contraintes de la base de données lors de la restauration.

Vues et vues matérialisées

Tenez compte des considérations suivantes lorsque vous utilisez des vues :

- Vues régulières : utilisez toujours la `INVOKER` sémantique. Les politiques de masquage de l'utilisateur actuel s'appliquent lors de l'interrogation de la vue, quel que soit le créateur de la vue.
- Vues matérialisées : lors de l'actualisation, ce sont les politiques de masquage du propriétaire de la vue matérialisée qui s'appliquent, et non celles de l'utilisateur effectuant l'actualisation. Si le propriétaire applique des politiques de masquage, la vue matérialisée contient toujours des données masquées.

Déchargement et restauration de données

`pg_dump`fonctionne comme un utilisateur normal de la base de données et applique des politiques de masquage en fonction des autorisations de l'utilisateur qui se connecte. Si un utilisateur masqué effectue un vidage, le fichier de sauvegarde contient des données masquées. `pg_columnmask`les politiques sont incluses dans le dump en tant que partie intégrante du schéma de base de données.

Une restauration réussie nécessite que tous les rôles référencés existent dans la base de données cible et que l'`pg_columnmask`extension soit installée sur la cible.

> **Note**
>
> À partir de PostgreSQL 18`pg_dump`, prend en charge `–no-policies` l'option qui exclut à la fois la sécurité au niveau des lignes (RLS) `pg_columnmask` et les politiques de masquage des vidages de bases de données. Pour plus d'informations, consultez pg_dump.

Enveloppeur de données étrangères

Lorsque vous utilisez des enveloppeurs de données étrangers, les politiques de masquage sur les tables distantes sont appliquées en fonction des autorisations de l'utilisateur mappé sur le serveur source, et non des autorisations de l'utilisateur demandeur local. Bien que vous puissiez accéder aux données distantes masquées via FDW, vous ne pouvez pas créer de politiques DDM ou RLS directement sur les tables étrangères de votre base de données locale.

Mise à jour des applications pour se connecter aux clusters de bases de données Aurora PostgreSQL à l’aide des nouveaux certificats SSL/TLS

Le 13 janvier 2023, Amazon RDS a publié de nouveaux certificats d’autorité de certification (CA) pour la connexion à vos clusters de bases de données Aurora à l’aide du protocole Secure Socket Layer ou Transport Layer Security (SSL/TLS). Vous trouverez ci-après des informations sur la mise à jour de vos applications afin d’utiliser les nouveaux certificats.

Cette rubrique peut vous aider à déterminer si des applications clientes utilisent le protocole SSL ou TLS pour se connecter à vos clusters de bases de données. Si tel est le cas, il vous est alors possible de vérifier si ces applications nécessitent une vérification du certificat pour se connecter.

> **Note**
>
> Certaines applications sont configurées pour se connecter aux clusters de bases de données Aurora PostgreSQL uniquement si la vérification du certificat sur le serveur s’effectue avec succès.

Pour ces applications, vous devez mettre à jour les magasins d'approbations des applications clientes afin d'inclure les nouveaux certificats de l'autorité de certification.

Une fois que vous avez mis à jour les certificats de l'autorité de certification dans les magasins d'approbations des applications clientes, vous pouvez soumettre les certificats de vos clusters de bases de données à une rotation. Nous vous recommandons vivement de tester ces procédures dans un environnement de développement ou intermédiaire avant de les implémenter dans vos environnements de production.

Pour plus d'informations sur la rotation de certificats, consultez Rotation de votre certificat SSL/TLS. Pour en savoir plus sur le téléchargement de certificats, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters. Pour de plus amples informations sur l'utilisation des protocoles SSL et TLS avec les clusters de bases de données PostgreSQL, consultez Sécurisation des données Aurora PostgreSQL avec SSL/TLS.

Rubriques

- Contrôle de la connexion des applications aux clusters de bases de données Aurora PostgreSQL avec le protocole SSL
- Contrôle de la nécessité d'une vérification du certificat du client pour qu'il puisse se connecter
- Mise à jour du magasin d'approbations de votre application
- Utilisation des connexions SSL/TLS pour différents types d'application

Contrôle de la connexion des applications aux clusters de bases de données Aurora PostgreSQL avec le protocole SSL

Dans la configuration du cluster de bases de données, vérifiez la valeur du paramètre `rds.force_ssl`. Par défaut, le paramètre `rds.force_ssl` est défini sur `0` (désactivé). Si le paramètre `rds.force_ssl` est défini sur `1` (activé), les clients doivent utiliser le protocole SSL/TLS pour se connecter. Pour plus d'informations sur les groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora.

Si `rds.force_ssl` n'a pas la valeur `1` (activé), interrogez la vue `pg_stat_ssl` pour vérifier les connexions établies avec le protocole SSL. Par exemple, la requête suivante retourne uniquement les connexions SSL et les informations sur les clients utilisant un protocole SSL.

```
select datname, usename, ssl, client_addr from pg_stat_ssl inner join pg_stat_activity
 on pg_stat_ssl.pid = pg_stat_activity.pid where ssl is true and usename<>'rdsadmin';
```

Seules les lignes utilisant des connexions SSL/TLS s'affichent avec les informations sur la connexion. Voici un exemple de sortie.

```
 datname  | usename | ssl | client_addr
----------+---------+-----+-------------
 benchdb  | pgadmin | t   | 53.95.6.13
 postgres | pgadmin | t   | 53.95.6.13
(2 rows)
```

La requête précédente n'affiche que les connexions actives au moment de la requête elle-même. L'absence de résultats n'implique pas forcément qu'aucune application n'utilise de connexions SSL. D'autres connexions SSL peuvent avoir été établies à un moment différent.

Contrôle de la nécessité d'une vérification du certificat du client pour qu'il puisse se connecter

Quand un client, tel que psql ou JDBC, est configuré avec la prise en charge du protocole SSL, le comportement par défaut est le suivant : le client essaie d'abord de se connecter à la base de données avec le protocole SSL. En cas d'impossibilité, le client revient à la connexion sans protocole SSL. Le mode `sslmode` utilisé par défaut pour les clients libpq (comme psql) et JDBC est défini sur `prefer`. Le certificat sur le serveur n'est vérifié que lorsque `sslrootcert` est fourni avec `sslmode` défini sur `verify-ca` ou `verify-full`. En cas de non-validité du certificat, une erreur est déclenchée.

Utilisez `PGSSLROOTCERT` pour vérifier le certificat avec la variable d'environnement `PGSSLMODE`, avec `PGSSLMODE` défini sur `verify-ca` ou `verify-full`.

```
PGSSLMODE=verify-full PGSSLROOTCERT=/fullpath/ssl-cert.pem psql -h
 pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com -U primaryuser -d postgres
```

Utilisez l'argument `sslrootcert` pour vérifier le certificat ayant `sslmode` comme format de chaîne de connexion et `sslmode` ayant la valeur `verify-ca` ou `verify-full`.

```
psql "host=pgdbidentifier.cxxxxxxxx.us-east-2.rds.amazonaws.com sslmode=verify-full
 sslrootcert=/full/path/ssl-cert.pem user=primaryuser dbname=postgres"
```

Par exemple, dans le cas précédent, si vous utilisez un certificat racine non valide, une erreur similaire à celle qui suit s'affiche sur votre client.

```
psql: SSL error: certificate verify failed
```

Mise à jour du magasin d'approbations de votre application

Pour plus d'informations sur la mise à jour du magasin d'approbations pour les applications PostgreSQL, consultez Secure TCP/IP Connections with SSL dans la documentation PostgreSQL.

> **Note**
>
> Lors de la mise à jour du magasin d'approbations, vous pouvez conserver les certificats plus anciens en complément de l'ajout des nouveaux certificats.

Mise à jour du magasin d'approbations de votre application pour JDBC

Vous pouvez mettre à jour le magasin d'approbations pour les applications qui utilisent JDBC dans le cadre de connexions SSL/TLS.

Pour plus d'informations sur le téléchargement du certificat racine, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.

Pour obtenir des exemples de scripts qui importent des certificats, consultez Exemple de script pour importer les certificats dans votre magasin d'approbations.

Utilisation des connexions SSL/TLS pour différents types d'application

Les informations suivantes se rapportent à l'utilisation de connexions SSL/TLS pour différents types d'application.

- psql

  Le client est appelé depuis la ligne de commande en spécifiant des options comme chaîne de connexion ou comme variables d'environnement. Pour les connexions SSL/TLS, les options pertinentes sont `sslmode` (variable d'environnement `PGSSLMODE`) et `sslrootcert` (variable d'environnement `PGSSLROOTCERT`).

Pour obtenir la liste complète des options, consultez Parameter Key Words dans la documentation PostgreSQL. Pour obtenir la liste complète des variables d'environnement, consultez Environment Variables dans la documentation PostgreSQL.

- pgAdmin

  Ce client basé sur un navigateur est une interface plus conviviale pour se connecter à une base de données PostgreSQL.

  Pour plus d'informations sur la configuration des connexions, consultez la documentation pgAdmin.

- JDBC

  JDBC permet de se connecter à la base de données avec des applications Java.

  Pour obtenir des informations générales sur la connexion à une base de données PostgreSQL avec JDBC, consultez Connecting to the Database dans la documentation PostgreSQL. Pour plus d'informations sur la connexion avec un protocole SSL/TLS, consultez Configuring the Client dans la documentation PostgreSQL.

- Python

  est une bibliothèque Python bien connue pour se connecter aux bases de données PostgreSQ `psycopg2`.

  Pour plus d'informations sur l'utilisation de `psycopg2`, consultez la documentation psycopg2. Pour obtenir un bref didacticiel sur la connexion à une base de données PostgreSQL, consultez Psycopg2 Tutorial. Vous trouverez des informations sur les options acceptées par la commande de connexion dans The psycopg2 module content.

> **⚠ Important**
>
> Une fois que vous avez déterminé que vos connexions de base de données utilisent SSL/TLS et que vous avez mis à jour le magasin d'approbations de votre application, vous pouvez mettre à jour votre base de données pour utiliser les certificats rds-ca-rsa2048-g1. Pour obtenir des instructions, consultez l'étape 3 dans Mise à jour de votre certificat CA en modifiant votre instance de base de données .

Utilisation de l’authentification Kerberos avec Aurora PostgreSQL

Vous pouvez utiliser Kerberos pour authentifier les utilisateurs lorsqu’ils se connectent à votre cluster qui exécute PostgreSQL. Pour ce faire, configurez votre de cluster de base de données à utiliser AWS Directory Service for Microsoft Active Directory pour l'authentification Kerberos. AWS Directory Service for Microsoft Active Directory est également appelé AWS Managed Microsoft AD. C'est une fonctionnalité disponible avec Directory Service. Pour en savoir plus, consultez Qu'est-ce que c'est Directory Service ? dans le Guide AWS Directory Service d'administration.

Pour commencer, créez un AWS Managed Microsoft AD répertoire pour stocker les informations d'identification des utilisateurs. Vous fournissez ensuite à votre cluster PostgreSQL le domaine d’Active Directory ainsi que d’autres informations. Lorsque les utilisateurs s’authentifient auprès de l’ de cluster de bases de données PostgreSQL, les demandes d’authentification sont transférées vers l’annuaire AWS Managed Microsoft AD .

Vous pouvez gagner du temps et de l’argent en conservant toutes les informations d’identification dans le même annuaire. Vous avez un endroit centralisé de stockage et de gestion des informations d’identification pour plusieurs clusters de base de données. L’utilisation d’un annuaire peut également améliorer votre profil de sécurité global.

Vous pouvez également accéder aux informations d’identification à partir de votre propre annuaire Microsoft Active Directory sur site. Pour ce faire, vous créez une relation de domaine d’approbation afin que l’annuaire AWS Managed Microsoft AD approuve votre annuaire Microsoft Active Directory sur site. De cette façon, vos utilisateurs peuvent accéder à vos clusters PostgreSQL avec la même expérience d’authentification unique (SSO) Windows que lorsqu’ils accèdent aux charges de travail de votre réseau sur site.

Une base de données peut utiliser les authentifications Kerberos Gestion des identités et des accès AWS (IAM) ou à la fois Kerberos et IAM. Toutefois, comme les authentifications Kerberos et IAM fournissent des méthodes d’authentification différentes, un utilisateur de base de données spécifique peut se connecter à une base de données en utilisant uniquement l’une ou l’autre méthode d’authentification, mais pas les deux. Pour plus d’informations sur l’authentification IAM, consultez Authentification de base de données IAM.

> **Note**
>
> RDS pour PostgreSQL ne prend pas en charge l’authentification Kerberos pour les groupes Active Directory.

Rubriques

Disponibilité des régions et des versions

La disponibilité et la prise en charge des fonctionnalités varient selon les versions spécifiques de chaque moteur de base de données, et selon les Régions AWS. Pour plus d’informations sur la disponibilité des versions et des régions d’Aurora PostgreSQL avec l’authentification Kerberos, consultez Authentification Kerberos avec Aurora PostgreSQL.

Présentation de l’authentification Kerberos pour les clusters de base de données PostgreSQL

Pour configurer l’authentification Kerberos pour un cluster de base de données PostgreSQL, exécutez les étapes suivantes, décrites plus en détails par la suite :

1. AWS Managed Microsoft AD À utiliser pour créer un AWS Managed Microsoft AD répertoire. Vous pouvez utiliser le AWS Management Console AWS CLI, le ou l' Directory Service API pour créer le répertoire. Veillez à ouvrir les ports sortants appropriés sur le groupe de sécurité de répertoire afin que le répertoire puisse communiquer avec le cluster.
2. Créez un rôle qui fournit à Amazon Aurora un accès pour passer des appels vers votre AWS Managed Microsoft AD annuaire. Pour ce faire, créez un rôle Gestion des identités et des accès AWS (IAM) qui utilise la politique IAM gérée. `AmazonRDSDirectoryServiceAccess`

   Pour que le rôle IAM autorise l'accès, le point de terminaison AWS Security Token Service (AWS STS) doit être activé dans la AWS région appropriée pour votre AWS compte. AWS STS les points de terminaison sont actifs par défaut dans tous les cas Régions AWS, et vous pouvez les utiliser sans autre action. Pour plus d'informations, consultez la section Activation et désactivation AWS STS dans une AWS région dans le guide de l'utilisateur IAM.
3. Créez et configurez des utilisateurs dans l' AWS Managed Microsoft AD annuaire à l'aide des outils Microsoft Active Directory. Pour plus d'informations sur la création d'utilisateurs dans votre

Active Directory, voir Gérer les utilisateurs et les groupes dans AWS Managed Microsoft AD dans le Guide d'Directory Service administration.

4. Si vous prévoyez de localiser le répertoire et l'instance de base de données dans différents AWS comptes ou clouds privés virtuels (VPCs), configurez le peering VPC. Pour plus d'informations, consultez Qu'est-ce que l'appairage de VPC ? dans le Guide d'appairage de VPC Amazon.
5. Créez ou modifiez un cluster de base de données PostgreSQL depuis la console, la CLI ou l'API RDS à l'aide de l'une des méthodes suivantes :
   - Création et connexion à un cluster de bases de données Aurora PostgreSQL
   - Modification d'un cluster de bases de données Amazon Aurora
   - Restauration à partir d'un instantané de cluster de bases de données
   - Restauration d'un cluster de bases de données à une date définie

   Vous pouvez localiser l' de cluster dans le même Amazon Virtual Private Cloud (VPC) que le répertoire ou dans un autre compte AWS ou VPC. Lors de la création ou de la modification du cluster de base de données PostgreSQL, procédez comme suit :
   - Fournissez l'identifiant du domaine (identifiant `d-*`) qui a été généré lors de la création de votre annuaire.
   - Fournissez le nom du rôle IAM que vous avez créé.
   - Veillez à ce que le groupe de sécurité de l'instance de base de données puisse recevoir le trafic entrant depuis le groupe de sécurité de l'annuaire.
6. Utilisez les informations d'identification de l'utilisateur principal RDS pour vous connecter au cluster de base de données PostgreSQL. Créez l'utilisateur dans PostgreSQL en vue de son identification externe. Les utilisateurs identifiés en externe peuvent se connecter au cluster de base de données PostgreSQL à l'aide de l'authentification Kerberos.

Configuration de l'authentification Kerberos pour les clusters de base de données PostgreSQL

Pour configurer l'authentification Kerberos, procédez comme suit :

Rubriques

- Étape 1 : créer un répertoire à l'aide de AWS Managed Microsoft AD
- Étape 2 : (Facultatif) Créer une relation de confiance entre votre annuaire Active Directory sur site et Directory Service

- Étape 3 : créer un rôle IAM pour Amazon Aurora (Amazon au Directory Service
- Étape 4 : Créer et configurer des utilisateurs
- Étape 5 : Activer le trafic entre VPC entre le répertoire et l'instance de base de données
- Étape 6 : Créer ou modifier une instance de de bases de données PostgreSQL
- Étape 7 : Créer des utilisateurs PostgreSQL pour vos principaux Kerberos
- Étape 8 : Configurer un client PostgreSQL

Étape 1 : créer un répertoire à l'aide de AWS Managed Microsoft AD

Directory Servicecrée un Active Directory entièrement géré dans le AWS cloud. Lorsque vous créez un AWS Managed Microsoft AD annuaire, il Directory Service crée deux contrôleurs de domaine et deux serveurs DNS pour vous. Les serveurs de répertoire sont créés dans des sous-réseaux différents d'un VPC. Cette redondance permet de s'assurer que votre annuaire reste accessible, y compris en cas de défaillance.

Lorsque vous créez un AWS Managed Microsoft AD annuaire, AWS Directory Service exécute les tâches suivantes en votre nom :

- Configuration d'un annuaire Active Directory dans votre VPC.
- Création d'un compte d'administrateur d'annuaire avec le nom d'utilisateur `Admin` et le mot de passe spécifié. Ce compte est utilisé pour gérer votre annuaire.

  > **Important**
  >
  > Assurez-vous d'enregistrer ce mot de passe. Directory Servicene stocke pas ce mot de passe et il ne peut pas être récupéré ou réinitialisé.

- Création d'un groupe de sécurité pour les contrôleurs de l'annuaire. Le groupe de sécurité doit autoriser la communication avec le cluster de base de données PostgreSQL.

Lorsque vous lancezAWS Directory Service for Microsoft Active Directory, AWS crée une unité organisationnelle (UO) qui contient tous les objets de votre répertoire. Cette unité organisationnelle, qui porte le nom NetBIOS que vous avez entré lorsque vous avez créé votre annuaire, est située dans la racine du domaine. La racine du domaine est détenue et gérée parAWS.

Le `Admin` compte créé avec votre AWS Managed Microsoft AD annuaire dispose d'autorisations pour les activités administratives les plus courantes de votre unité d'organisation :

- Création, mise à jour et suppression des utilisateurs
- Ajouter des ressources à votre domaine, comme des serveurs de fichiers ou d'impression, puis attribuer des autorisations pour ces ressources aux utilisateurs dans votre unité organisationnelle
- Créez des conteneurs OUs et des conteneurs supplémentaires
- Déléguer des autorités
- Restaurer des objets supprimés de la corbeille Active Directory
- Exécuter les modules Active Directory et DNS (Domain Name Service) pour Windows PowerShell sur le service Web Active Directory

Le compte `Admin` dispose également de droits pour exécuter les activités suivantes au niveau du domaine :

- Gérer les configurations DNS (ajouter, supprimer ou mettre à jour des enregistrements, des zones et des redirecteurs)
- Afficher les journaux d'événements DNS
- Afficher les journaux d'événements de sécurité

Pour créer un répertoire avec AWS Managed Microsoft AD

1. Dans le panneau de navigation de la console Directory Service, choisissez Directories (Répertoires), puis Set up directory (Configurer le répertoire).
2. Choisissez AWS Managed Microsoft AD. AWS Managed Microsoft ADest la seule option actuellement prise en charge pour une utilisation avec Amazon Aurora
3. Choisissez Suivant.
4. Sur la page Enter directory information (Saisir les détails du répertoire), renseignez les informations suivantes :

   Edition

   Choisissez l'édition qui correspond à vos besoins.

   Nom de DNS de l'annuaire

   Nom complet de l'annuaire, par exemple **`corp.example.com`**.

Nom NetBIOS de l’annuaire

Nom court facultatif pour l’annuaire, par exemple `CORP`.

Description de l’annuaire

Description facultative de l’annuaire.

Mot de passe administrateur

Mot de passe de l’administrateur de l’annuaire. Le processus de création d’un annuaire crée un compte d’administrateur avec le nom d’utilisateur `Admin` et ce mot de passe.

Le mot de passe de l’administrateur de l’annuaire ne peut pas contenir le terme « admin ». Le mot de passe est sensible à la casse et doit comporter entre 8 et 64 caractères. Il doit également contenir au moins un caractère de trois des quatre catégories suivantes :

- Lettres minuscules (a–z)
- Lettres majuscules (A–Z)
- Chiffres (0–9)
- Caractères non alphanumériques (~!@#$%^&*_-+=`|\(){}[]:;"'<>,.?/)

Confirmer le mot de passe

Saisissez à nouveau le mot de passe de l’administrateur.

> **Important**
>
> Assurez-vous d'enregistrer ce mot de passe. Directory Servicene stocke pas ce mot de passe et il ne peut pas être récupéré ou réinitialisé.

5. Choisissez Suivant.
6. Sur la page Choose VPC and subnets (Choisir un VPC et des sous-réseaux), indiquez les informations suivantes :

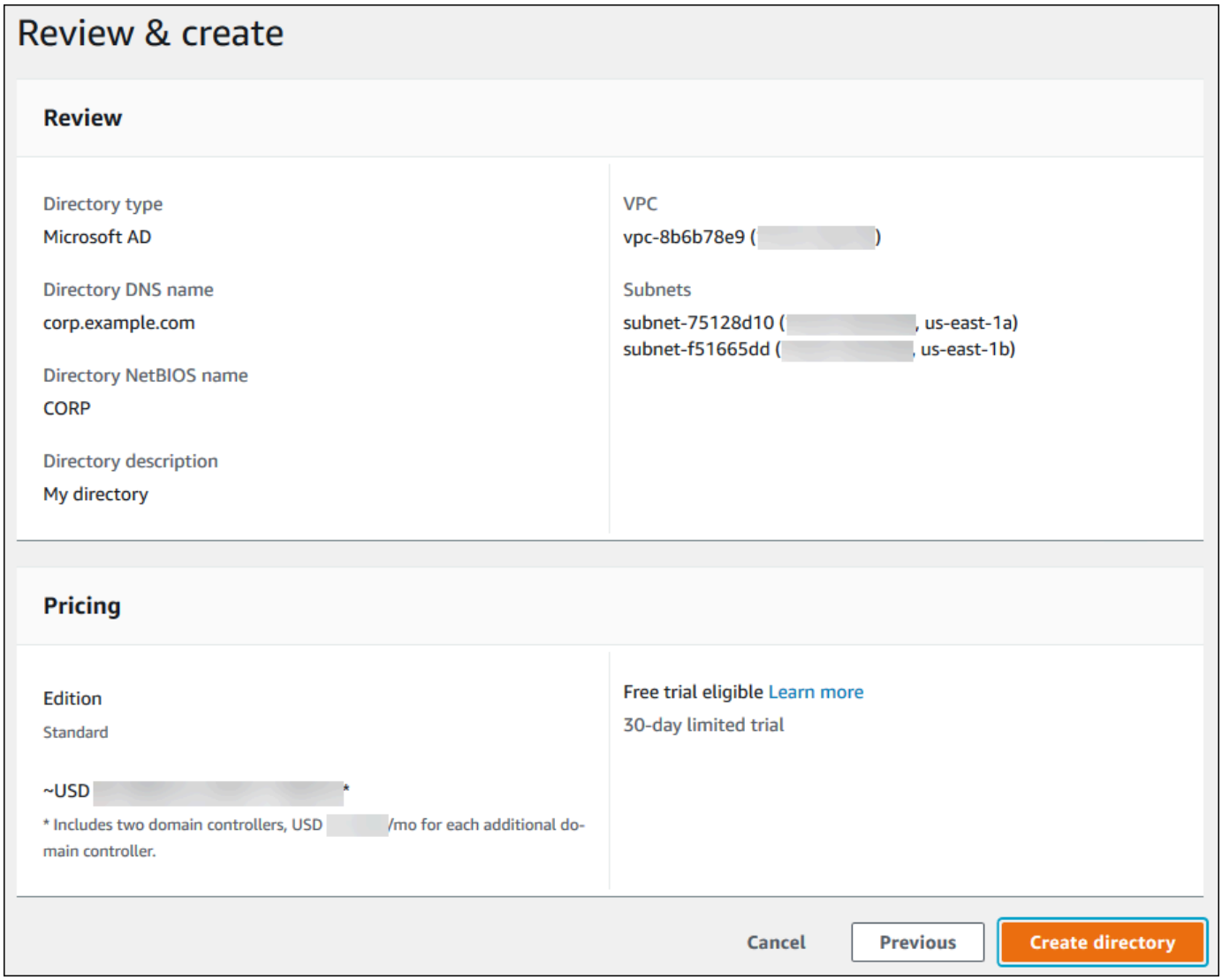
   VPC

   Sélectionnez le VPC pour l’annuaire. Vous pouvez créer le cluster de base de données PostgreSQL dans ce même VPC ou dans un autre VPC.

Sous-réseaux

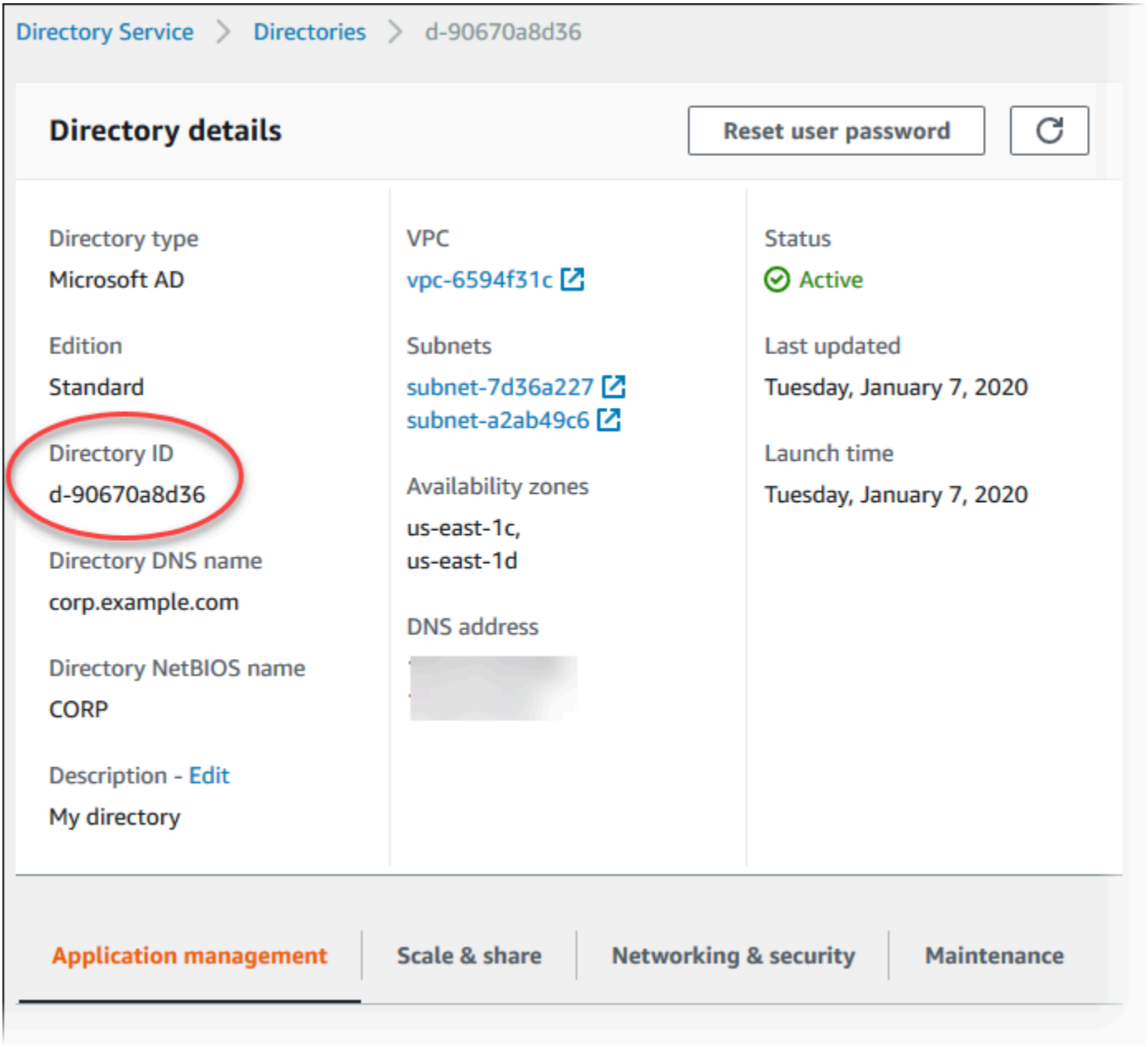
Choisissez les sous-réseaux pour les serveurs d'annuaires. Les deux sous-réseaux doivent être dans des zones de disponibilité différentes.

7. Choisissez Suivant.
8. Vérifiez les informations du répertoire. Si vous devez apporter des modifications, choisissez Previous (Précédent) et entrez ces modifications. Lorsque les informations sont correctes, choisissez Create directory (Créer l'annuaire).



La création de l'annuaire prend plusieurs minutes. Lorsqu'il est créé, la valeur du champ Statut devient Actif.

Pour consulter les informations relatives à votre annuaire, choisissez l'ID de l'annuaire dans la liste. Notez la valeur de Directory ID (ID du répertoire). Vous en aurez besoin pour créer ou modifier votre instance de base de données PostgreSQL.



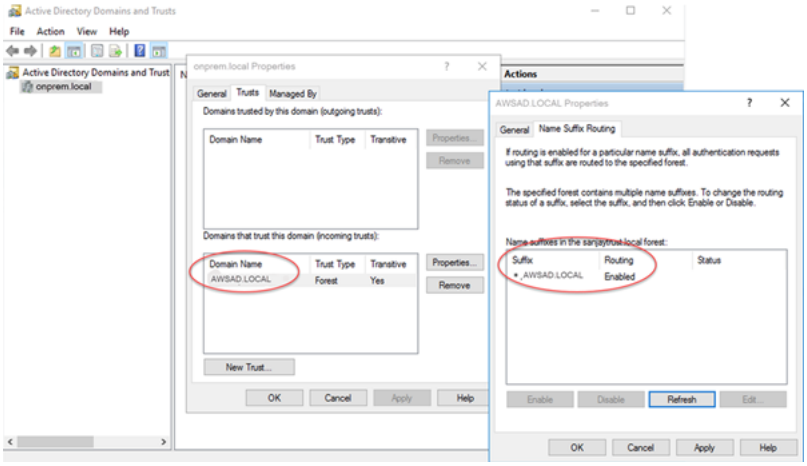
Étape 2 : (Facultatif) Créer une relation de confiance entre votre annuaire Active Directory sur site et Directory Service

Si vous ne prévoyez pas d'utiliser votre propre Microsoft Active Directory sur site, passez à Étape 3 : créer un rôle IAM pour Amazon Aurora (Amazon au Directory Service.

Pour obtenir l'authentification Kerberos à l'aide de votre Active Directory local, vous devez créer une relation de domaine de confiance à l'aide d'une approbation forestière entre votre Microsoft Active Directory local et l'AWS Managed Microsoft ADannuaire (créé dans). Étape 1 : créer un répertoire à l'aide de AWS Managed Microsoft AD La confiance peut être unidirectionnelle, lorsque l'AWS Managed Microsoft ADannuaire fait confiance à Microsoft Active Directory local. L'approbation peut également être bidirectionnelle. Dans ce cas, les deux Active Directory s'approuvent mutuellement. Pour plus d'informations sur la configuration des approbations à l'aide Directory Service de la section Quand créer une relation de confiance dans le Guide d'AWS Directory Serviceadministration.

> **Note**
>
> Si vous utilisez un annuaire Microsoft Active Directory sur site :
>
> - Les clients Windows doivent se connecter en utilisant le nom de domaine du Directory Service endpoint plutôt que rds.amazonaws.com. Pour de plus amples informations, veuillez consulter Connexion à PostgreSQL avec l'authentification Kerberos.
> - Les clients Windows ne peuvent pas se connecter à l'aide de points de terminaison Aurora personnalisés. Pour en savoir plus, consultez Connexions de point de terminaison Amazon Aurora.
> - Pour les bases de données globales :
>   - Les clients Windows peuvent se connecter à l'aide de points de terminaison d'instance ou de points de terminaison de cluster situés uniquement dans la base de données principale Région AWS de la base de données globale.
>   - Les clients Windows ne peuvent pas se connecter à l'aide des points de terminaison du cluster dans le secondaireRégions AWS.

Assurez-vous que le nom de domaine de votre Microsoft Active Directory sur site inclut un routage de suffixe DNS correspondant à la relation d'approbation nouvellement créée. La capture d'écran suivante présente un exemple.



Étape 3 : créer un rôle IAM pour Amazon Aurora (Amazon au Directory Service

Pour qu'Amazon Aurora puisse vous appelerDirectory Service, votre AWS compte a besoin d'un rôle IAM qui utilise la politique IAM gérée. `AmazonRDSDirectoryServiceAccess` Ce rôle permet à Amazon Aurora d'appeler Directory Service. (Notez que le rôle IAM auquel accéder Directory Service est différent du rôle IAM utilisé.) Authentification de base de données IAM

Lorsque vous créez une instance de base de données à l'aide de AWS Management Console et que votre compte utilisateur de console dispose de l'`iam:CreateRole`autorisation, la console crée automatiquement le rôle IAM nécessaire. Dans ce cas, le nom du rôle est `rds-directoryservice-kerberos-access-role`. Sinon, vous devez créer le rôle IAM manuellement. Lorsque vous créez ce rôle IAM`Directory Service`, choisissez et associez la politique AWS gérée `AmazonRDSDirectoryServiceAccess` à celui-ci.

Pour plus d'informations sur la création de rôles IAM pour un service, consultez la section Création d'un rôle pour déléguer des autorisations à un AWS service dans le Guide de l'utilisateur IAM.

> **Note**
>
> Le rôle IAM utilisé pour l'authentification Windows pour RDS for Microsoft SQL Server ne peut pas être utilisé pour Amazon Aurora.

Vous pouvez également créer des stratégies avec les autorisations obligatoires au lieu d'utiliser la politique gérée `AmazonRDSDirectoryServiceAccess`. Dans ce cas, le rôle IAM doit avoir la politique d'approbation IAM suivante :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.amazonaws.com",
          "rds.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Le rôle doit également avoir la politique de rôle IAM suivante.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "ds:DescribeDirectories",
        "ds:AuthorizeApplication",
        "ds:UnauthorizeApplication",
        "ds:GetAuthorizedApplicationDetails"
      ],
    "Effect": "Allow",
    "Resource": "*"
    }
  ]
}
```

Pour l'opt-inRégions AWS, utilisez les principes de service spécifiques à la région dans les politiques de confiance des rôles IAM. Lorsque vous créez une stratégie d’approbation pour les services dans ces régions, indiquez le code de région dans le principal de service.

L’exemple suivant présente une stratégie d’approbation qui fait appel à des principaux de service spécifiques à la région :

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "directoryservice.rds.REGION-CODE.amazonaws.com",
          "rds.REGION-CODE.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Remplacez REGION-CODE par le code de votre région spécifique. Utilisez par exemple les principaux de service suivants pour la région Asie-Pacifique (Melbourne) :

```
"Service": [
  "directoryservice.rds.ap-southeast-4.amazonaws.com",
  "rds.ap-southeast-4.amazonaws.com"
]
```

Étape 4 : Créer et configurer des utilisateurs

Vous pouvez créer des utilisateurs à l’aide de l’outil Active Directory Users and Computers. C’est l’un des outils Active Directory Domain Services et Active Directory Lightweight Directory Services. Pour

plus d'informations, consultez Ajouter des utilisateurs et des ordinateurs au domaine Active Directory dans la documentation Microsoft. Dans ce cas, les utilisateurs sont des individus ou d'autres entités, tels que leurs ordinateurs, qui font partie du domaine et dont les identités sont conservées dans l'annuaire.

Pour créer des utilisateurs dans un Directory Service annuaire, vous devez être connecté à une EC2 instance Amazon basée sur Windows qui est membre de l'Directory Serviceannuaire. Parallèlement, vous devez être connecté en tant qu'utilisateur disposant de privilèges pour créer des utilisateurs. Pour plus d'informations, consultez Créer un utilisateur dans le Guide d'administration AWS Directory Service.

Étape 5 : Activer le trafic entre VPC entre le répertoire et l'instance de base de données

Si vous avez l'intention de rechercher le répertoire et le cluster de base de données dans le même VPC, ignorez cette étape et passez à Étape 6 : Créer ou modifier une instance de de bases de données PostgreSQL.

Si vous prévoyez de localiser le répertoire et l'instance de base de données différemment VPCs, configurez le trafic inter-VPC à l'aide du peering VPC ou de Transit Gateway. AWS

La procédure suivante active le trafic entre les utilisateurs de VPCs l'appairage VPC. Suivez les instructions de Qu'est-ce que l'appairage de VPC ? dans le Guide de l'appairage Amazon Virtual Private Cloud.

Pour activer le trafic entre VPC à l'aide de l'appairage de VPC

1. Configurez les règles de routage de VPC appropriées afin de veiller à ce que le trafic réseau puisse être acheminé dans les deux sens.
2. Veillez à ce que le groupe de sécurité de l'instance de base de données puisse recevoir le trafic entrant depuis le groupe de sécurité de l'annuaire.
3. Assurez-vous qu'il n'existe aucune règle de liste de contrôle d'accès (ACL) pour bloquer le trafic.

Si le répertoire appartient à un autre AWS compte, vous devez le partager.

Pour partager le répertoire entre AWS comptes

1. Commencez à partager le répertoire avec le AWS compte dans lequel l'instance de base de données sera créée en suivant les instructions du Tutoriel : Partage de votre répertoire

Microsoft AD AWS géré pour une connexion de EC2 domaine fluide dans le Guide d'Directory Serviceadministration.

2. Connectez-vous à la Directory Service console à l'aide du compte de l'instance de base de données et assurez-vous que le domaine possède le `SHARED` statut requis avant de continuer.
3. Lorsque vous êtes connecté à la Directory Service console à l'aide du compte de l'instance de base de données, notez la valeur de l'ID du répertoire. Vous utilisez cet ID d'annuaire pour joindre l'instance de base de données au domaine.

Étape 6 : Créer ou modifier une instance de de bases de données PostgreSQL

Créez ou modifiez un cluster de base de données PostgreSQL en vue de son utilisation avec votre répertoire. Vous pouvez utiliser la console, la CLI ou l'API RDS pour associer un cluster de base de données à un répertoire. Vous pouvez effectuer cette opération de différentes manières :

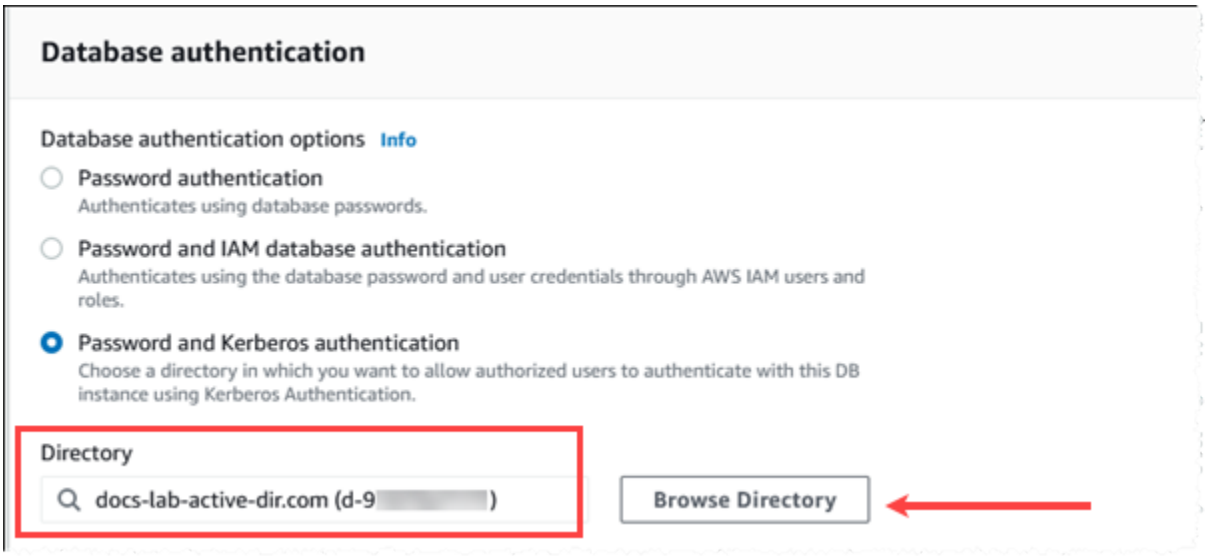
- Créez un nouveau cluster de base de données PostgreSQL à l'aide de la console, de la commande create-db-clusterCLI ou de l'opération DBCluster Create RDS API. Pour obtenir des instructions, veuillez consulter Création et connexion à un cluster de bases de données Aurora PostgreSQL.
- Modifiez un cluster de base de données PostgreSQL existant à l'aide de la console, de la commande modify-db-clusterCLI ou de l'opération DBCluster Modify RDS API. Pour obtenir des instructions, veuillez consulter Modification d'un cluster de bases de données Amazon Aurora.
- Restaurez un cluster de base de données PostgreSQL à partir d'un instantané de base de données à l'aide de la console, de la commande CLI restore-db-cluster-from-db-snapshot ou de l'opération Restore From RDS API. DBCluster DBSnapshot Pour obtenir des instructions, veuillez consulter Restauration à partir d'un instantané de cluster de bases de données.
- Restaurez un cluster de base de données PostgreSQL à point-in-time l'aide de la console, de la commande restore-db-instance-to- point-in-time CLI ou de l'opération DBCluster ToPointInTime Restore RDS API. Pour obtenir des instructions, veuillez consulter Restauration d'un cluster de bases de données à une date définie.

L'authentification Kerberos est uniquement prise en charge pour les clusters de base de données PostgreSQL dans un VPC. Le cluster de base de données peut se trouver dans le même VPC que le répertoire ou dans un autre VPC. Le cluster de base de données doit utiliser un groupe de sécurité qui accepte les entrées et les sorties dans le VPC du répertoire pour permettre au cluster de base de données de communiquer avec le répertoire.

> **Note**
>
> L'activation de l'authentification Kerberos n'est actuellement pas prise en charge sur le cluster de bases de données Aurora PostgreSQL lors de la migration à partir de RDS pour PostgreSQL. Vous ne pouvez activer l'authentification Kerberos que sur un cluster de bases de données Aurora PostgreSQL autonome.

Console

Lorsque vous utilisez la console pour créer, modifier ou restaurer un cluster de bases de données, choisissez Kerberos authentication (Authentification Kerberos) dans la section Database authentication (Authentification de base de données). Ensuite, choisissez Browse Directory (Parcourir le répertoire). Sélectionnez le répertoire ou choisissez Create a new directory (Créer un nouveau répertoire) pour utiliser Directory Service.



AWS CLI

Lorsque vous utilisez leAWS CLI, les paramètres suivants sont requis pour que l' de cluster de base de données puisse utiliser le répertoire que vous avez créé :

- Pour le paramètre `--domain`, vous devez indiquer l'identifiant du domaine (identifiant « d-* ») généré lors de la création de l'annuaire.
- Pour le paramètre `--domain-iam-role-name`, utilisez le rôle que vous avez créé qui utilise la politique IAM gérée `AmazonRDSDirectoryServiceAccess`.

Par exemple, la commande de CLI suivante modifie un cluster de base de données de façon à utiliser un répertoire.

```
aws rds  modify-db-cluster --db-cluster-identifier mydbinstance --domain d-Directory-ID
 --domain-iam-role-name role-name
```

> **Important**
>
> Si vous modifiez un cluster de base de données de façon à activer l'authentification Kerberos, redémarrez le cluster de base de données après avoir effectué la modification.

Étape 7 : Créer des utilisateurs PostgreSQL pour vos principaux Kerberos

À ce stade, votre cluster de bases de données Aurora PostgreSQL est joint au domaine AWS Managed Microsoft AD. Les utilisateurs que vous avez créés dans l'annuaire dans Étape 4 : Créer et configurer des utilisateurs doivent être configurés en tant qu'utilisateurs de base de données PostgreSQL et bénéficier de privilèges leur permettant de se connecter à la base de données. Pour ce faire, vous devez vous connecter en tant qu'utilisateur de base de données doté de privilèges `rds_superuser`. Par exemple, si vous avez accepté les valeurs par défaut lors de la création de votre cluster de bases de données Aurora PostgreSQL, vous utilisez `postgres`, comme indiqué dans les étapes suivantes.

Pour créer des utilisateurs de base de données PostgreSQL pour les principaux Kerberos

1. Utilisez `psql` pour vous connecter à votre point de terminaison d'instance de base de données du cluster de bases de données Aurora PostgreSQL à l'aide de `psql`. L'exemple suivant utilise le compte `postgres` par défaut pour le rôle `rds_superuser`.

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

2. Créez un nom d'utilisateur de base de données pour chaque principal Kerberos (nom d'utilisateur Active Directory) auquel vous souhaitez accorder l'accès à la base de données. Utilisez le nom d'utilisateur canonique (identité) tel que défini dans l'instance Active Directory, c'est-à-dire un `alias` en minuscule (nom d'utilisateur dans Active Directory) et le nom en majuscule du domaine Active Directory pour ce nom d'utilisateur. Le nom d'utilisateur Active Directory est un utilisateur authentifié de manière externe. Utilisez donc des guillemets autour du nom, comme indiqué ci-dessous.

```
postgres=> CREATE USER "username@CORP.EXAMPLE.COM" WITH LOGIN;
CREATE ROLE
```

3. Accordez le rôle `rds_ad` à l'utilisateur de la base de données.

```
postgres=> GRANT rds_ad TO "username@CORP.EXAMPLE.COM";
GRANT ROLE
```

Une fois que vous avez fini de créer tous les utilisateurs PostgreSQL pour vos identités utilisateur Active Directory, les utilisateurs peuvent accéder au cluster de bases de données Aurora PostgreSQL à l'aide de leurs informations d'identification Kerberos.

Les utilisateurs de base de données qui s'authentifient à l'aide de Kerberos doivent utiliser des machines clientes membres du domaine Active Directory.

Les utilisateurs de base de données auxquels le rôle `rds_ad` a été attribué ne peuvent pas disposer également du rôle `rds_iam`. Cela s'applique également aux adhésions imbriquées. Pour plus d'informations, consultez Authentification de base de données IAM.

Configuration de votre cluster de bases de données Aurora PostgreSQL pour les noms d'utilisateur insensibles à la casse

Les versions 14.5, 13.8, 12.12 et 11.17 d'Aurora PostgreSQL prennent en charge le paramètre PostgreSQL `krb_caseins_users`. Ce paramètre prend en charge les noms d'utilisateur Active Directory insensibles à la casse. Par défaut, ce paramètre est défini sur false, de sorte que les noms d'utilisateur sont interprétés par Aurora PostgreSQL en tenant compte de la casse. Il s'agit du comportement par défaut de toutes les anciennes versions d'Aurora PostgreSQL. Toutefois, vous pouvez définir ce paramètre sur `true` dans le groupe de paramètres de votre cluster de bases de données personnalisé et autoriser votre cluster de bases de données Aurora PostgreSQL à interpréter les noms d'utilisateur, sans tenir compte de la casse. Pensez à le faire pour faciliter la tâche de vos utilisateurs de base de données, qui peuvent parfois se tromper dans la casse de leur nom d'utilisateur lorsqu'ils s'authentifient à l'aide d'Active Directory.

Pour modifier le paramètre `krb_caseins_users`, votre cluster de bases de données Aurora PostgreSQL doit utiliser un groupe de paramètres de cluster de bases de données personnalisé. Pour obtenir des informations sur l'utilisation d'un groupe de paramètres de cluster de bases de données, consultez Groupes de paramètres pour Amazon Aurora.

Vous pouvez utiliser le AWS CLI ou le AWS Management Console pour modifier le réglage. Pour de plus amples informations, veuillez consulter Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

Étape 8 : Configurer un client PostgreSQL

Pour configurer un client PostgreSQL, procédez comme suit :

- Créez un fichier krb5.conf (ou équivalent) pointant vers le domaine.
- Vérifiez que le trafic peut circuler entre l'hôte client etDirectory Service. Utilisez un utilitaire réseau tel que Netcat pour les opérations suivantes :
  - Vérifiez le trafic via DNS pour le port 53.
  - Vérifiez le trafic dépassé TCP/UDP pour le port 53 et pour Kerberos, qui inclut les ports 88 et 464 pour. Directory Service
- Vérifiez que le trafic peut circuler entre l'hôte du client et l'instance de base de données via le port de la base de données. Par exemple, utilisez psql pour vous connecter à la base de données et y accéder.

Voici un exemple de contenu du fichier krb5.conf pour. AWS Managed Microsoft AD

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```

Vous trouverez ci-après un exemple de contenu krb5.conf pour un Microsoft Active Directory sur site.

```
[libdefaults]
 default_realm = EXAMPLE.COM
[realms]
 EXAMPLE.COM = {
  kdc = example.com
  admin_server = example.com
```

```
 }
 ONPREM.COM = {
  kdc = onprem.com
  admin_server = onprem.com
 }
[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
 .onprem.com = ONPREM.COM
 onprem.com = ONPREM.COM
 .rds.amazonaws.com = EXAMPLE.COM
 .amazonaws.com.cn = EXAMPLE.COM
 .amazon.com = EXAMPLE.COM
```

Gestion d'un cluster de bases de données Aurora PostgreSQL dans un domaine Active Directory

Vous pouvez utiliser la console, la CLI ou l'API RDS pour gérer votre cluster de bases de données et sa relation avec Microsoft Active Directory. Par exemple, vous pouvez associer un annuaire Active Directory de façon à activer l'authentification Kerberos. Vous pouvez également supprimer l'association d'un annuaire Active Directory pour désactiver l'authentification Kerberos. Vous pouvez également transférer un cluster de base de données vers un autre élément de même type afin de subir une authentification en externe par un annuaire Microsoft Active Directory.

Par exemple, la CLI vous permet d'effectuer les actions suivantes :

- Pour retenter l'activation de l'authentification Kerberos en cas d'échec d'appartenance, utilisez la commande de CLI modify-db-cluster. Spécifiez l'ID d'annuaire du membre actuel pour l'option `--domain`.
- Pour désactiver l'authentification Kerberos sur une instance de base de données, utilisez la commande de CLI modify-db-cluster. Spécifiez `none` pour l'option `--domain`.
- Pour transférer une instance de base de données d'un domaine vers un autre, utilisez la commande de CLI modify-db-cluster. Spécifiez l'identifiant du nouveau domaine pour l'option `--domain`.

Présentation de l'appartenance au domaine

Une fois que vous avez créé ou modifié votre cluster de base de données, les instances de base de données deviennent membres du domaine. Vous pouvez consulter le statut de l'appartenance

au domaine pour l'instance de base de données dans la console ou en exécutant la commande describe-db-instances de la CLI. Le statut de l'instance de base de données peut avoir les valeurs suivantes :

- `kerberos-enabled` – L'instance de base de données a l'authentification Kerberos activée.
- `enabling-kerberos` – AWS est le processus d'activation de l'authentification Kerberos sur cette instance de base de données.
- `pending-enable-kerberos` – L'activation de l'authentification Kerberos est en attente sur cette instance de base de données.
- `pending-maintenance-enable-kerberos` – AWS tentera d'activer l'authentification Kerberos sur cette instance de base de données lors de la prochaine fenêtre de maintenance planifiée.
- `pending-disable-kerberos` – La désactivation de l'authentification Kerberos est en attente sur cette instance de base de données.
- `pending-maintenance-disable-kerberos` – AWS tentera de désactiver l'authentification Kerberos sur cette instance de base de données lors de la prochaine fenêtre de maintenance planifiée.
- `enable-kerberos-failed` – Un problème de configuration a empêché AWS d'activer l'authentification Kerberos sur l'instance de base de données. Corrigez le problème de configuration avant de réémettre la commande de modification de l'instance de base de données.
- `disabling-kerberos` – AWS est en train de désactiver l'authentification Kerberos sur cette instance de base de données.

Une demande d'activation de l'authentification Kerberos peut échouer à cause d'un problème de connectivité réseau ou d'un rôle IAM incorrect. Dans certains cas, la tentative d'activation de l'authentification Kerberos peut échouer lorsque vous créez ou modifiez un cluster de base de données. Si tel est le cas, vérifiez que vous utilisez le rôle IAM correct, puis modifiez le cluster de base de données afin d'effectuer son rattachement au domaine.

Connexion à PostgreSQL avec l'authentification Kerberos

Vous pouvez vous connecter à PostgreSQL via l'authentification Kerberos avec l'interface pgAdmin ou avec une CLI telle que psql. Pour plus d'informations sur la connexion, consultez Connexion à un cluster de bases de données Amazon Aurora PostgreSQL. Pour plus d'informations sur l'obtention du point de terminaison, du numéro de port et d'autres détails nécessaires à la connexion, consultez Affichage des points de terminaison d'un cluster Aurora.

> **Note**
>
> L'authentification et le chiffrement GSSAPI dans PostgreSQL sont implémentés par la bibliothèque Kerberos `libkrb5.so`. Des fonctionnalités telles que `postgres_fdw` et `dblink` s'appuient également sur cette même bibliothèque pour établir des connexions sortantes nécessitant une authentification ou un chiffrement Kerberos.

pgAdmin

Pour vous connecter à PostgreSQL avec l'authentification Kerberos en utilisant pgAdmin, procédez comme suit :

1. Lancez l'application pgAdmin sur votre ordinateur client.
2. Dans l'onglet Dashboard (Tableau de bord), choisissez Add New Server (Ajouter un nouveau serveur).
3. Dans la boîte de dialogue Create - Server (Créer – Serveur), entrez un nom sur l'onglet General (Général) pour identifier le serveur dans pgAdmin.
4. Dans l'onglet Connection (Connexion), entrez les informations suivantes à partir de votre base de données Aurora PostgreSQL .
   - Pour Host (Hôte), entrez le point de terminaison de l'instance en écriture de votre cluster de bases de données Aurora PostgreSQL. Un point de terminaison ressemble à ce qui suit :

     ```
     AUR-cluster-instance.111122223333.aws-region.rds.amazonaws.com
     ```

     Pour se connecter à un Microsoft Active Directory sur site à partir d'un client Windows, vous utilisez le nom de domaine de l'Active Directory AWS géré au lieu de `rds.amazonaws.com` du point de terminaison de l'hôte. Par exemple, supposons que le nom de domaine pour AWS Managed Active Directory soit `corp.example.com`. Puis, pour Host (Hôte), le point de terminaison serait spécifié comme suit :

     ```
     AUR-cluster-instance.111122223333.aws-region.corp.example.com
     ```

   - Pour Port, entrez le port attribué.
   - Pour Maintenance database (Base de données de maintenance), entrez le nom de la base de données initiale à laquelle le client se connectera.

- Pour Username (Nom d'utilisateur), saisissez le nom d'utilisateur que vous avez entré pour l'authentification Kerberos dans Étape 7 : Créer des utilisateurs PostgreSQL pour vos principaux Kerberos .

5. Choisissez Enregistrer.

Psql

Pour vous connecter à PostgreSQL avec l'authentification Kerberos en utilisant psql, procédez comme suit :

1. A partir d'une invite de commande, exécutez la commande suivante.

   ```
   kinit username
   ```

   Remplacez *username* par le nom de l'utilisateur. À l'invite, entrez le mot de passe stocké dans le Microsoft Active Directory pour l'utilisateur.

2. Si le cluster de bases de données PostgreSQL utilise un VPC accessible au public, placez une adresse IP pour le point de terminaison de votre cluster de bases de données dans votre fichier `/etc/hosts` sur le client EC2. Par exemple, les commandes suivantes permettent d'obtenir l'adresse IP et de la placer dans le fichier `/etc/hosts`.

   ```
   % dig +short PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com
   ;; Truncated, retrying in TCP mode.
   ec2-34-210-197-118.AWS-Region.compute.amazonaws.com.
   34.210.197.118

   % echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.rds.amazonaws.com" >> /etc/
   hosts
   ```

   Si vous utilisez une instance Microsoft Active Directory sur site à partir d'un client Windows, vous devez vous connecter à l'aide d'un point de terminaison spécifique. Au lieu d'utiliser le domaine Amazon `rds.amazonaws.com` dans le point de terminaison hôte, utilisez le nom de domaine d'AWS Managed Active Directory.

   Par exemple, supposons que le nom de domaine de votre AWS Managed Active Directory soit `corp.example.com`. Alors, utilisez le format *PostgreSQL-endpoint.AWS-Region.corp.example.com* pour le point de terminaison et placez-le dans le fichier `/etc/hosts`.

```
% echo " 34.210.197.118  PostgreSQL-endpoint.AWS-Region.corp.example.com" >> /etc/
hosts
```

3. Utilisez la commande psql suivante pour vous connecter à un cluster de base de données PostgreSQL intégré(e) à Active Directory. Utilisez un point de terminaison de cluster ou d'instance.

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.rds.amazonaws.com postgres
```

Pour vous connecter au cluster de bases de données PostgreSQL à partir d'un client Windows à l'aide d'un Active Directory sur site, utilisez la commande psql suivante avec le nom de domaine de l'étape précédente (`corp.example.com`):

```
psql -U username@CORP.EXAMPLE.COM -p 5432 -h PostgreSQL-endpoint.AWS-
Region.corp.example.com postgres
```

Utilisation de groupes de sécurité AD pour le contrôle d’accès Aurora PostgreSQL

À partir des versions 14.10 et 15.5 d’Aurora PostgreSQL, le contrôle d’accès Aurora PostgreSQL peut être géré à l’aide des groupes de sécurité AWS Directory Service for Microsoft Active Directory (AD). Les versions antérieures d’Aurora PostgreSQL prennent en charge l’authentification basée sur Kerberos avec AD seulement pour les utilisateurs individuels. Chaque utilisateur AD devait être explicitement ajouté au cluster de bases de données pour pouvoir y accéder.

Au lieu d’ajouter explicitement chaque utilisateur AD au cluster de bases de données en fonction des besoins de l’entreprise, vous pouvez tirer parti des groupes de sécurité AD comme expliqué ci-dessous :

- Les utilisateurs AD sont membres de différents groupes de sécurité AD dans un annuaire Active Directory. Ils ne sont pas régis par l’administrateur du cluster de bases de données, mais sont basés sur les exigences de l’entreprise et sont gérés par un administrateur AD.
- Les administrateurs de clusters de bases de données créent des rôles de base de données dans les instances de base de données en fonction des exigences de l’entreprise. Ces rôles de base de données peuvent avoir des autorisations ou des privilèges distincts.
- Les administrateurs de clusters de bases de données configurent un mappage entre les groupes de sécurité AD et les rôles de base de données pour chaque cluster de bases de données.
- Les utilisateurs de bases de données peuvent accéder aux clusters de bases de données à l’aide de leurs informations d’identification AD. L’accès est basé sur l’appartenance au groupe de sécurité AD. Les utilisateurs AD obtiennent ou perdent automatiquement l’accès en fonction de leur appartenance à un groupe AD.

Prérequis

Assurez-vous d’effectuer les étapes suivantes avant de configurer l’extension pour les groupes de sécurité AD :

- Configuration de l’authentification Kerberos pour les clusters de bases de données PostgreSQL. Pour plus d’informations, consultez Configuration de l’authentification Kerberos pour les clusters de bases de données PostgreSQL.

> **Note**
>
> Pour les groupes de sécurité AD, ignorez l'étape 7 : Créer des utilisateurs PostgreSQL pour vos principaux Kerberos dans cette procédure de configuration.

> **Important**
>
> Si vous activez des groupes de sécurité AD sur un cluster Aurora PostgreSQL sur lequel l'authentification Kerberos est déjà activée, vous pouvez rencontrer des problèmes d'authentification. Cela se produit lorsque vous ajoutez `pg_ad_mapping` au paramètre `shared_preload_libraries` et que vous redémarrez la base de données. Lorsque vous utilisez des points de terminaison de cluster, les tentatives de connexion à l'aide d'un compte AD qui n'est pas un utilisateur de base de données doté du rôle `rds_ad` peuvent échouer. Cela peut également provoquer des pannes du moteur. Pour résoudre ce problème, désactivez puis réactivez l'authentification Kerberos sur le cluster. Cette solution de contournement est requise pour les instances existantes mais n'affecte pas les instances créées après avril 2025.

- Gestion d'un cluster de bases de données dans un domaine. Pour plus d'informations, consultez Gestion d'un cluster de bases de données dans un domaine.

Configuration de l'extension pg_ad_mapping

Aurora PostgreSQL fournit désormais une extension `pg_ad_mapping` pour gérer le mappage entre les groupes de sécurité AD et les rôles de base de données dans le cluster Aurora PostgreSQL. Pour plus d'informations sur les fonctions fournies par `pg_ad_mapping`, consultez Utilisation des fonctions de l'extension `pg_ad_mapping`.

Pour configurer l'extension `pg_ad_mapping` sur votre cluster de bases de données Aurora PostgreSQL, vous ajoutez d'abord `pg_ad_mapping` aux bibliothèques partagées au niveau du groupe de paramètres de base de données personnalisé pour votre cluster de bases de données Aurora PostgreSQL. Pour plus d'informations sur la création d'un groupe de paramètres de cluster de bases de données personnalisé, consultez Groupes de paramètres pour Amazon Aurora. Ensuite, vous installez l'extension `pg_ad_mapping`. Les procédures de cette section vous guident. Pour ce faire, vous pouvez utiliser la AWS Management Console ou la AWS CLI.

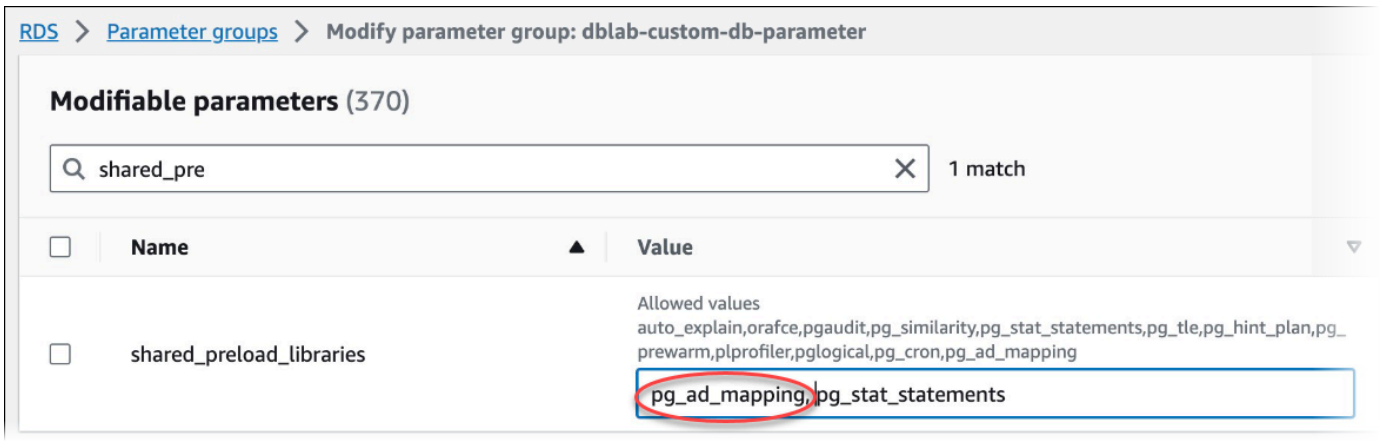
Vous devez disposer d'autorisations en tant que rôle `rds_superuser` pour effectuer toutes ces tâches.

Les étapes suivantes supposent que votre cluster de bases de données Aurora PostgreSQL est associé à un groupe de paramètres de cluster de bases de données personnalisés.

Console

Pour configurer l'extension **`pg_ad_mapping`**

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL.
3. Ouvrez l'onglet Configuration de l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL. Parmi les détails de l'instance, trouvez le lien Groupe de paramètres.
4. Cliquez sur le lien pour ouvrir les paramètres personnalisés associés à votre cluster de bases de données Aurora PostgreSQL.
5. Dans le champ de recherche Parameters (Paramètres), tapez `shared_pre` pour trouver le paramètre `shared_preload_libraries`.
6. Choisissez Edit parameters (Modifier les paramètres) pour accéder aux valeurs des propriétés.
7. Ajoutez `pg_ad_mapping` à la liste dans le champ Values (Valeurs). Utilisez une virgule pour séparer les éléments de la liste de valeurs.



8. Redémarrez l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL afin que vos modifications du paramètre `shared_preload_libraries` prennent effet.

9. Lorsque l'instance est disponible, vérifiez que `pg_ad_mapping` a été initialisé. Utilisez `psql` pour vous connecter à l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL, puis exécutez la commande suivante.

```
SHOW shared_preload_libraries;
shared_preload_libraries
-------------------------
rdsutils,pg_ad_mapping
(1 row)
```

10. Une fois `pg_ad_mapping` initialisé, vous pouvez maintenant créer l'extension. Vous devez créer l'extension après avoir initialisé la bibliothèque pour pouvoir commencer à utiliser les fonctions fournies par cette extension.

```
CREATE EXTENSION pg_ad_mapping;
```

11. Fermez la session `psql`.

```
labdb=> \q
```

AWS CLI

Pour configurer pg_ad_mapping

Pour configurer pg_ad_mapping à l'aide de l'AWS CLI, appelez l'opération modify-db-parameter-group pour ajouter ce paramètre dans votre groupe de paramètres personnalisé, comme indiqué dans la procédure suivante.

1. Utilisez la commande AWS CLI suivante pour ajouter `pg_ad_mapping` au paramètre `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_ad_mapping,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilisez la commande AWS CLI suivante pour redémarrer l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL afin que pg_ad_mapping soit initialisé.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Lorsque l'instance est disponible, vous pouvez vérifier que `pg_ad_mapping` a été initialisé. Utilisez `psql` pour vous connecter à l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL, puis exécutez la commande suivante.

```
SHOW shared_preload_libraries;
shared_preload_libraries
-------------------------
rdsutils,pg_ad_mapping
(1 row)
```

   Une fois pg_ad_mapping initialisé, vous pouvez maintenant créer l'extension.

```
CREATE EXTENSION pg_ad_mapping;
```

4. Fermez la session `psql` afin de pouvoir utiliser l'AWS CLI.

```
labdb=> \q
```

Récupération du SID du groupe Active Directory dans PowerShell

Un identifiant de sécurité (SID) permet d'identifier de manière unique un principal ou un groupe de sécurité. Chaque fois qu'un groupe de sécurité ou un compte est créé dans Active Directory, un SID lui est attribué. Pour récupérer le SID du groupe de sécurité AD depuis Active Directory, vous pouvez utiliser l'applet de commande Get-ADGroup à partir de la machine cliente Windows associée à ce domaine Active Directory. Le paramètre Identity spécifie le nom du groupe Active Directory pour obtenir le SID correspondant.

L'exemple suivant renvoie le SID du groupe AD *adgroup1*.

```
C:\Users\Admin> Get-ADGroup -Identity adgroup1 | select SID

            SID
-----------------------------------------------
```

```
S-1-5-21-3168537779-1985441202-1799118680-1612
```

Mappage du rôle de base de données avec le groupe de sécurité AD

Vous devez explicitement configurer les groupes de sécurité AD dans la base de données en tant que rôles de base de données PostgreSQL. Un utilisateur AD faisant partie d'au moins un groupe de sécurité AD provisionné aura accès à la base de données. Vous ne devez pas accorder `rds_ad role` à un rôle de base de données basé sur le groupe de sécurité AD. L'authentification Kerberos pour le groupe de sécurité sera déclenchée en utilisant le suffixe du nom de domaine tel que *`user1@example.com.`* Ce rôle de base de données ne peut pas utiliser l'authentification par mot de passe ni l'authentification IAM pour accéder à la base de données.

> **Note**
>
> Les utilisateurs AD qui ont un rôle de base de données correspondant dans la base de données et auxquels le rôle `rds_ad` a été accordé ne pourront pas se connecter dans le cadre du groupe de sécurité AD. Ils y auront accès via le rôle de base de données en tant qu'utilisateur individuel.

Prenons l'exemple d'accounts-group, qui est un groupe de sécurité dans AD que vous souhaitez provisionner dans Aurora PostgreSQL en tant que rôle de compte (accounts-role).

Groupe de sécurité AD	Rôle de base de données PostgreSQL
accounts-group	accounts-role

Lorsque vous mappez le rôle de base de données avec le groupe de sécurité AD, vous devez vous assurer que l'attribut LOGIN du rôle de base de données est défini et que ce rôle dispose du privilège CONNECT pour la base de données de connexion requise.

```
postgres => alter role accounts-role login;

ALTER ROLE
postgres => grant connect on database accounts-db to accounts-role;
```

L'administrateur peut maintenant procéder à la création du mappage entre le groupe de sécurité AD et le rôle de base de données PostgreSQL.

```
admin=>select pgadmap_set_mapping('accounts-group', 'accounts-role', <SID>, <Weight>);
```

Pour plus d'informations sur la récupération du SID du groupe de sécurité AD, consultez Récupération du SID du groupe Active Directory dans PowerShell.

Dans certains cas, un utilisateur AD peut appartenir à plusieurs groupes. Dans ce cas, l'utilisateur AD hérite des privilèges du rôle de base de données, qui a été attribué avec le poids le plus élevé. Si les deux rôles ont le même poids, l'utilisateur AD hérite des privilèges du rôle de base de données correspondant au mappage récemment ajouté. Il est recommandé de spécifier des poids qui reflètent les autorisations/privilèges relatifs des rôles de base de données individuels. Plus les autorisations ou privilèges d'un rôle de base de données sont élevés, plus le poids qui doit être associé à l'entrée de mappage est élevé. Cela permet d'éviter l'ambiguïté de deux mappages ayant le même poids.

Le tableau suivant présente un exemple de mappage entre les groupes de sécurité AD et les rôles de base de données Aurora PostgreSQL.

Groupe de sécurité AD	Rôle de base de données PostgreSQL	Weight
accounts-group	accounts-role	7
sales-group	sales-role	10
dev-group	dev-role	7

Dans l'exemple suivant, `user1` hérite des privilèges de sales-role puisqu'il a le poids le plus élevé, tandis que `user2` hérite des privilèges de `dev-role`, car le mappage pour ce rôle été créé après `accounts-role`, qui partage le même poids que `accounts-role`.

Nom d'utilisateur	Appartenance au groupe de sécurité
user1	accounts-group sales-group
user2	accounts-group dev-group

Les commandes psql permettant d'établir, de répertorier et d'effacer les mappages sont présentées ci-dessous. Actuellement, il n'est pas possible de modifier une seule entrée de mappage. L'entrée existante doit être supprimée, et le mappage recréé.

```
admin=>select pgadmap_set_mapping('accounts-group', 'accounts-role', 'S-1-5-67-890',
 7);
admin=>select pgadmap_set_mapping('sales-group', 'sales-role', 'S-1-2-34-560', 10);
admin=>select pgadmap_set_mapping('dev-group', 'dev-role', 'S-1-8-43-612', 7);

admin=>select * from pgadmap_read_mapping();

ad_sid       | pg_role        | weight | ad_grp
-------------+----------------+--------+---------------
S-1-5-67-890 | accounts-role | 7      | accounts-group
S-1-2-34-560 | sales-role    | 10     | sales-group
S-1-8-43-612 | dev-role      | 7      | dev-group
(3 rows)
```

Journalisation/audit de l'identité des utilisateurs AD

Utilisez la commande suivante pour déterminer le rôle de base de données hérité par l'utilisateur actuel ou par l'utilisateur de la session :

```
postgres=>select session_user, current_user;

session_user | current_user
-------------+--------------
dev-role     | dev-role

(1 row)
```

Pour déterminer l'identité du principal de sécurité AD, utilisez la commande suivante :

```
postgres=>select principal from pg_stat_gssapi where pid = pg_backend_pid();
```

```
 principal
-------------------------
 user1@example.com

(1 row)
```

À l'heure actuelle, l'identité de l'utilisateur AD n'est pas visible dans les journaux d'audit. Le paramètre `log_connections` peut être activé pour enregistrer l'établissement d'une session de base de données. Pour plus d'informations, consultez log_connections. La sortie correspondante inclut l'identité de l'utilisateur AD, comme indiqué ci-dessous. Le PID du système dorsal associé à cette sortie pourra ensuite aider à attribuer les actions à l'utilisateur AD réel.

```
pgrole1@postgres:[615]:LOG: connection authorized: user=pgrole1
 database=postgres application_name=psql GSS (authenticated=yes, encrypted=yes,
 principal=Admin@EXAMPLE.COM)
```

Limitations

- Microsoft Entra ID connu sous le nom d'Azure Active Directory n'est pas pris en charge.

Utilisation des fonctions de l'extension **`pg_ad_mapping`**

L'extension `pg_ad_mapping` a pris en charge les fonctions suivantes :

pgadmap_set_mapping

Cette fonction établit le mappage entre le groupe de sécurité AD et le rôle de base de données avec un poids associé.

Syntaxe

```
pgadmap_set_mapping(
ad_group,
db_role,
ad_group_sid,
weight)
```

Arguments

Paramètre	Description
ad_group	Nom du groupe AD. La valeur ne peut pas être une chaîne nulle ou vide.
db_role	Rôle de base de données à mapper au groupe AD spécifié. La valeur ne peut pas être une chaîne nulle ou vide.
ad_group_sid	Identifiant de sécurité utilisé pour identifier de manière unique le groupe AD. Cette valeur commence par « S-1- » et ne peut pas être une chaîne nulle ou vide. Pour plus d'informations, consultez Récupération du SID du groupe Active Directory dans PowerShell.
weight	Poids associé au rôle de base de données. Le rôle ayant le plus de poids est prioritaire lorsque l'utilisateur est membre de plusieurs groupes. La valeur par défaut de weight est 1.

Type de retour

None

Notes d'utilisation

Cette fonction ajoute un nouveau mappage entre le groupe de sécurité AD et le rôle de base de données. Elle ne peut être exécutée que sur l'instance de base de données principale du cluster de bases de données par un utilisateur disposant du privilège rds_superuser.

Exemples

```
postgres=> select pgadmap_set_mapping('accounts-group','accounts-
role','S-1-2-33-12345-67890-12345-678',10);

pgadmap_set_mapping

(1 row)
```

pgadmap_read_mapping

Cette fonction répertorie les mappages entre le groupe de sécurité AD et le rôle de base de données définis à l'aide de la fonction `pgadmap_set_mapping`.

Syntaxe

```
pgadmap_read_mapping()
```

Arguments

None

Type de retour

Paramètre	Description
ad_group_sid	Identifiant de sécurité utilisé pour identifier de manière unique le groupe AD. Cette valeur commence par « S-1- » et ne peut pas être une chaîne nulle ou vide. Pour plus d'informations, consultez Récupération du SID du groupe Active Directory dans PowerShell.accounts-role@example.com
db_role	Rôle de base de données à mapper au groupe AD spécifié. La valeur ne peut pas être une chaîne nulle ou vide.
weight	Poids associé au rôle de base de données. Le rôle ayant le plus de poids est prioritaire lorsque l'utilisateur est membre de plusieurs groupes. La valeur par défaut de weight est 1.
ad_group	Nom du groupe AD. La valeur ne peut pas être une chaîne nulle ou vide.

Notes d'utilisation

Appelez cette fonction pour répertorier tous les mappages disponibles entre le groupe de sécurité AD et le rôle de base de données.

Exemples

```
postgres=> select * from pgadmap_read_mapping();

ad_sid                               | pg_role       | weight | ad_grp
-------------------------------------+---------------+--------+------------------
S-1-2-33-12345-67890-12345-678       | accounts-role | 10     | accounts-group
(1 row)

(1 row)
```

pgadmap_reset_mapping

Cette fonction réinitialise un ou tous les mappages définis à l'aide de la fonction `pgadmap_set_mapping`.

Syntaxe

```
pgadmap_reset_mapping(
ad_group_sid,
db_role,
weight)
```

Arguments

Paramètre	Description
ad_group_sid	Identifiant de sécurité utilisé pour identifier de manière unique le groupe AD.
db_role	Rôle de base de données à mapper au groupe AD spécifié.
weight	Poids associé au rôle de base de données.

Si aucun argument n'est fourni, tous les mappages entre le groupe AD et le rôle de base de données sont réinitialisés. Tous les arguments doivent être fournis ou aucun.

Type de retour

None

Notes d’utilisation

Appelez cette fonction pour supprimer un mappage spécifique entre un groupe AD et un rôle de base de données ou pour réinitialiser tous les mappages. Cette fonction ne peut être exécutée que sur l’instance de base de données principale du cluster de bases de données par un utilisateur disposant du privilège `rds_superuser`.

Exemples

```
postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+-------------------
 S-1-2-33-12345-67890-12345-678 | accounts-role| 10          | accounts-group
 S-1-2-33-12345-67890-12345-666 | sales-role   | 10          | sales-group

(2 rows)
postgres=> select pgadmap_reset_mapping('S-1-2-33-12345-67890-12345-678', 'accounts-
role', 10);

pgadmap_reset_mapping
(1 row)

postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+---------------
 S-1-2-33-12345-67890-12345-666 | sales-role   | 10          | sales-group

(1 row)
postgres=> select pgadmap_reset_mapping();

pgadmap_reset_mapping
(1 row)

postgres=> select * from pgadmap_read_mapping();

   ad_sid                        | pg_role      | weight      | ad_grp
---------------------------------+--------------+-------------+--------------
 (0 rows)
```

Migration des données vers Amazon Aurora avec compatibilité PostgreSQL

Vous avez plusieurs options pour la migration des données à partir de votre base de données existante vers un cluster de bases de données Édition compatible avec Amazon Aurora PostgreSQL. Vos options de migration dépendent également de la base de données à partir de laquelle vous effectuez la migration et de la taille des données que vous migrez. Les options qui s'offrent à vous sont les suivantes :

Migration d'une instance de base de données RDS pour PostgreSQL à l'aide d'un instantané

Vous pouvez migrer des données directement d'un instantané de base de données RDS pour PostgreSQL vers un cluster de bases de données Aurora PostgreSQL.

Migration d'une instance de base de données RDS pour PostgreSQL à l'aide d'un réplica en lecture Aurora

Vous pouvez aussi migrer à partir d'une instance de base de données RDS pour PostgreSQL en créant un réplica en lecture Aurora PostgreSQL d'une instance de base de données RDS pour PostgreSQL. Lorsque le retard de réplica entre l'instance de base de données RDS pour PostgreSQL et le réplica en lecture Aurora PostgreSQL est égal à zéro, vous pouvez arrêter la réplication. À ce stade, vous pouvez faire du réplica en lecture Aurora un cluster de bases de données Aurora PostgreSQL autonome pour la lecture et l'écriture.

Importation de données depuis Amazon S3 vers Aurora PostgreSQL

Vous pouvez migrer des données en les important à partir d'Amazon S3 dans une table appartenant à un cluster de bases de données Aurora PostgreSQL.

Migration à partir d'une base de données qui n'est pas compatible avec PostgreSQL

Vous pouvez utiliser AWS Database Migration Service (AWS DMS) pour migrer les données d'une base de données qui n'est pas compatible avec PostgreSQL. Pour plus d'informations sur AWS DMS, consultez Présentation d'AWS Database Migration Service dans le Guide de l'utilisateur AWS Database Migration Service.

> **Note**
>
> L'activation de l'authentification Kerberos n'est actuellement pas prise en charge sur le cluster de bases de données Aurora PostgreSQL lors de la migration à partir de RDS pour

PostgreSQL. Vous ne pouvez activer l'authentification Kerberos que sur un cluster de bases de données Aurora PostgreSQL autonome.

Pour obtenir la liste des Régions AWS où Aurora est disponible, consultez Amazon Aurora dans la Références générales AWS.

Important

Si vous prévoyez de migrer une instance de base de données RDS pour PostgreSQL vers un cluster de bases de données Aurora PostgreSQL dans un avenir proche, nous vous recommandons vivement de désactiver les mises à niveau automatiques mineures de version pour l'instance de base de données dès le début de la phase de planification de la migration. La migration vers Aurora PostgreSQL peut être retardée si la version de RDS pour PostgreSQL n'est pas encore prise en charge par Aurora PostgreSQL.
Pour plus d'informations sur Aurora PostgreSQL les versions, consultez Versions du moteur pour Amazon Aurora PostgreSQL.

Migration d'un instantané d'une instance de base de données RDS pour PostgreSQL vers un cluster de bases de données Aurora PostgreSQL.

Pour créer un cluster de bases de données Aurora PostgreSQL, vous pouvez migrer un instantané d'une instance de base de données RDS pour PostgreSQL. Le nouveau cluster de bases de données Aurora PostgreSQL est rempli avec les données de l'instance de base de données RDS pour PostgreSQL d'origine. Pour plus d'informations sur la création d'un instantané de base de données, consultez Création d'un instantané de base de données.

Dans certains cas, l'instantané de bases de données peut ne pas se trouver dans la Région AWS où vous souhaitez localiser vos données. Dans ce cas, utilisez la console Amazon RDS pour copier l'instantané de bases de données vers cette Région AWS. Pour plus d'informations sur la copie d'un instantané de base de données, consultez Copie d'un instantané.

Vous pouvez migrer des instantanés RDS pour PostgreSQL compatibles avec les versions d'Aurora PostgreSQL disponibles dans la Région AWS donnée. Par exemple, vous pouvez migrer un instantané d'instance de base de données RDS pour PostgreSQL 11.1 vers Aurora PostgreSQL version 11.4, 11.7, 11.8 ou 11.9 dans la Région USA Ouest (Californie du Nord). Vous pouvez

migrer un instantané RDS PostgreSQL 10.11 vers Aurora PostgreSQL 10.11, 10.12, 10.13 et 10.14. Autrement dit, l'instantané RDS pour PostgreSQL doit utiliser une version mineure identique ou inférieure à celle d'Aurora PostgreSQL.

Vous pouvez aussi choisir que votre nouveau cluster de bases de données Aurora PostgreSQL soit chiffré au repos à l'aide d'une AWS KMS key. Cette option est uniquement disponible pour les instantanés de base de données non chiffrés.

Pour migrer un instantané de base de données RDS pour PostgreSQL vers un cluster de bases de données Aurora PostgreSQL, vous pouvez utiliser la console AWS Management Console, l'AWS CLI ou l'API RDS. Lorsque vous utilisez la AWS Management Console, celle-ci prend les mesures nécessaires pour créer le cluster de bases de données et l'instance principale.

Console

Pour migrer un instantané de base de données PostgreSQL à l'aide de la console RDS

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Instantanés.
3. Sur la page Instantanés, sélectionnez l'instantané RDS pour PostgreSQL que vous voulez migrer vers un cluster de bases de données Aurora PostgreSQL.
4. Sélectionnez Actions, puis Migrer l'instantané.
5. Définissez les valeurs suivantes sur la page Migrer la base de données :

   - Version du moteur de base de données : choisissez la version du moteur de base de données que vous souhaitez utiliser pour la nouvelle instance migrée.
   - Identifiant d'instance de base de données : saisissez un nom de cluster de bases de données, unique pour votre compte dans la Région AWS choisie. Cet identifiant est utilisé dans les adresses de point de terminaison des instances de votre cluster DB. Vous pouvez choisir de complexifier le nom, par exemple en incluant la Région AWS et le moteur de base de données que vous avez choisis, comme **`aurora-cluster1`**.

     L'identifiant d'instance de base de données obéit aux contraintes suivantes :

     - Il doit contenir entre 1 et 63 caractères alphanumériques ou traits d'union.
     - Son premier caractère doit être une lettre.
     - Il ne peut pas se terminer par un trait d'union ou contenir deux traits d'union consécutifs.

  - Il doit être unique pour toutes les instances de base de données par compte AWS et par Région AWS.
- Classe d'instance de base de données : sélectionnez une classe d'instance de base de données qui possède le stockage et la capacité requis pour votre base de données. Par exemple, `db.r6g.large`. Les volumes de cluster Aurora croissent automatiquement au fur et à mesure que la quantité de données de votre base de données augmente. Par conséquent, vous devez uniquement choisir une classe d'instance de base de données qui correspond à vos besoins de stockage actifs. Pour plus d'informations, consultez Présentation du stockage Amazon Aurora.
- Virtual Private Cloud (VPC) : si vous disposez d'un VPC existant, vous pouvez l'utiliser avec votre cluster de bases de données Aurora PostgreSQL en sélectionnant ici votre identifiant VPC, soit par exemple `vpc-a464d1c1`. Pour plus d'informations sur la création d'un VPC, consultez Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement).

  Vous pouvez solliciter la création d'un VPC par Amazon RDS, en sélectionnant Créer un VPC.
- Groupe de sous-réseaux de base de données : si vous avez un groupe de sous-réseaux existant, vous pouvez utiliser ce groupe de sous-réseaux avec votre cluster de base de données Aurora PostgreSQL en sélectionnant ici votre identifiant de groupe de sous-réseaux, par exemple `gs-subnet-group1`.
- Accessible publiquement : sélectionnez Non pour configurer les instances de votre cluster de bases de données de façon à ce qu'elles ne soient accessibles que par les ressources situées à l'intérieur de votre VPC. Choisissez Oui pour spécifier que les instances de votre cluster de bases de données sont accessibles par les ressources du réseau public.

  > **Note**
  >
  > Il n'est pas nécessaire que votre cluster de bases de données de production se trouve dans un sous-réseau public, parce que seuls vos serveurs d'applications nécessitent l'accès à votre cluster de bases de données. Si votre cluster de base de données n'a pas besoin d'être dans un sous-réseau public, définissez Accessible publiquement sur Non.
- Groupe de sécurité VPC : sélectionnez un groupe de sécurité VPC pour autoriser l'accès à votre base de données.

- Zone de disponibilité : choisissez la zone de disponibilité devant héberger l'instance principale de votre cluster de bases de données Aurora PostgreSQL. Pour qu'Amazon RDS choisisse une Zone de disponibilité pour vous, sélectionnez Aucune préférence.
- Port de la base de données : saisissez le port par défaut à utiliser lors de la connexion aux instances du cluster de bases de données Aurora PostgreSQL. La valeur par défaut est 5432.

  > **Note**
  >
  > Il se peut que vous soyez derrière un pare-feu d'entreprise qui n'autorise pas l'accès aux ports par défaut, tels que le port par défaut PostgreSQL 5432. Dans ce cas, fournissez une valeur de port que votre pare-feu d'entreprise autorise. Souvenez-vous plus tard de cette valeur de port lors de votre connexion au cluster de bases de données Aurora PostgreSQL.

- Chiffrement : sélectionnez Activer le chiffrement pour que votre nouveau cluster de bases de données Aurora PostgreSQL soit chiffré au repos. Choisissez également une clé KMS comme valeur de AWS KMS key.
- Mise à niveau automatique des versions mineures : choisissez Enable auto minor version upgrade (Activer la mise à niveau automatique de versions mineures) si vous voulez permettre à votre cluster de bases de données Aurora PostgreSQL de recevoir automatiquement les mises à niveau de versions mineures du moteur de base de données PostgreSQL lorsqu'elles deviennent disponibles.

  L'option Mise à niveau automatique des versions mineures ne s'applique qu'aux mises à niveau vers les versions mineures du moteur PostgreSQL pour votre cluster de bases de données Aurora PostgreSQL. Elle ne s'applique pas aux correctifs réguliers appliqués pour maintenir la stabilité du système.

6. Choisissez Migrer pour migrer votre instantané de base de données.
7. Sélectionnez Bases de données pour afficher le nouveau cluster de bases de données. Sélectionnez le nouveau cluster de bases de données pour surveiller la progression de la migration. Lorsque la migration est terminée, le statut du cluster est Available (Disponible). Sous l'onglet Connectivité et sécurité, vous trouverez le point de terminaison du cluster à utiliser pour la connexion à l'instance de scripteur principale du cluster de bases de données. Pour plus d'informations sur la connexion à un cluster de bases de données Aurora PostgreSQL, consultez Connexion à un cluster de bases de données Amazon Aurora.

AWS CLI

L'utilisation de l'AWS CLI pour migrer un instantané de bases de données RDS pour PostgreSQL vers Aurora PostgreSQL implique deux commandes AWS CLI séparées. Tout d'abord, utilisez la commande `restore-db-cluster-from-snapshot` de l'AWS CLI afin de créer un cluster de bases de données Aurora PostgreSQL. Vous utilisez ensuite la commande `create-db-instance` pour créer l'instance de base de données primaire dans le nouveau cluster afin de terminer la migration. La procédure suivante crée un cluster de bases de données Aurora PostgreSQL avec une instance de base de données primaire ayant la même configuration que l'instance de base de données utilisée pour créer l'instantané.

Pour migrer un instantané de bases de données RDS pour PostgreSQL vers un cluster de bases de données Aurora PostgreSQL.

1. Utilisez la commande describe-db-snapshots pour obtenir des informations sur l'instantané de bases de données que vous souhaitez migrer. Vous pouvez spécifier le paramètre `--db-instance-identifier` ou `--db-snapshot-identifier` dans la commande. Si vous ne spécifiez pas l'un de ces paramètres, vous obtenez tous les instantanés.

   ```
   aws rds describe-db-snapshots --db-instance-identifier <your-db-instance-name>
   ```

2. La commande renvoie tous les détails de configuration des instantanés créés à partir de l'instance de base de données spécifiée. Dans la sortie, recherchez l'instantané que vous souhaitez migrer et localisez son Amazon Resource Name (ARN). Pour en savoir plus sur les ARN Amazon RDS, consultez Amazon Relational Database Service (Amazon RDS). Un ARN est similaire à ce qui suit.

   ```
   “DBSnapshotArn": "arn:aws:rds:aws-region:111122223333:snapshot:<snapshot_name>"
   ```

   Vous trouverez également dans la sortie les détails de configuration de l'instance de base de données RDS pour PostgreSQL, tels que la version du moteur, le stockage alloué, si l'instance de base de données est chiffrée ou non, etc.

3. Utilisez la commande restore-db-cluster-from-snapshot pour démarrer la migration. Spécifiez les paramètres suivants :

   - `--db-cluster-identifier` : nom que vous souhaitez donner au cluster de bases de données Aurora PostgreSQL. Ce cluster de bases de données Aurora est la cible de votre migration d'instantanés de bases de données.

- `--snapshot-identifier` : l'Amazon Resource Name (ARN) de l'instantané de bases de données à migrer.
- `--engine` : spécifiez `aurora-postgresql` pour le moteur de cluster de bases de données Aurora.
- `--kms-key-id` : ce paramètre facultatif vous permet de créer un cluster de bases de données Aurora PostgreSQL chiffré à partir d'un instantané de bases de données non chiffré. Il vous permet également de choisir une clé de chiffrement différente, pour le cluster de bases de données ; de celle utilisée pour l'instantané de bases de données.

> **Note**
>
> Vous ne pouvez pas créer de cluster de bases de données Aurora PostgreSQL non chiffré à partir d'un instantané de bases de données chiffré.

Sans le paramètre `--kms-key-id` spécifié comme illustré ci-après, la commande restore-db-cluster-from-snapshot AWS CLI crée un cluster de bases de données Aurora PostgreSQL vide, chiffré à l'aide de la même clé que l'instantané de bases de données, ou non chiffré si l'instantané de bases de données source n'est pas chiffré.

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier cluster-name \
    --snapshot-identifier arn:aws:rds:aws-region:111122223333:snapshot:your-
snapshot-name \
    --engine aurora-postgresql
```

Pour Windows :

```
aws rds restore-db-cluster-from-snapshot ^
    --db-cluster-identifier new_cluster ^
    --snapshot-identifier arn:aws:rds:aws-region:111122223333:snapshot:your-
snapshot-name ^
    --engine aurora-postgresql
```

4. La commande renvoie des détails sur le cluster de bases de données Aurora PostgreSQL créé pour la migration. Vous pouvez vérifier l'état du cluster de bases de données Aurora PostgreSQL à l'aide de la commande describe-db-clusters de l'AWS CLI.

```
aws rds describe-db-clusters --db-cluster-identifier cluster-name
```

5. Lorsque le cluster de bases de données devient « disponible », utilisez la commande create-db-instance pour remplir le cluster de bases de données Aurora PostgreSQL avec l'instance de base de données basée sur votre instantané de bases de données Amazon RDS. Spécifiez les paramètres suivants :

   - `--db-cluster-identifier` : nom du nouveau cluster de bases de données Aurora PostgreSQL que vous avez créé à l'étape précédente.
   - `--db-instance-identifier` : nom que vous souhaitez donner à l'instance de base de données. Cette instance devient le nœud primaire de votre cluster de bases de données Aurora PostgreSQL.
   - `----db-instance-class` : spécifiez la classe d'instance de base de données à utiliser. Choisissez parmi les classes d'instance de base de données prises en charge par la version Aurora PostgreSQL vers laquelle vous migrez. Pour plus d'informations, consultez Types de classes d'instance de base de données et Moteurs de base de données pris en charge pour les classes d'instance de base de données.
   - `--engine` : spécifiez `aurora-postgresql` pour l'instance de base de données

   Vous pouvez également créer l'instance de base de données avec une configuration différente de celle de l'instantané de bases de données source, en transmettant les options appropriées dans la commande `create-db-instance` de l'AWS CLI. Pour plus d'informations, consultez la commande create-db-instance.

   Pour Linux, macOS ou Unix :

```
aws rds create-db-instance \
    --db-cluster-identifier cluster-name \
    --db-instance-identifier --db-instance-class db.instance.class \
    --engine aurora-postgresql
```

   Pour Windows :

```
aws rds create-db-instance ^
    --db-cluster-identifier cluster-name ^
    --db-instance-identifier --db-instance-class db.instance.class ^
    --engine aurora-postgresql
```

Lorsque le processus de migration est terminé, le cluster Aurora PostgreSQL possède une instance de base de données primaire remplie.

Migration des données d’une instance de base de données RDS pour PostgreSQL vers un cluster de bases de données Aurora PostgreSQL à l’aide d’un réplica en lecture Aurora

Vous pouvez utiliser une instance de base de données RDS pour PostgreSQL comme base d’un nouveau cluster de bases de données Aurora PostgreSQL en utilisant un réplica en lecture Aurora pour le processus de migration. L'option Aurora read replica n'est disponible que pour la migration au sein du même compte, Région AWS et elle n'est disponible que si la région propose une version compatible d'Aurora PostgreSQL pour votre instance de base de données RDS pour PostgreSQL. Par compatible, nous entendons que la version d’Aurora PostgreSQL est la même que la version de RDS pour PostgreSQL, ou qu’il s’agit d’une version mineure supérieure dans la même famille de versions majeures.

Par exemple, pour utiliser cette technique pour migrer une instance de base de données RDS pour PostgreSQL 11.14, la Région doit proposer Aurora PostgreSQL version 11.14 ou une version mineure ultérieure dans la famille PostgreSQL version 11.

Rubriques

Présentation de la migration de données à l’aide d’un réplica en lecture Aurora

La migration d’un instantané d’une instance de base de données RDS pour PostgreSQL vers un cluster de bases de données Aurora PostgreSQL est une procédure en plusieurs étapes. Tout d’abord, vous créez un réplica en lecture Aurora de votre instance de base de données RDS pour PostgreSQL source. Cela démarre un processus de réplication à partir de votre instance de base de données RDS pour PostgreSQL vers un cluster de bases de données spécial appelé cluster Replica. Le cluster Replica se compose uniquement d’un réplica en lecture Aurora (instance de lecteur).

> **Note**
>
> La migration peut prendre plusieurs heures par téraoctet de données.

Promotion d’un réplica en lecture Aurora PostgreSQL

Après avoir créé un cluster de bases de données Aurora PostgreSQL, procédez comme suit pour promouvoir le réplica Aurora :

1. Arrêtez toute charge de travail d’écriture de base de données sur l’instance de base de données RDS pour PostgreSQL source.
2. Obtenez le `WAL LSN` actuel à partir de l’instance de base de données RDS pour PostgreSQL source :

```
SELECT pg_current_wal_lsn();
pg_current_wal_lsn
--------------------
0/F0000318
(1 row)
```

3. Sur le cluster de réplica Aurora PostgreSQL, vérifiez que le LSN réutilisé est supérieur au LSN indiqué à l’étape 2 :

```
SELECT pg_last_wal_replay_lsn();
pg_last_wal_replay_lsn
------------------------
0/F0000400
(1 row)
```

   Vous pouvez également utiliser la requête suivante sur l’instance de base de données RDS pour PostgreSQL source :

```
SELECT restart_lsn FROM pg_replication_slots;
```

4. Faites la promotion du cluster de réplica Aurora PostgreSQL.

   Lorsque la réplication s’arrête, le cluster de réplica est promu au statut de cluster de bases de données Aurora PostgreSQL autonome, et le lecteur est promu au statut d’instance d’enregistreur de ce cluster. À ce stade, vous pouvez ajouter des instances au cluster de bases de données Aurora PostgreSQL pour le dimensionner en fonction de votre cas d’utilisation. Si

vous n'avez plus besoin de l'instance de base de données RDS pour PostgreSQL d'origine, vous pouvez la supprimer.

Vous ne pouvez pas créer de réplica en lecture Aurora si votre instance de base de données RDS pour PostgreSQL dispose déjà d'un réplica en lecture Aurora ou entre Régions.

Préparation de la migration de données à l'aide d'un réplica en lecture Aurora

> **Note**
>
> Lorsque vous préparez la migration de données vers Aurora PostgreSQL, il est important d'identifier et de gérer les tables qui ne sont pas journalisées de manière appropriée. Pour plus d'informations, consultez Gestion des tables non journalisées pendant la migration.

Pendant le processus de migration à l'aide du réplica en lecture Aurora, les mises à jour apportées de l'instance de base de données RDS pour PostgreSQL source sont répliquées de façon asynchrone vers le réplica en lecture Aurora du cluster de réplica. Le processus utilise la fonctionnalité de réplication de streaming native de PostgreSQL, qui stocke les segments de journaux write-ahead (WAL, write-ahead logs) sur l'instance source. Avant de commencer ce processus de migration, assurez-vous que votre instance dispose d'une capacité de stockage suffisante en vérifiant les valeurs des métriques répertoriées dans le tableau.

Métrique	Description
`FreeStorageSpace`	Espace de stockage disponible. Unités : octets
`OldestReplicationSlotLag`	Durée du retard pour les données WAL du réplica le plus en retard. Unités : mégaoctets
`RDSToAuroraPostgreSQLReplicaLag`	Durée en secondes du retard d'un cluster de bases de données Aurora PostgreSQL par rapport à l'instance de base de données RDS source.

Métrique	Description
`TransactionLogsDiskUsage`	Espace disque utilisé par les journaux de transactions. Unités : mégaoctets

Pour plus d’informations sur la surveillance de votre instance RDS, consultez Surveillance dans le Guide de l’utilisateur Amazon RDS.

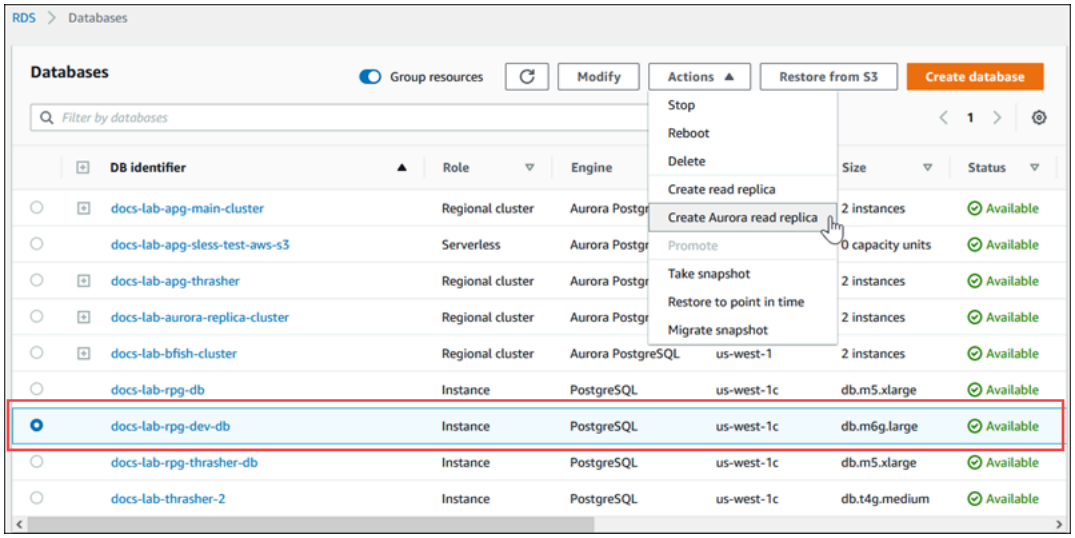
Création d’un réplica en lecture Aurora

Vous pouvez créer une réplique de lecture Aurora pour une instance de base de données RDS pour PostgreSQL en utilisant le ou le. AWS Management Console AWS CLI L'option permettant de créer une réplique de lecture Aurora à l'aide de n' AWS Management Console est disponible que si elle Région AWS propose une version compatible d'Aurora PostgreSQL. En d’autres termes, elle n’est disponible que s’il existe une version Aurora PostgreSQL identique à la version RDS pour PostgreSQL ou une version mineure ultérieure dans la même famille de versions majeures.

Console

Pour créer un réplica en lecture Aurora à partir d’une instance de base de données PostgreSQL source

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez l’instance de base de données RDS pour PostgreSQL que vous souhaitez utiliser comme source pour votre réplica en lecture Aurora. Sous Actions, choisissez Créer un réplica en lecture Aurora. Si ce choix ne s’affiche pas, cela signifie qu’une version compatible Aurora PostgreSQL n’est pas disponible dans la Région.



4. Sur la page Create Aurora read replica settings (Créer des paramètres de réplica en lecture Aurora), configurez les propriétés du cluster de bases de données Aurora PostgreSQL, comme indiqué dans le tableau suivant. Le cluster de bases de données Replica est créé à partir d'un instantané de l'instance de base de données source à l'aide du même nom et mot de passe d'utilisateur principal que la source. Vous ne pouvez donc pas les modifier pour le moment.

Option	Description
Classe d'instance de base de données	Choisissez une classe d'instance de base de données qui répond aux exigences de traitement et de mémoire de l'instance principale du cluster de bases de données. Pour plus d'informations, consultez Classes d'instance de base de données Amazon Aurora.
déploiement multi-AZ	Non disponible pendant la migration
Identifiant d'instance de base de données	Saisissez le nom que vous souhaitez donner à l'instanc e de base de données. Cet identifiant est utilisé dans l'adresse de point de terminaison de l'instance principal e du nouveau cluster de bases de données. L'identifiant d'instance de base de données obéit aux contraintes suivantes : • Il doit contenir entre 1 et 63 caractères alphanumé riques ou traits d'union.

Option	Description
	• Son premier caractère doit être une lettre. • Il ne peut pas se terminer par un trait d'union ou contenir deux traits d'union consécutifs. • Il doit être unique pour toutes les instances de base de données, pour chaque AWS compte, pour chacun Région AWS.
Cloud privé virtuel (VPC)	Choisissez le VPC pour héberger le cluster de bases de données. Choisissez Créer un nouveau VPC pour qu'Amazon RDS vous crée un VPC. Pour plus d'informations, consultez Prérequis des clusters de bases de données.
Groupe de sous-réseaux de base de données	Choisissez le groupe de sous-réseaux de base de données à utiliser pour le cluster de bases de données. Choisissez Créer un groupe de sous-réseaux DB pour qu'Amazon RDS crée un groupe de sous-réseaux DB pour vous. Pour plus d'informations, consultez Prérequis des clusters de bases de données.
Accessible publiquement	Choisissez Oui pour attribuer au cluster de bases de données une adresse IP publique. Sinon, choisisse z Non. Les instances dans votre cluster de bases de données peuvent être un mélange d'instances de base de données publiques et privées. Pour plus d'informa tions sur le masquage des instances de l'accès public, consultez Masquer un cluster de bases de données dans un VPC depuis Internet.
Zone de disponibilité	Déterminez si vous voulez spécifier une zone de disponibilité particulière. Pour plus d'informations sur les zones de disponibilité, consultez Régions et zones de disponibilité.

Option	Description
Groupes de sécurité VPC	Choisissez un ou plusieurs groupes de sécurité VPC pour sécuriser l'accès réseau au cluster de bases de données. Choisissez Create new VPC security group (Créer un groupe de sécurité VPC) pour qu'Amazon RDS crée un groupe de sécurité VPC pour vous. Pour plus d'informations, consultez Prérequis des clusters de bases de données.
Port de la base de données	Spécifiez le port utilisé par les applications et les utilitaires pour accéder à la base de données. Les clusters de bases de données Aurora PostgreSQL utilisent par défaut le port PostgreSQL 5432. Les pare-feu de certaines entreprises bloquent les connexions vers ce port. Si le pare-feu de votre entreprise bloque le port par défaut, choisissez un autre port pour le nouveau cluster DB.
Groupe de paramètres de base de données	Choisissez un groupe de paramètres de base de données pour le cluster de bases de données Aurora PostgreSQL. Aurora possède un groupe de paramètres de base de données par défaut que vous pouvez utiliser, ou bien vous pouvez créer votre propre groupe de paramètres de base de données. Pour plus d'informations sur les groupes de paramètres DB, consultez Groupes de paramètres pour Amazon Aurora.

Option	Description
Groupe de paramètres de cluster de bases de données	Choisissez un groupe de paramètres de cluster de bases de données pour le cluster de bases de données Aurora PostgreSQL. Aurora possède un groupe de paramètres de cluster de bases de données par défaut que vous pouvez utiliser, ou bien vous pouvez créer votre propre groupe de paramètres de cluster de bases de données. Pour plus d'informations sur les groupes de paramètres de cluster DB, consultez Groupes de paramètres pour Amazon Aurora.
Chiffrement	Choisissez Activer le chiffrement pour que votre nouveau cluster de bases de données Aurora soit chiffré au repos. Si vous choisissez Activer le chiffrement, vous devez également choisir une clé KMS comme valeur de AWS KMS key.
Priorité	Choisissez une priorité de basculement pour le cluster DB. Si vous ne choisissez pas de valeur, la valeur par défaut est tier-1. Cette priorité détermine l'ordre dans lequel les réplicas Aurora sont promus lors de la reprise après une défaillance de l'instance principale. Pour plus d'informations, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.
Période de rétention des sauvegardes	Choisissez la durée, comprise entre 135 et 35 jours, pendant laquelle Aurora conservera les copies de sauvegarde de la base de données. Les copies de sauvegarde peuvent être utilisées pour les point-in-time restaurations (PITR) de votre base de données à la seconde près.

Option	Description
Surveillance améliorée	Choisissez Activer la surveillance améliorée pour activer la collecte de métriques en temps réel pour le système d'exploitation sur lequel votre cluster de bases de données s'exécute. Pour plus d'informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.
Monitoring Role (Rôle de surveillance)	Disponible uniquement si vous avez choisi Activer la surveillance améliorée. Le rôle Gestion des identités et des accès AWS (IAM) à utiliser pour la surveillance améliorée. Pour plus d'informations, consultez Configuration et activation de la surveillance améliorée.
Granularité	Disponible uniquement si vous avez choisi Activer la surveillance améliorée. Définissez l'intervalle, en secondes, entre les recueils des métriques pour votre cluster DB.
Mise à niveau automatique de versions mineures	Choisissez Oui pour activer votre cluster de bases de données Aurora PostgreSQL afin qu'il reçoive automatiquement les mises à niveau des versions mineures du moteur de base de données PostgreSQL lorsqu'elles deviennent disponibles. L'option Mise à niveau automatique des versions mineures ne s'applique qu'aux mises à niveau vers les versions mineures du moteur PostgreSQL pour votre cluster de bases de données Aurora PostgreSQL. Elle ne s'applique pas aux correctifs réguliers appliqués pour maintenir la stabilité du système.
Fenêtre de maintenance	Choisissez l'intervalle de temps hebdomadaire (en UTC) pendant lequel la maintenance du système peut se produire.

5. Choisissez Créer un réplica en lecture.

AWS CLI

Pour créer une réplique de lecture Aurora à partir d'une instance de base de données RDS source pour PostgreSQL à l'aide de, vous devez d'abord utiliser AWS CLI la commande create-db-clusterCLI pour créer un cluster de base de données Aurora vide. Une fois que le cluster de bases de données existe, vous pouvez utiliser la commande create-db-instance pour créer l'instance principale de votre cluster de bases de données. L'instance principale est la première instance qui est créée dans un cluster de bases de données Aurora. Dans ce cas, elle est créée initialement comme un réplica en lecture Aurora de votre instance de base de données RDS pour PostgreSQL. À la fin du processus, votre instance de base de données RDS pour PostgreSQL a bien été migrée vers un cluster de bases de données Aurora PostgreSQL.

Vous n'avez pas besoin de spécifier le compte utilisateur principal (généralement, `postgres`), son mot de passe ou le nom de la base de données. La réplique de lecture Aurora les obtient automatiquement à partir de l'instance de base de données RDS pour PostgreSQL source que vous identifiez lorsque vous appelez les commandes. AWS CLI

Vous devez spécifier la version du moteur à utiliser pour le cluster de bases de données Aurora PostgreSQL et l'instance de base de données. La version que vous spécifiez doit correspondre à l'instance de base de données RDS pour PostgreSQL source. Si l'instance de base de données RDS pour PostgreSQL source est chiffrée, vous devez également spécifier le chiffrement pour l'instance principale du cluster de bases de données Aurora PostgreSQL. La migration d'une instance chiffrée vers un cluster de bases de données Aurora non chiffré n'est pas prise en charge.

Les exemples suivants créent un cluster de bases de données Aurora PostgreSQL nommé `my-new-aurora-cluster` qui va utiliser une instance source de base de données RDS non chiffrée. Vous devez d'abord créer le cluster de bases de données Aurora PostgreSQL en appelant la commande CLI `create-db-cluster`. L'exemple montre comment utiliser le paramètre optionnel `--storage-encrypted` pour spécifier que le cluster de bases de données doit être chiffré. Comme la base de données source n'est pas chiffrée, la commande `--kms-key-id` est utilisée pour spécifier la clé à utiliser. Pour obtenir plus d'informations sur les paramètres obligatoires et facultatifs, consultez la liste après l'exemple.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster  \
    --db-cluster-identifier my-new-aurora-cluster \
    --db-subnet-group-name my-db-subnet
    --vpc-security-group-ids sg-11111111
    --engine aurora-postgresql \
```

```
    --engine-version same-as-your-rds-instance-version \
    --replication-source-identifier arn:aws:rds:aws-region:111122223333:db/rpg-source-
db \
    --storage-encrypted \
    --kms-key-id arn:aws:kms:aws-
region:111122223333:key/11111111-2222-3333-444444444444
```

Pour Windows :

```
aws rds create-db-cluster  ^
    --db-cluster-identifier my-new-aurora-cluster ^
    --db-subnet-group-name my-db-subnet ^
    --vpc-security-group-ids sg-11111111 ^
    --engine aurora-postgresql ^
    --engine-version same-as-your-rds-instance-version ^
    --replication-source-identifier arn:aws:rds:aws-region:111122223333:db/rpg-source-
db ^
    --storage-encrypted ^
    --kms-key-id arn:aws:kms:aws-
region:111122223333:key/11111111-2222-3333-444444444444
```

Dans la liste suivante, vous trouverez de plus amples informations sur certaines des options présentées dans l'exemple. Sauf indication contraire, ces paramètres sont obligatoires.

- `--db-cluster-identifier` : vous devez donner un nom à votre nouveau cluster de bases de données Aurora PostgreSQL.
- `--db-subnet-group-name` : créez votre cluster de bases de données Aurora PostgreSQL dans le même sous-réseau de base de données que l'instance de base de données source.
- `--vpc-security-group-ids` : spécifiez le groupe de sécurité pour votre cluster de bases de données Aurora PostgreSQL.
- `--engine-version` : spécifiez la version à utiliser pour le cluster de bases de données Aurora PostgreSQL. Cette version doit être la même que celle utilisée par votre instance de base de données RDS pour PostgreSQL ou doit correspondre à une version mineure supérieure.
- `--replication-source-identifier` : identifiez votre instance de base de données RDS pour PostgreSQL à l'aide de son nom de ressource Amazon (ARN). Pour plus d'informations sur Amazon RDS ARNs, consultez Amazon Relational Database Service (Amazon RDS) dans Références générales AWSIe. de votre cluster de bases de données.
- `--storage-encrypted` : facultatif. Utilisez cette méthode uniquement si nécessaire pour spécifier le chiffrement comme suit :

- Utilisez ce paramètre lorsque l'instance de base de données source possède un stockage chiffré. L'appel à `create-db-cluster` échoue si vous n'utilisez pas ce paramètre avec une instance de base de données source dont le stockage est chiffré. Si vous souhaitez utiliser une clé différente pour le cluster de bases de données Aurora PostgreSQL de celle utilisée par l'instance de base de données source, vous devez également spécifier la clé `--kms-key-id`.
- Utilisez cette méthode si le stockage de l'instance de base de données source n'est pas chiffré mais que vous souhaitez que le cluster de bases de données Aurora PostgreSQL utilise le chiffrement. Dans ce cas, vous devez également identifier la clé de chiffrement à utiliser avec le paramètre `--kms-key-id`.

- `--kms-key-id` : facultatif. Lorsque la méthode est utilisée, vous pouvez spécifier la clé à utiliser pour le chiffrement du stockage (`--storage-encrypted`) en utilisant l'ARN de la clé, son ID, son ARN d'alias ou son nom d'alias. Ce paramètre n'est nécessaire que dans les situations suivantes :
  - Pour choisir une clé différente pour le cluster de bases de données Aurora PostgreSQL que celle utilisée par l'instance de base de données source.
  - Pour créer un cluster chiffré à partir d'une source non chiffrée. Dans ce cas, vous devez spécifier la clé qu'Aurora PostgreSQL doit utiliser pour le chiffrement.

Après avoir créé le cluster de bases de données Aurora PostgreSQL, vous créez ensuite l'instance principale en utilisant la commande CLI `create-db-instance`, comme indiqué ci-dessous :

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance \
    --db-cluster-identifier my-new-aurora-cluster \
    --db-instance-class db.x2g.16xlarge \
    --db-instance-identifier rpg-for-migration \
    --engine aurora-postgresql
```

Pour Windows :

```
aws rds create-db-instance ^
    --db-cluster-identifier my-new-aurora-cluster ^
    --db-instance-class db.x2g.16xlarge ^
    --db-instance-identifier rpg-for-migration ^
    --engine aurora-postgresql
```

Dans la liste suivante, vous trouverez de plus amples informations sur certaines des options présentées dans l'exemple.

- `--db-cluster-identifier` : spécifiez le nom du cluster de bases de données Aurora PostgreSQL que vous avez créé avec la commande `create-db-instance` dans les étapes précédentes.
- `--db-instance-class` : nom de la classe d'instance de base de données à utiliser pour votre instance principale, par exemple, `db.r4.xlarge`, `db.t4g.medium`, `db.x2g.16xlarge` et ainsi de suite. Pour obtenir une liste des classes d'instance de base de données disponibles, consultez Types de classes d'instance de base de données.
- `--db-instance-identifier` : indiquez le nom à donner à votre instance principale.
- `--engine aurora-postgresql` : spécifiez `aurora-postgresql` pour le moteur.

API RDS

Pour créer un réplica en lecture Aurora à partir d'une instance de base de données RDS pour PostgreSQL, utilisez d'abord l'opération de l'API RDS `CreateDBCluster` pour créer un cluster de bases de données Aurora pour le réplica en lecture Aurora créé à partir de votre instance de base de données RDS pour PostgreSQL. Lorsque le cluster de bases de données Aurora PostgreSQL est disponible, vous utilisez la commande `CreateDBInstance` pour créer l'instance principale du cluster de bases de données Aurora.

Vous n'avez pas besoin de spécifier le compte utilisateur principal (généralement, `postgres`), son mot de passe ou le nom de la base de données. Le réplica en lecture Aurora obtient automatiquement ces informations à partir de l'instance de base de données source RDS pour PostgreSQL spécifiée avec `ReplicationSourceIdentifier`.

Vous devez spécifier la version du moteur à utiliser pour le cluster de bases de données Aurora PostgreSQL et l'instance de base de données. La version que vous spécifiez doit correspondre à l'instance de base de données RDS pour PostgreSQL source. Si l'instance de base de données RDS pour PostgreSQL source est chiffrée, vous devez également spécifier le chiffrement pour l'instance principale du cluster de bases de données Aurora PostgreSQL. La migration d'une instance chiffrée vers un cluster de bases de données Aurora non chiffré n'est pas prise en charge.

Pour créer le cluster de bases de données Aurora pour le réplica en lecture Aurora, utilisez l'opération API RDS `CreateDBCluster` avec les paramètres suivants :

- `DBClusterIdentifier` : nom du cluster de bases de données à créer.

- `DBSubnetGroupName` : nom du groupe de sous-réseaux de base de données à associer à ce cluster de bases de données.
- `Engine=aurora-postgresql` : nom du moteur à utiliser.
- `ReplicationSourceIdentifier` : Amazon Resource Name (ARN) de l'instance de base de données PostgreSQL source. Pour plus d'informations sur Amazon RDS ARNs, consultez Amazon Relational Database Service (Amazon RDS) dans le.Référence générale d'Amazon Web Services Si `ReplicationSourceIdentifier` identifie une source chiffrée, Amazon RDS utilise votre clé KMS par défaut, sauf si vous spécifiez une clé différente à l'aide de l'option `KmsKeyId`.
- `VpcSecurityGroupIds`— La liste des groupes de sécurité Amazon EC2 VPC à associer à ce cluster de base de données.
- `StorageEncrypted` : indique que le cluster de bases de données est chiffré. Lorsque vous utilisez ce paramètre sans spécifier également la valeur `ReplicationSourceIdentifier`, Amazon RDS utilise votre clé KMS par défaut.
- `KmsKeyId` : la clé pour un cluster chiffré. Lorsque la méthode est utilisée, vous pouvez spécifier la clé à utiliser pour le chiffrement du stockage en utilisant l'ARN de la clé, son ID, son ARN d'alias ou son nom d'alias.

Pour plus d'informations, consultez `CreateDBCluster` dans la référence de l'API Amazon RDS.

Une fois que le cluster de bases de données Aurora est disponible, vous pouvez alors créer une instance principale pour celui-ci en utilisant l'opération API RDS `CreateDBInstance` avec les paramètres suivants :

- `DBClusterIdentifier` : nom du cluster de bases de données.
- `DBInstanceClass` : nom de la classe d'instance de base de données à utiliser pour votre instance principale.
- `DBInstanceIdentifier` : nom de votre instance principale.
- `Engine=aurora-postgresql` : nom du moteur à utiliser.

Pour plus d'informations, consultez `CreateDBInstance` dans la référence de l'API Amazon RDS.

Promotion d'un réplica en lecture Aurora

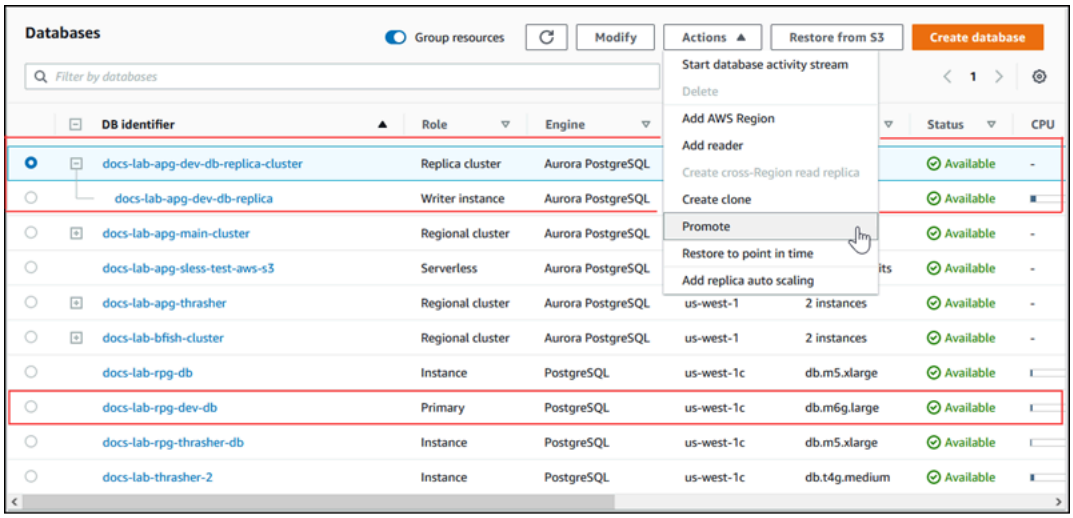
La migration vers Aurora PostgreSQL n'est pas achevée tant que vous n'avez pas fait la promotion du cluster Replica. Par conséquent, ne supprimez pas encore l'instance de base de données RDS pour PostgreSQL source.

Avant de promouvoir le cluster Replica, assurez-vous que l'instance de base de données RDS pour PostgreSQL ne dispose pas de transactions en cours de traitement ou d'autres activités d'écriture dans la base de données. Lorsque le décalage de réplica sur le réplica de lecture Aurora atteint zéro (0), vous pouvez promouvoir le cluster Replica. Pour plus d'informations sur la surveillance du décalage de réplica, consultez Surveillance de la réplication Aurora PostgreSQL et Métriques de niveau instance pour Amazon Aurora.

Console

Pour promouvoir un réplica en lecture Aurora en cluster de bases de données Aurora

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster Replica.



4. Pour Actions, choisissez Promote (Promouvoir). Cette opération peut prendre quelques minutes et entraîner une durée d'indisponibilité.

Une fois le processus terminé, le cluster de réplica Aurora est un cluster de bases de données Aurora PostgreSQL régional, avec une instance d'enregistreur contenant les données de l'instance de base de données RDS pour PostgreSQL.

AWS CLI

Pour transformer une réplique de lecture Aurora en cluster de base de données autonome, utilisez la promote-read-replica-db-cluster AWS CLI commande.

Example

Pour Linux, macOS ou Unix :

```
aws rds promote-read-replica-db-cluster \
    --db-cluster-identifier myreadreplicacluster
```

Pour Windows :

```
aws rds promote-read-replica-db-cluster ^
    --db-cluster-identifier myreadreplicacluster
```

API RDS

Pour transformer une réplique de lecture Aurora en cluster de base de données autonome, utilisez l'opération d'API RDS. PromoteReadReplicaDBCluster

Après avoir promu le cluster Replica, vous pouvez confirmer que la promotion est terminée en vérifiant le journal des événements, comme suit.

Pour confirmer que le cluster Replica Aurora a été promu

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, sélectionnez Events.
3. Dans la page Events (Événements), recherchez le nom de votre cluster dans la liste Source (Source). Chaque événement comporte une source, un type, une heure et un message. Vous pouvez voir tous les événements survenus dans votre Région AWS pour votre compte. Une promotion réussie génère le message suivant.

   ```
   Promoted Read Replica cluster to a stand-alone database cluster.
   ```

Une fois la promotion terminée, l'instance de base de données RDS pour PostgreSQL source et le cluster de bases de données Aurora PostgreSQL sont dissociés. Vous pouvez diriger vos applications clientes vers le point de terminaison du réplica en lecture Aurora. Pour plus d'informations sur les points de terminaison Aurora, consultez Connexions de point de terminaison Amazon Aurora. À ce stade, vous pouvez supprimer en toute sécurité l'instance de base de données.

Optimisation des performances des requêtes dans Aurora PostgreSQL

L'optimisation des performances des requêtes est cruciale, car elle permet aux bases de données de fonctionner plus rapidement et plus efficacement tout en utilisant moins de ressources, ce qui se traduit par une meilleure expérience utilisateur et des coûts d'exploitation réduits. Amazon Aurora PostgreSQL fournit plusieurs fonctionnalités permettant d'optimiser les performances des requêtes pour les charges de travail PostgreSQL.

Rubriques

Amélioration des performances des requêtes pour Aurora PostgreSQL avec Aurora Optimized Reads

Vous pouvez accélérer le traitement des requêtes pour Aurora PostgreSQL avec Aurora Optimized Reads. Une instance de base de données Aurora PostgreSQL qui utilise Aurora Optimized Reads permet d'améliorer jusqu'à 8 fois le temps de latence des requêtes et de réaliser des économies pouvant atteindre 30 % pour les applications comportant des jeux de données volumineux, qui dépassent la capacité de mémoire d'une instance de base de données.

Rubriques

Présentation d’Aurora Optimized Reads dans PostgreSQL

Aurora Optimized Reads est disponible par défaut lorsque vous créez un cluster de base de données qui utilise des instances R6gd, R8gd et Intel R6id basées sur Graviton avec un stockage non volatile memory express (). NVMe Il est disponible à partir des versions PostgreSQL suivantes :

- Versions 14.12 et supérieures, versions 15.7 et supérieures, versions 16.3 et supérieures, versions 17.4 et supérieures pour les instances R8gd
- Versions 14.9 et supérieures, versions 15.4 et supérieures, 16.1 et toutes les versions supérieures pour les instances R6gd et R6id

Aurora Optimized Reads prend en charge deux fonctionnalités : le cache à plusieurs niveaux et les objets temporaires.

Cache à plusieurs niveaux Optimized Reads : grâce au cache à plusieurs niveaux, vous pouvez étendre la capacité de mise en cache de votre instance de base de données jusqu’à 5 fois la mémoire de l’instance. Cela permet de maintenir le cache à jour automatiquement de manière à ce qu’il contienne les données les plus récentes et les plus homogènes sur le plan transactionnel, et de libérer ainsi les applications de la charge de gestion de la mise à jour des données des solutions de mise en cache externes basées sur des ensembles de résultats. Il réduit jusqu’à 8 fois le temps de latence des requêtes qui récupéraient auparavant des données du stockage Aurora.

Dans Aurora, la valeur pour `shared_buffers` dans le groupe de paramètres par défaut est généralement définie sur environ 75 % de la mémoire disponible. Toutefois, pour les types d'instances r8gd, r6gd et r6id, Aurora réduit l'`shared_buffers`espace de 4,5 % pour héberger les métadonnées du cache de lectures optimisées.

Objets temporaires optimisés compatibles Reads : à l'aide d'objets temporaires, vous pouvez accélérer le traitement des requêtes en plaçant les fichiers temporaires générés par PostgreSQL sur le stockage local. NVMe Cela réduit le trafic à destination d’Elastic Block Storage (EBS) sur le réseau. La latence et le débit sont jusqu’à deux fois supérieurs pour les requêtes avancées qui trient, joignent ou fusionnent les gros volumes de données qui ne rentrent pas dans la capacité de mémoire disponible sur une instance de base de données.

Sur un cluster optimisé pour les E/S Aurora, Optimized Reads utilise à la fois le cache hiérarchisé et les objets temporaires stockés. NVMe Grâce au cache à plusieurs niveaux Optimized Reads, Aurora alloue deux fois la mémoire de l’instance aux objets temporaires, environ 10 % de l’espace de

stockage aux opérations internes et le reste du stockage sous forme de cache à plusieurs niveaux. Sur un cluster Aurora standard, Optimized Reads utilise uniquement des objets temporaires.

Les clusters optimisés pour les E/S Aurora vous permettent de redimensionner l'espace alloué aux objets temporaires compatibles avec Optimized Reads à l'aide du paramètre dynamique `aurora_temp_space_size` au niveau de l'instance. Cette fonctionnalité de redimensionnement est disponible à partir des versions PostgreSQL suivantes :

- Version 16.8 et toutes les versions ultérieures
- 15.12 et versions 15 ultérieures
- 14.17 et versions 14 ultérieures

Avec ce paramètre, vous pouvez redimensionner la capacité pour qu'elle atteigne de 2 à 6 fois la mémoire de l'instance sans avoir à redémarrer le moteur de base de données. Lorsque vous élargissez l'espace des objets temporaires, cette modification prend effet immédiatement, quelles que soient les charges de travail simultanées. Toutefois, lorsque vous réduisez l'espace, cette modification ne prend effet qu'une fois qu'il y a suffisamment d'espace inutilisé dans les objets temporaires pour répondre à la nouvelle demande de taille. Une fois que vous avez redimensionné des objets temporaires compatibles avec Optimized Reads, le cache à plusieurs niveaux s'ajuste automatiquement pour utiliser l'espace disponible.

Engine	Configuration du stockage en cluster	Objets temporaires Optimized Reads	Cache à plusieurs niveaux Optimized Reads	Versions prises en charge
Aurora PostgreSQL-Compatible Edition	Standard	Oui	Non	• Versions 14.9 et supérieures, versions 15.4 et supérieures, 16.1 et toutes les versions supérieures pour les instances r6gd et r6id
	Optimisé pour les E/S	Oui	Oui	

Engine	Configuration du stockage en cluster	Objets temporaires Optimized Reads	Cache à plusieurs niveaux Optimized Reads	Versions prises en charge
				• Versions 14.12 et supérieures, versions 15.7 et supérieures, versions 16.3 et supérieures, versions 17.4 et supérieures pour les instances r8gd

> **Note**
>
> Un basculement entre des clusters optimisés pour l'IO et des clusters standard sur une classe d'instance de base de données NVMe basée entraîne le redémarrage immédiat du moteur de base de données.

Dans Aurora PostgreSQL, utilisez le paramètre `temp_tablespaces` pour configurer l'espace de la table sur lequel les objets temporaires seront stockés.

Pour vérifier si les objets temporaires sont configurés, utilisez la commande suivante :

```
postgres=> show temp_tablespaces;
temp_tablespaces
---------------------
aurora_temp_tablespace
(1 row)
```

`aurora_temp_tablespace`Il s'agit d'un tablespace configuré par Aurora qui pointe vers le stockage NVMe local. Vous ne pouvez pas modifier ce paramètre ou revenir au stockage Amazon EBS.

Pour vérifier si le cache Optimized Reads est activé, utilisez la commande suivante :

```
postgres=> show shared_preload_libraries;
                shared_preload_libraries
------------------------------------------------------------
rdsutils,pg_stat_statements,aurora_optimized_reads_cache
```

Utilisation d’Aurora Optimized Reads

Lorsque vous provisionnez une instance de base de données Aurora PostgreSQL avec l'une des instances de base de données basées, NVMe l'instance de base de données utilise automatiquement les lectures optimisées Aurora.

Pour activer Aurora Optimized Reads, effectuez l’une des actions suivantes :

- Créez un cluster de base de données Aurora PostgreSQL à l'aide de l'une des classes d'instances de base de données basées sur NVMe la base de données. Pour de plus amples informations, veuillez consulter Création d’un cluster de bases de données Amazon Aurora.
- Modifiez un cluster de base de données Aurora PostgreSQL existant pour utiliser l'une des classes d'instances de base de données basées sur NVMe la base de données. Pour de plus amples informations, veuillez consulter Modification d’un cluster de bases de données Amazon Aurora.

Aurora Optimized Reads est disponible partout Régions AWS où une ou plusieurs classes d'instances de base de données avec stockage NVMe SSD local sont prises en charge. Pour de plus amples informations, veuillez consulter Classes d’instance de base de données Amazon Aurora.

Pour revenir à une instance Aurora en lecture non optimisée, modifiez la classe d'instance de base de données de votre instance Aurora en une classe d'instance similaire sans stockage NVMe éphémère pour les charges de travail de votre base de données. Par exemple, si la classe d’instance de base de données actuelle est db.r6gd.4xlarge, choisissez db.r6g.4xlarge pour revenir en arrière. Pour plus d’informations, consultez Modification d’une instance de base de données Amazon RDS.

Cas d’utilisation d’Aurora Optimized Reads

Cache à plusieurs niveaux Optimized Reads

Voici quelques cas d’utilisation du cache à plusieurs niveaux Optimized Reads :

- Applications à l'échelle d'Internet telles que le traitement des paiements, la facturation, le commerce électronique avec des performances strictes SLAs.

- Des tableaux de bord de reporting en temps réel qui exécutent des centaines de requêtes ponctuelles à des fins de metrics/data collecte.
- Applications d'IA générative avec l'extension pgvector pour rechercher des voisins exacts ou les voisins les plus proches parmi des millions de vecteurs intégrés.

Objets temporaires Optimized Reads

Voici quelques cas d'utilisation des objets temporaires Optimized Reads :

- Requêtes analytiques qui incluent des expressions de table communes (CTEs), des tables dérivées et des opérations de regroupement.
- Réplicas en lecture qui gèrent les requêtes non optimisées pour une application.
- Requêtes de création de rapports dynamiques ou à la demande avec des opérations complexes telles que GROUP BY et ORDER BY qui ne peuvent pas toujours utiliser les index appropriés.
- Opérations `CREATE INDEX` ou `REINDEX` pour le tri.
- Autres charges de travail utilisant des tables temporaires internes.

Surveillance des instances de base de données qui utilisent Aurora Optimized Reads

Vous pouvez surveiller vos requêtes qui utilisent le cache à plusieurs niveaux Optimized Reads à l'aide de la commande EXPLAIN comme illustré dans l'exemple suivant :

```
Postgres=> EXPLAIN (ANALYZE, BUFFERS) SELECT c FROM sbtest15 WHERE id=100000000


QUERY PLAN
------------------------------------------------------------------------------------
 Index Scan using sbtest15_pkey on sbtest15  (cost=0.57..8.59 rows=1 width=121) (actual
 time=0.287..0.288 rows=1 loops=1)
   Index Cond: (id = 100000000)
   Buffers: shared hit=3 read=2 aurora_orcache_hit=2
   I/O Timings: shared/local read=0.264
 Planning:
   Buffers: shared hit=33 read=6 aurora_orcache_hit=6
   I/O Timings: shared/local read=0.607
 Planning Time: 0.929 ms
 Execution Time: 0.303 ms
(9 rows)
Time: 2.028 ms
```

> **Note**
>
> Les champs `aurora_orcache_hit` et `aurora_storage_read` de la section `Buffers` du plan d'explication ne sont affichés que lorsqu'Optimized Reads est activé et que le résultat est supérieur à zéro. Le champ de lecture est le total des champs `aurora_orcache_hit` et `aurora_storage_read`.

Vous pouvez surveiller les instances de base de données qui utilisent les lectures optimisées Aurora à l'aide CloudWatch des métriques suivantes :

- `AuroraOptimizedReadsCacheHitRatio`
- `FreeEphemeralStorage`
- `ReadIOPSEphemeralStorage`
- `ReadLatencyEphemeralStorage`
- `ReadThroughputEphemeralStorage`
- `WriteIOPSEphemeralStorage`
- `WriteLatencyEphemeralStorage`
- `WriteThroughputEphemeralStorage`

Ces métriques fournissent des données sur le stockage disponible dans le stockage d'instances, les IOPS et le débit. Pour plus d'informations sur ces métriques, consultez Métriques de niveau instance pour Amazon Aurora.

Vous pouvez également utiliser l'`pg_proctab`extension pour surveiller le NVMe stockage.

```
postgres=>select * from pg_diskusage();

major | minor |       devname       | reads_completed | reads_merged | sectors_read |
 readtime | writes_completed | writes_merged | sectors_written | writetime | current_io
 | iotime  | totaliotime
------+-------+---------------------+-----------------+--------------+--------------
+----------+------------------+---------------+-----------------+-----------
+------------+---------+-------------
```

```
      |       | rdstemp              |           23264 |            0 |       191450 |
  11670 |          1750892 |             0 |        24540576 |    819350 |          0 |
 3847580 |      831020
      |       | rdsephemeralstorage |           23271 |            0 |       193098 |
   2620 |           114961 |             0 |        13845120 |    130770 |          0 |
  215010 |      133410
(2 rows)
```

Bonnes pratiques pour Aurora Optimized Reads

Utilisez les bonnes pratiques suivantes pour Aurora Optimized Reads :

- Surveillez l'espace de stockage disponible sur le magasin d'instances à l'aide de la CloudWatch métrique`FreeEphemeralStorage`. Si le stockage d'instances atteint sa limite en raison de la charge de travail de l'instance de base de données, ajustez la simultanéité et les requêtes qui utilisent des objets temporaires de manière intensive, ou modifiez le stockage d'instances pour qu'il utilise une classe d'instance de base de données plus grande.
- Surveillez la CloudWatch métrique du taux de réussite du cache Optimized Reads. Des opérations telles que VACUUM modifient très rapidement un grand nombre de blocs. Cela peut entraîner une baisse temporaire du taux d'accès. L'extension `pg_prewarm` peut être utilisée pour charger des données dans le cache de tampon, ce qui permet à Aurora d'écrire de manière proactive certains de ces blocs dans le cache Optimized Reads.
- Vous pouvez activer la gestion du cache de cluster pour réchauffer le cache d tampon et le cache à plusieurs niveaux sur un lecteur de niveau 0, qui sera utilisé comme cible de basculement. Lorsque la gestion du cache de cluster est activée, le cache de tampon est scanné périodiquement pour écrire les pages susceptibles d'être expulsées dans le cache à plusieurs niveaux. Pour plus d'informations sur la gestion du cache de cluster, consultez Récupération rapide après basculement avec la gestion des caches de clusters pour Aurora PostgreSQL.

Optimisation des sous-requêtes corrélées dans Aurora PostgreSQL

Une sous-requête corrélée référence les colonnes de la table à partir de la requête externe. Elle est évaluée une fois pour chaque ligne renvoyée par la requête externe. Dans l'exemple suivant, la sous-requête référence une colonne de la table ot. Cette table n'est pas incluse dans la clause FROM de la sous-requête, mais elle est référencée dans la clause FROM de la requête externe. Si la table ot contient 1 million de lignes, la sous-requête doit être évaluée 1 million de fois.

```
SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT AVG(it.b) FROM it WHERE it.a = ot.a);
```

> **Note**
>
> - La transformation des sous-requêtes et le cache de sous-requêtes sont disponibles dans Aurora PostgreSQL à partir de la version 16.8, tandis que Babelfish pour Aurora PostgreSQL prend en charge ces fonctionnalités à partir de la version 4.2.0.
> - À partir des versions 4.6.0 et 5.2.0 de Babelfish pour Aurora PostgreSQL, les paramètres suivants contrôlent ces fonctionnalités :
>   - babelfishpg_tsql.apg_enable_correlated_scalar_transform
>   - babelfishpg_tsql.apg_enable_subquery_cache
>
>   Par défaut, ces deux paramètres sont activés.

Amélioration des performances des requêtes Aurora PostgreSQL à l'aide de la transformation des sous-requêtes

Aurora PostgreSQL peut accélérer les sous-requêtes corrélées en les transformant en jointures externes équivalentes. Cette optimisation s'applique aux deux types de sous-requêtes corrélées suivants :

- Sous-requêtes renvoyant une valeur agrégée unique et apparaissant dans la liste SELECT.

  ```
  SELECT ot.a, ot.b, (SELECT AVG(it.b) FROM it WHERE it.a = ot.a) FROM ot;
  ```

- Sous-requêtes renvoyant une valeur agrégée unique et apparaissant dans une clause WHERE.

  ```
  SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT AVG(it.b) FROM it WHERE it.a = ot.a);
  ```

Activation de la transformation dans la sous-requête

Pour activer la transformation des sous-requêtes corrélées en jointures externes équivalentes, définissez le paramètre `apg_enable_correlated_scalar_transform` sur `ON`. La valeur par défaut de ce paramètre est `OFF`.

Vous pouvez modifier les paramètres du cluster ou de l'instance pour définir les paramètres. Pour en savoir plus, consultez Groupes de paramètres pour Amazon Aurora.

Vous pouvez également configurer le paramètre uniquement pour la session en cours à l'aide de la commande suivante :

```
SET apg_enable_correlated_scalar_transform TO ON;
```

Vérification de la transformation

Utilisez la commande EXPLAIN pour vérifier si la sous-requête corrélée a été transformée en jointure externe dans le plan de requête.

Lorsque la transformation est activée, la partie de sous-requête corrélée applicable est transformée en jointure externe. Exemples :

```
postgres=> CREATE TABLE ot (a INT, b INT);
CREATE TABLE
postgres=> CREATE TABLE it (a INT, b INT);
CREATE TABLE

postgres=> SET apg_enable_correlated_scalar_transform TO ON;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT
 AVG(it.b) FROM it WHERE it.a = ot.a);

                        QUERY PLAN
-------------------------------------------------------------
 Hash Join
   Hash Cond: (ot.a = apg_scalar_subquery.scalar_output)
   Join Filter: ((ot.b)::numeric < apg_scalar_subquery.avg)
   ->  Seq Scan on ot
   ->  Hash
         ->  Subquery Scan on apg_scalar_subquery
               ->  HashAggregate
                     Group Key: it.a
                     ->  Seq Scan on it
```

La même requête n'est pas transformée lorsque le paramètre GUC est activé (OFF). Le plan n'aura pas de jointure externe mais un sous-plan à la place.

```
postgres=> SET apg_enable_correlated_scalar_transform TO OFF;
```

```
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT
 AVG(it.b) FROM it WHERE it.a = ot.a);
                QUERY PLAN
---------------------------------------
 Seq Scan on ot
   Filter: ((b)::numeric < (SubPlan 1))
   SubPlan 1
     ->  Aggregate
           ->  Seq Scan on it
                 Filter: (a = ot.a)
```

Limitations

- La sous-requête doit figurer dans la liste SELECT ou dans l'une des conditions de la clause WHERE. Dans le cas contraire, elle ne sera pas transformée.
- La sous-requête doit renvoyer une fonction d'agrégation. Les fonctions d'agrégation définies par l'utilisateur ne sont pas prises en charge pour la transformation.
- Une sous-requête dont l'expression de retour n'est pas une fonction d'agrégation simple ne sera pas transformée.
- La condition corrélée dans les clauses WHERE de la sous-requête doit être une référence de colonne simple. Dans le cas contraire, elle ne sera pas transformée.
- La condition corrélée dans les clauses WHERE de la sous-requête doit être un simple prédicat d'égalité.
- La sous-requête ne peut pas contenir de clause HAVING ou GROUP BY.
- La clause WHERE de la sous-requête peut contenir un ou plusieurs prédicats combinés à AND.

> **Note**
>
> L'impact de la transformation sur les performances varie en fonction de votre schéma, de vos données et de votre charge de travail. L'exécution de sous-requêtes corrélées avec la transformation peut améliorer considérablement les performances à mesure que le nombre de lignes générées par la requête externe augmente. Nous vous recommandons vivement de tester cette fonctionnalité dans un environnement hors production avec votre schéma, vos données et votre charge de travail réels avant de l'activer dans un environnement de production.

Utilisation du cache de sous-requêtes pour améliorer les performances des requêtes Aurora PostgreSQL

Aurora PostgreSQL prend en charge un cache de sous-requêtes pour stocker les résultats des sous-requêtes corrélées. Cette fonctionnalité ignore les exécutions répétées de sous-requêtes corrélées lorsque leurs résultats se trouvent déjà dans le cache.

Comprendre le cache de sous-requêtes

Le nœud Memoize de PostgreSQL est l'élément clé du cache de sous-requêtes. Le nœud Memoize gère une table de hachage dans le cache local pour mapper les valeurs des paramètres d'entrée aux lignes de résultats des requêtes. La limite de mémoire de la table de hachage est le produit de work_mem et hash_mem_multiplier. Pour en savoir plus, consultez Consommation des ressources.

Lors de l'exécution des requêtes, le cache de sous-requêtes utilise le taux d'accès au cache pour estimer si le cache améliore les performances des requêtes et pour décider, au moment de l'exécution d'une requête, s'il convient de continuer à utiliser le cache. Le taux d'accès au cache est le rapport entre le nombre d'accès au cache et le nombre total de requêtes. Par exemple, si une sous-requête corrélée doit être exécutée 100 fois et que 70 de ces résultats d'exécution peuvent être extraits du cache, le taux d'accès au cache est de 0,7.

Pour chaque perte de mémoire cache par apg_subquery_cache_interval, l'avantage du cache de sous-requêtes est évalué en vérifiant si le taux d'accès au cache est supérieur à apg_subquery_cache_hit_rate_threshold. Si ce n'est pas le cas, le cache sera supprimé de la mémoire, et l'exécution de la requête reviendra à la réexécution initiale de la sous-requête non mise en cache.

Paramètres qui contrôlent le comportement du cache de sous-requêtes

Le tableau suivant répertorie les paramètres qui contrôlent le comportement du cache de sous-requêtes.

Paramètre	Description	Par défaut	Autorisé
apg_enable_subquery_cache	Permet d'utiliser le cache pour les sous-requêtes scalaires corrélées.	OFF	ON, OFF

Paramètre	Description	Par défaut	Autorisé
apg_subquery_cache_check_interval	Définit la fréquence, en nombre de pertes de mémoire cache, pour évaluer le taux d'accès au cache des sous-requêtes.	500	0–2147483647
apg_subquery_cache_hit_rate_threshold	Définit le seuil du taux d'accès au cache de sous-requêtes.	0.3	0.0-1.0

> **Note**
>
> - Des valeurs plus élevées pour `apg_subquery_cache_check_interval` peuvent améliorer la précision de l'estimation des avantages du cache basée sur le taux d'accès au cache, mais augmentent la surcharge du cache, car le taux d'accès au cache ne sera pas évalué tant que la table de cache ne contiendra pas le nombre de lignes défini pour `apg_subquery_cache_check_interval`.
> - Des valeurs plus élevées pour `apg_subquery_cache_hit_rate_threshold` indiquent un biais en faveur de l'abandon du cache de sous-requêtes et du retour à la réexécution des sous-requêtes d'origine, non mises en cache.

Vous pouvez modifier les paramètres du cluster ou de l'instance pour définir les paramètres. Pour en savoir plus, consultez Groupes de paramètres pour Amazon Aurora.

Vous pouvez également configurer le paramètre uniquement pour la session en cours à l'aide de la commande suivante :

```
SET apg_enable_subquery_cache TO ON;
```

Activation du cache de sous-requêtes dans Aurora PostgreSQL

Lorsque le cache de sous-requêtes est activé, Aurora PostgreSQL l'applique pour enregistrer les résultats des sous-requêtes. Dans ce cas, le plan de requête a un nœud Memoize sous SubPlan.

Par exemple, la séquence de commandes suivante montre le plan d'exécution estimé d'une sous-requête simple corrélée sans cache de sous-requêtes.

```
postgres=> SET apg_enable_subquery_cache TO OFF;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT it.b
 FROM it WHERE it.a = ot.a);

             QUERY PLAN
-----------------------------------
 Seq Scan on ot
   Filter: (b < (SubPlan 1))
   SubPlan 1
     ->  Seq Scan on it
           Filter: (a = ot.a)
```

Après l'activation d'`apg_enable_subquery_cache`, le plan de requête contient un nœud Memoize sous le nœud SubPlan, indiquant que la sous-requête prévoit d'utiliser le cache.

```
postgres=> SET apg_enable_subquery_cache TO ON;
SET
postgres=> EXPLAIN (COSTS FALSE) SELECT ot.a, ot.b FROM ot WHERE ot.b < (SELECT it.b
 FROM it WHERE it.a = ot.a);

             QUERY PLAN
-----------------------------------
 Seq Scan on ot
   Filter: (b < (SubPlan 1))
   SubPlan 1
     ->  Memoize
           Cache Key: ot.a
           Cache Mode: binary
           ->  Seq Scan on it
                 Filter: (a = ot.a)
```

Le plan d'exécution des requêtes proprement dit contient plus de détails sur le cache de sous-requêtes, y compris les accès au cache et les pertes de mémoire cache. La sortie suivante montre le plan d'exécution réel de l'exemple de requête ci-dessus après l'insertion de certaines valeurs dans les tables.

```
postgres=> EXPLAIN (COSTS FALSE, TIMING FALSE, ANALYZE TRUE) SELECT ot.a, ot.b FROM ot
 WHERE ot.b < (SELECT it.b FROM it WHERE it.a = ot.a);
```

```
                QUERY PLAN
-------------------------------------------------------------------------------
 Seq Scan on ot (actual rows=2 loops=1)
   Filter: (b < (SubPlan 1))
   Rows Removed by Filter: 8
   SubPlan 1
     ->  Memoize (actual rows=0 loops=10)
           Cache Key: ot.a
           Cache Mode: binary
           Hits: 4  Misses: 6  Evictions: 0  Overflows: 0  Memory Usage: 1kB
           ->  Seq Scan on it (actual rows=0 loops=6)
                 Filter: (a = ot.a)
                 Rows Removed by Filter: 4
```

Le nombre total d'accès au cache est de 4, et le nombre total de pertes de mémoire cache est de 6. Si le nombre total d'accès au cache et de pertes de cache mémoire est inférieur au nombre de boucles dans le nœud Memoize, cela signifie que l'évaluation du taux d'accès au cache n'a pas été réussie et que le cache a été nettoyé et abandonné à un moment donné. L'exécution de la sous-requête est ensuite renvoyée à la réexécution initiale non mise en cache.

Limitations

Le cache de sous-requêtes ne prend pas en charge certains modèles de sous-requêtes corrélées. Ces types de requêtes sont exécutés sans cache, même si le cache de sous-requêtes est activé :

- Sous-requêtes corrélées IN/EXISTS/ANY/ALL
- Sous-requêtes corrélées contenant des fonctions non déterministes.
- Sous-requêtes corrélées qui font référence à des colonnes de table externes avec des types de données qui ne prennent pas en charge les opérations de hachage ou d'égalité.

Amélioration des performances des requêtes à l'aide d'une jointure adaptative

Présentation

La jointure adaptative est une fonctionnalité de présentation d'Aurora PostgreSQL 17.4 qui permet d'améliorer les performances des requêtes. Cette fonctionnalité est désactivée par défaut, mais vous pouvez l'activer à l'aide des paramètres de configuration utilisateur globale (GUC). Comme il s'agit d'une fonctionnalité de présentation, les valeurs des paramètres par défaut peuvent changer.

Lorsqu'elle est activée, la jointure adaptative permet d'optimiser les performances des requêtes en passant dynamiquement d'une jointure de boucle imbriquée à une jointure de hachage lors de l'exécution. Ce changement se produit lorsque l'optimiseur PostgreSQL a mal choisi une jointure de boucle imbriquée en raison d'estimations de cardinalité inexactes.

Configuration de jointure adaptative

Vous pouvez contrôler la jointure adaptative à l'aide de trois paramètres GUC suivants :

Paramètres de configuration de jointure adaptative

Paramètre GUC	Description	Options par défaut et de configuration
apg_adaptive_join_ crossover_multiplicateur	Ce multiplicateur fonctionne avec le point de croisement de ligne pour déterminer quand passer d'une boucle imbriquée à une jointure de hachage. Le point de croisement de ligne est l'endroit où l'optimiseur SQL estime que les opérations de boucle imbriquée et de jointure de hachage ont le même coût. Une valeur de multiplicateur plus élevée réduit la probabilité que la jointure adaptative passe à une jointure de hachage.	Contrôle si la jointure adaptative est activée • Valeur par défaut : -1 (désactivé) • Plage valide : -1 à DBL_MAX • Pour activer : définissez sur >= 1
apg_adaptive_join_ cost_threshold	Ce paramètre définit un seuil de coût minimal pour les requêtes. La jointure adaptative se désactive automatiq uement pour les requêtes inférieures à ce seuil. Cela permet d'éviter les surcharges de performances dans les requêtes simples où le coût de planification d'une jointure adaptativ e pourrait dépasser les avantages du passage d'une boucle imbriquée à une jointure de hachage.	Définit le seuil de coût minimum pour la requête • Valeur par défaut : 100 • Plage valide : 0 à DBL_MAX

Paramètre GUC	Description	Options par défaut et de configuration
apg_enable_parameterized_adaptive_join	Ce paramètre, lorsqu'il est activé, étend la fonctionnalité de jointure adaptative aux jointures par boucles imbriquées paramétrées. Par défaut, la jointure adaptative fonctionne uniquement avec les jointures par boucle imbriquées non paramétrées, car elles sont plus susceptibles de bénéficier du passage à la jointure de hachage. Les jointures par boucle imbriquée paramétrées sont généralement plus performantes, ce qui rend le passage à la jointure de hachage moins critique.	Contrôle le comportement de jointure adaptative pour les jointures de boucle imbriquée • Valeur par défaut : false • Valeurs valides : true/false • Lorsque la valeur est false : fonctionne uniquement avec des jointures de boucles imbriquées non paramétrées • Lorsque la valeur est true : fonctionne avec des jointures de boucles imbriquées paramétrées et non paramétrées **Note** Il est nécessaire que le `apg_adaptive_join_crossover_multiplier` soit activé au préalable

Utilisation de tables non journalisées dans Aurora PostgreSQL

Amazon Aurora PostgreSQL prend en charge les tables non journalisées qui sont sécurisées et préservent l'intégrité des données même en cas de défaillance ou de basculement d'une instance d'enregistreur. Dans PostgreSQL standard, les tables non journalisées contournent le journal WAL (Write Ahead Log) pendant les opérations d'écriture, ce qui se traduit par des vitesses d'écriture plus rapides. Cependant, cela se fait au détriment de la durabilité, car les tables non journalisées ne sont pas sécurisées et peuvent perdre des données en cas de panne du système ou d'arrêt incorrect. Ces tables non journalisées sont automatiquement tronquées en cas de panne ou d'arrêt incorrect. Leur contenu et leurs index ne sont pas non plus répliqués sur les serveurs de secours.

En revanche, Aurora PostgreSQL gère différemment les tables non journalisées en raison de son architecture de stockage distribuée. Cela est dû au fait que le système de stockage d'Aurora ne repose pas sur le journal WAL PostgreSQL traditionnel pour sa durabilité. Cependant, les avantages généralement associés aux tables non journalisées dans PostgreSQL standard en termes de performances ne sont peut-être pas aussi significatifs dans Aurora. Cela est dû à l'architecture de stockage distribué d'Aurora, qui peut entraîner une surcharge supplémentaire par rapport au stockage local utilisé dans PostgreSQL standard.

Lors de l'utilisation de tables non journalisées dans Aurora PostgreSQL, tenez compte des éléments suivants :

- Vous ne pouvez accéder aux tables non journalisées qu'à partir du nœud d'enregistreur du cluster de bases de données Aurora.
- Les nœuds de lecteur ne peuvent accéder aux tables non journalisées que lorsqu'ils sont promus au statut d'enregistreur.
- Lorsque vous essayez d'accéder à des tables non journalisées à partir d'un nœud de lecteur, l'erreur suivante s'affiche :

  ```
  cannot access temporary or unlogged relations during recovery.
  ```

Création de tables non journalisées

Pour créer une table non journalisée dans Aurora PostgreSQL, ajoutez le mot clé UNLOGGED dans l'instruction CREATE TABLE :

```
CREATE UNLOGGED TABLE staging_sales_data (
```

```
    transaction_id bigint,
    customer_id bigint,
    product_id bigint,
    transaction_date date,
    amount NUMERIC
);
```

Gestion des tables non journalisées pendant la migration

Lorsque vous préparez la migration de données vers Aurora PostgreSQL, il est important d'identifier et de gérer les tables qui ne sont pas journalisées de manière appropriée. Les tables non journalisées ne sont pas journalisées par WAL et sont exclues du flux de réplication. Elles ne sont donc pas répliquées sur un réplica en lecture Aurora.

Convertissez les tables non journalisées en tables journalisées ou supprimez-les si elles ne sont pas utilisées avant de créer un réplica en lecture Aurora.

Pour vérifier la présence de tables non journalisées dans chaque base de données de l'instance, utilisez la commande suivante :

```
SELECT oid, relfilenode, relname, relpersistence, relkind
FROM pg_class
WHERE relpersistence ='u';
```

Pour reconvertir une table non journalisée en table journalisée, utilisez la commande suivante :

```
ALTER TABLE table_name SET LOGGED;
```

Cette opération réécrit l'intégralité de la table et y applique un verrou exclusif jusqu'à ce qu'elle soit terminée. Pour les grandes tables, cela peut entraîner des durées d'indisponibilité importantes.

Conversion de tables non journalisées en tables journalisées

Lorsque vous devez reconvertir une table non journalisée en table journalisée, vous pouvez utiliser la commande suivante :

```
ALTER TABLE table_name SET LOGGED;
```

Cette opération réécrit l'intégralité de la table et y applique un verrou exclusif jusqu'à ce que l'opération soit terminée. Pour les grandes tables, cela peut entraîner des durées d'indisponibilité importantes.

Tables non journalisées et réplication logique

Les tables non journalisées ne sont généralement pas incluses dans la réplication logique, car la réplication logique repose sur le journal WAL pour capturer et transférer les modifications. Par défaut, les modifications apportées aux tables non journalisées ne sont pas journalisées dans le journal WAL et sont exclues du flux de réplication, ce qui les rend inadaptées aux cas d'utilisation nécessitant une réplication logique. Cependant, Aurora PostgreSQL fournit un paramètre appelé `rds.logically_replicate_unlogged_tables` qui vous permet de contrôler ce comportement :

- Lorsque `rds.logically_replicate_unlogged_tables` est défini sur 0 (désactivé), les tables non journalisées sont exclues de la réplication logique.
- Lorsque `rds.logically_replicate_unlogged_tables` est défini sur 1 (activé), les tables non journalisées sont incluses dans la réplication logique.

> **Note**
>
> Dans Aurora PostgreSQL, le paramètre `rds.logically_replicate_unlogged_tables` est défini par défaut sur 1 (activé) dans les versions 14 et antérieures, et sur 0 (désactivé) dans les versions 15 et ultérieures.

Utilisation de la fonction autovacuum de PostgreSQL sur Amazon Aurora PostgreSQL

Nous vous conseillons vivement d'utiliser la fonction autovacuum afin de maintenir l'intégrité de votre instance de base de données PostgreSQL. La fonction autovacuum automatise le lancement des commandes VACUUM et ANALYZE. Elle vérifie les tables ayant eu un grand nombre de tuples

insérés, mis à jour ou supprimés. Après cette vérification, elle récupère le stockage en supprimant les données ou les tuples obsolètes de la base de données PostgreSQL.

Par défaut, la fonction autovacuum est activée sur les instances de base de données Aurora PostgreSQL que vous créez en utilisant l'un des groupes de paramètres de base de données PostgreSQL par défaut. Les autres paramètres de configuration associés à la fonction autovacuum sont également définis par défaut. Comme ces valeurs par défaut sont relativement génériques, vous pouvez bénéficier du réglage de certains paramètres associés à la fonction d'autovacuum pour votre charge de travail spécifique.

Vous trouverez ci-dessous de plus amples informations sur l'autovacuum et sur la façon de régler certains de ses paramètres sur votre instance de base de données Aurora PostgreSQL.

Rubriques

Allocation de mémoire pour la fonction autovacuum

L'un des paramètres les plus importants qui influencent les performances d'autovacuum est `autovacuum_work_mem`. Dans Aurora PostgreSQL versions 14 et antérieures, le paramètre `autovacuum_work_mem` est défini sur -1, ce qui indique que le paramètre `maintenance_work_mem` est utilisé à la place. Pour toutes les autres versions, `autovacuum_work_mem` est déterminé par GREATEST({DBInstanceClassMemory/32768}, 65536).

Les opérations de vacuum manuel utilisent toujours le paramètre `maintenance_work_mem`, avec le paramètre par défaut GREATEST({DBInstanceClassMemory/63963136*1024}, 65536). Il peut également être ajusté au niveau de la session à l'aide de la commande `SET` pour des opérations `VACUUM` manuelles plus ciblées.

`autovacuum_work_mem` détermine la mémoire permettant à l'autovacuum de contenir les identifiants des tuples inactifs (`pg_stat_all_tables.n_dead_tup`) pour les opérations de vacuum d'index.

Lors des calculs visant à déterminer la valeur du paramètre `autovacuum_work_mem`, tenez compte des points suivants :

- Si vous définissez une valeur trop basse pour ce paramètre, le processus vacuum pourrait avoir à analyser la table plusieurs fois pour mener à bien sa tâche. Ces nombreuses analyses peuvent avoir un impact négatif sur les performances. Pour les instances plus grandes, le réglage de `maintenance_work_mem` ou `autovacuum_work_mem` sur au moins 1 Go peut améliorer les performances des opérations de vacuum des tables contenant un grand nombre de tuples inactifs. Cependant, dans les versions 16 et antérieures de PostgreSQL, l'utilisation de la mémoire de vacuum est limitée à 1 Go, ce qui est suffisant pour traiter environ 179 millions de tuples inactifs en un seul passage. Si une table contient plus de tuples inactifs que cela, l'opération de vacuum doit effectuer plusieurs passages dans les index de la table, ce qui augmente considérablement le temps requis. À partir de la version 17 de PostgreSQL, il n'y a pas de limite de 1 Go et l'autovacuum peut traiter plus de 179 millions de tuples en utilisant des arbres radix.

  Un identifiant de tuple a une taille de 6 octets. Pour estimer la mémoire nécessaire afin d'effectuer le vacuum d'un index d'une table, interrogez `pg_stat_all_tables.n_dead_tup` pour chercher le nombre de tuples inactifs, puis multipliez ce nombre par 6 pour déterminer la mémoire requise pour effectuer le vacuum de l'index en un seul passage. Vous pouvez utiliser la requête suivante :

  ```
  SELECT
      relname AS table_name,
      n_dead_tup,
      pg_size_pretty(n_dead_tup * 6) AS estimated_memory
  FROM
      pg_stat_all_tables
  WHERE
      relname = 'name_of_the_table';
  ```

- Le paramètre `autovacuum_work_mem` fonctionne en conjonction avec le paramètre `autovacuum_max_workers`. Chaque application de travail parmi `autovacuum_max_workers`

peut utiliser la mémoire que vous allouez. Si vous avez beaucoup de petites tables, allouez plus de `autovacuum_max_workers` et moins de `autovacuum_work_mem`. Si vous avez de grandes tables (d'une taille supérieure à 100 Go), allouez plus de mémoire et moins de processus de travail. Vous devez avoir alloué suffisamment de mémoire pour pouvoir prendre en charge votre plus grande table. Assurez-vous donc que la combinaison des processus de travail et de la mémoire est égale à la mémoire totale que vous souhaitez allouer.

Réduction de la probabilité de bouclage de l'ID de transaction

Dans certains cas, les valeurs du groupe de paramètres associées à la fonction autovacuum peuvent ne pas être suffisamment agressives pour empêcher le bouclage de l'ID de transaction. Pour résoudre ce problème, Aurora PostgreSQL fournit un mécanisme qui adapte automatiquement les valeurs des paramètres d'autovacuum. Autovacuum adaptatif est une fonctionnalité pour Aurora PostgreSQL. Une explication détaillée du bouclage de l'ID de transaction figure dans la documentation PostgreSQL.

L'autovacuum adaptatif est activé par défaut pour les instances Aurora PostgreSQL avec le paramètre dynamique `rds.adaptive_autovacuum` défini sur ON. Nous vous recommandons vivement de garder cette option activée. Toutefois, pour désactiver le réglage adaptatif des paramètres d'autovacuum, définissez le paramètre `rds.adaptive_autovacuum` sur 0 ou OFF.

Le bouclage de l'ID de transaction reste possible même lorsque Aurora Amazon RDS ajuste les paramètres d'autovacuum. Nous vous encourageons à implémenter une alarme Amazon CloudWatch pour le bouclage de l'ID de transaction. Pour plus d'informations, consultez l'article Implement an early warning system for transaction ID wraparound in RDS pour PostgreSQL sur le blog de base de données AWS.

Lorsque le réglage adaptatif des paramètres d'autovacuum est activé, Amazon RDS commence à ajuster les paramètres d'autovacuum lorsque la métrique CloudWatch `MaximumUsedTransactionIDs` atteint la valeur du paramètre `autovacuum_freeze_max_age` ou 500 000 000, quelle que soit la valeur la plus élevée.

Amazon RDS continue à ajuster les paramètres pour la fonction autovacuum si une table continue à s'orienter vers le bouclage de l'ID de transaction. Chacun de ces ajustements dédie plus de ressources à la fonction d'autovacuum pour éviter le bouclage. Amazon RDS met à jour les paramètres suivants associés à la fonction d'autovacuum :

- autovacuum_vacuum_cost_delay

- autovacuum_vacuum_cost_limit
- `autovacuum_work_mem`
- autovacuum_naptime

RDS modifie ces paramètres seulement si la nouvelle valeur rend la fonction d'autovacuum plus agressive. Ces paramètres sont modifiés dans la mémoire sur l'instance de base de données. Les valeurs figurant dans le groupe de paramètres ne sont pas modifiées. Pour afficher les paramètres en mémoire actuels, utilisez la commande SQL SHOW de PostgreSQL.

Chaque fois que Amazon RDS modifie l'un de ces paramètres d'autovacuum, il génère un événement pour l'instance de base de données concernée. Cet événement est visible sur l'AWS Management Console et via l'API Amazon RDS. Une fois que la métrique CloudWatch `MaximumUsedTransactionIDs` est repassée sous le seuil, Amazon RDS réinitialise les paramètres associés à la fonction autovacuum en mémoire en rétablissant les valeurs spécifiées dans le groupe de paramètres. Il génère ensuite un autre événement correspondant à cette modification.

Déterminer si les tables de votre base de données ont besoin d'une opération VACUUM

Vous pouvez utiliser la requête suivante pour afficher le nombre de transactions dégelées dans une base de données. La colonne `datfrozenxid` de la ligne `pg_database` d'une base de données est une limite inférieure appliquée aux ID de transaction normaux qui apparaissent dans cette base de données. Cette colonne représente le minimum des valeurs `relfrozenxid` par table au sein de la base de données.

```
SELECT datname, age(datfrozenxid) FROM pg_database ORDER BY age(datfrozenxid) desc
 limit 20;
```

Par exemple, les résultats de l'exécution de la requête précédente pourraient être les suivants.

```
datname    | age
mydb       | 1771757888
template0  | 1721757888
template1  | 1721757888
rdsadmin   | 1694008527
postgres   | 1693881061
(5 rows)
```

Lorsque l'âge d'une base de données atteint 2 milliards d'ID de transactions, un bouclage de l'ID de transaction (XID) se produit et la base de données passe en lecture seule. Vous pouvez utiliser cette requête pour produire une métrique et l'exécuter plusieurs fois par jour. Par défaut, autovacuum est défini pour conserver un âge de transactions inférieur à 200,000,000 (`autovacuum_freeze_max_age`).

Un exemple de politique de surveillance peut ressembler à ceci :

- Définissez la valeur `autovacuum_freeze_max_age` sur 200 millions de transactions.
- Si une table atteint les 500 millions de transactions dégelées, elle déclenche une alarme de faible gravité. Ce n'est pas une valeur déraisonnable, mais elle peut indiquer que la fonction d'autovacuum ne suit pas.
- Si l'âge d'une table atteint 1 milliard, cela doit être considéré comme une alarme exigeant une action. En général, il est conseillé de conserver des âges plus proches de `autovacuum_freeze_max_age` pour des raisons de performances. Nous vous recommandons d'enquêter en appliquant les recommandations suivantes.
- Si une table atteint les 1,5 million de transactions non vidées, elle déclenche une alarme de haute gravité. En fonction de la vitesse à laquelle votre base de données utilise les ID de transaction, cette alarme peut indiquer que le système n'a presque plus de temps pour exécuter le processus d'autovacuum. Dans ce cas, nous vous recommandons une résolution immédiate.

Si une table enfreint constamment ces seuils, vous devez continuer à modifier vos paramètres d'autovacuum. Par défaut, l'utilisation manuelle de VACUUM (pour lequel les retards basés sur les coûts sont désactivés) est plus agressive que le processus d'autovacuum par défaut, mais elle est également plus intrusive pour le système dans son ensemble.

Nous vous recommandons la procédure suivante :

- Gardez tout cela à l'esprit et activez un mécanisme de surveillance afin de connaître l'âge de vos transactions les plus anciennes.

  Pour plus d'informations sur la création d'un processus qui vous avertisse du bouclage des ID de transaction, consultez le billet de blog de base de données AWS Implement an early warning system for transaction ID wraparound in Amazon RDS pour PostgreSQL.
- Pour les tables plus occupées, procédez régulièrement au gel manuel du processus de vacuum pendant une fenêtre de maintenance, en plus de compter sur la fonction d'autovacuum. Pour plus

d'informations sur le gel manuel du processus vacuum, consultez Réalisation d'un gel manuel du processus vacuum.

Déterminer les tables actuellement éligibles pour autovacuum

Souvent, une ou deux tables ont besoin d'une opération VACUUM. Les tables dont la valeur `relfrozenxid` est supérieure au nombre de transactions dans `autovacuum_freeze_max_age` sont toujours ciblées par la fonction d'autovacuum. Sinon, si le nombre de tuples rendus obsolètes depuis la dernière opération VACUUM dépasse le seuil de vacuum, la table est vidée.

Le seuil d'autovacuum est défini comme suit :

```
Vacuum-threshold = vacuum-base-threshold + vacuum-scale-factor * number-of-tuples
```

où le `vacuum base threshold` est `autovacuum_vacuum_threshold`, le `vacuum scale factor` est `autovacuum_vacuum_scale_factor` et le `number of tuples` est `pg_class.reltuples`.

Pendant que vous êtes connecté à votre base de données, exécutez la requête suivante pour afficher la liste des tables qu'autovacuum considère comme éligibles pour une action vacuum.

```
WITH vbt AS (SELECT setting AS autovacuum_vacuum_threshold FROM
pg_settings WHERE name = 'autovacuum_vacuum_threshold'),
vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor FROM
pg_settings WHERE name = 'autovacuum_vacuum_scale_factor'),
fma AS (SELECT setting AS autovacuum_freeze_max_age FROM pg_settings WHERE name =
 'autovacuum_freeze_max_age'),
sto AS (select opt_oid, split_part(setting, '=', 1) as param,
split_part(setting, '=', 2) as value from (select oid opt_oid, unnest(reloptions)
 setting from pg_class) opt)
SELECT '"'||ns.nspname||'"."'||c.relname||'"' as relation,
pg_size_pretty(pg_table_size(c.oid)) as table_size,
age(relfrozenxid) as xid_age,
coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age,
(coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples)
AS autovacuum_vacuum_tuples, n_dead_tup as dead_tuples FROM
pg_class c join pg_namespace ns on ns.oid = c.relnamespace
join pg_stat_all_tables stat on stat.relid = c.oid join vbt on (1=1) join vsf on (1=1)
 join fma on (1=1)
```

```
left join sto cvbt on cvbt.param = 'autovacuum_vacuum_threshold' and c.oid =
 cvbt.opt_oid
left join sto cvsf on cvsf.param = 'autovacuum_vacuum_scale_factor' and c.oid =
 cvsf.opt_oid
left join sto cfma on cfma.param = 'autovacuum_freeze_max_age' and c.oid = cfma.opt_oid
WHERE c.relkind = 'r' and nspname <> 'pg_catalog'
AND (age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
OR coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
c.reltuples <= n_dead_tup)
ORDER BY age(relfrozenxid) DESC LIMIT 50;
```

Déterminer si autovacuum est en cours d'exécution et pour combien de temps

Si vous avez besoin de vider manuellement une table, vous devez déterminer si autovacuum est en cours d'exécution. Si c'est le cas, vous devrez peut-être ajuster les paramètres pour le faire fonctionner plus efficacement, ou mettre fin à autovacuum afin de pouvoir exécuter manuellement VACUUM.

Utilisez la requête suivante pour déterminer si autovacuum est en cours d'exécution, pendant combien de temps il a été en cours d'exécution et s'il est en attente sur une autre session.

```
SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
 xact_runtime, query
FROM pg_stat_activity
WHERE upper(query) LIKE '%VACUUM%'
ORDER BY xact_start;
```

Après l'exécution de la requête, vous devez obtenir un résultat similaire à ce qui suit.

```
 datname | usename  |  pid  | state  | wait_event |      xact_runtime       | query
 --------+----------+-------+--------+------------+-------------------------
+-----------------------------------------------------------------------------------------------
 mydb    | rdsadmin | 16473 | active |            | 33 days 16:32:11.600656 |
 autovacuum: VACUUM ANALYZE public.mytable1 (to prevent wraparound)
 mydb    | rdsadmin | 22553 | active |            | 14 days 09:15:34.073141 |
 autovacuum: VACUUM ANALYZE public.mytable2 (to prevent wraparound)
 mydb    | rdsadmin | 41909 | active |            | 3 days 02:43:54.203349  |
 autovacuum: VACUUM ANALYZE public.mytable3
```

```
mydb    | rdsadmin |   618 | active |            | 00:00:00                |
SELECT datname, usename, pid, state, wait_event, current_timestamp - xact_start AS
xact_runtime, query+
        |          |       |        |            |                         | FROM
pg_stat_activity
          +
        |          |       |        |            |                         | WHERE
query like '%VACUUM%'
         +
        |          |       |        |            |                         | ORDER BY
xact_start;
      +
```

Plusieurs problèmes peuvent occasionner des longueurs d'exécution (plusieurs jours) d'une session autovacuum. Le problème le plus courant est que la valeur de votre paramètre `maintenance_work_mem` est trop basse pour la taille de la table ou pour la fréquence des mises à jour.

Nous vous recommandons d'utiliser la formule suivante pour définir la valeur du paramètre `maintenance_work_mem`.

```
GREATEST({DBInstanceClassMemory/63963136*1024},65536)
```

De courtes sessions autovacuum peuvent également indiquer des problèmes :

- Cela peut indiquer un nombre `autovacuum_max_workers` insuffisant pour votre charge de travail. Dans ce cas, vous devez indiquer le nombre d'exécutants.
- Cela peut indiquer une corruption d'index (la fonction d'autovacuum se bloque et redémarre sur la même relation, mais ne progresse pas). Dans ce cas, exécutez un `vacuum freeze verbose` *`table`* manuel pour voir la cause exacte.

Réalisation d’un gel manuel du processus vacuum

Vous pouvez effectuer un gel manuel sur une table pour laquelle un processus vacuum est déjà en cours. C’est utile si vous avez identifié une table avec un âge proche de 2 milliards de transactions (ou supérieur à tous les seuils que vous surveillez).

Les étapes suivantes sont fournies à titre informatif et il existe plusieurs variantes de ce processus. Par exemple, pendant le test, supposons que vous trouviez que la valeur du paramètre `maintenance_work_mem` a été définie trop bas et que vous devez agir immédiatement sur une

table. Toutefois, vous ne voulez pas renvoyer l'instance à l'expéditeur pour le moment. À l'aide des requêtes des sections précédentes, vous déterminez quelle table pose problème et remarquez une session autovacuum en cours d'exécution depuis longtemps. Vous savez que vous devez modifier le paramètre `maintenance_work_mem`, mais vous devez également agir immédiatement et effectuer un processus vacuum sur la table concernée. La procédure suivante montre ce que vous devez faire dans cette situation.

Pour procéder manuellement au gel du processus vacuum

1. Ouvrez les deux sessions de la base de données contenant la table sur laquelle vous voulez effectuer le processus vacuum. Pour la seconde session, utilisez « écran » ou un autre utilitaire qui gère la session si votre connexion est abandonnée.
2. Dans la première session, obtenez le PID de la session autovacuum en cours d'exécution sur la table.

   Exécutez la requête suivante pour obtenir le PID de la session autovacuum.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) LIKE '%VACUUM%' ORDER BY
   xact_start;
   ```
3. Dans la deuxième session, calculez la quantité de mémoire dont vous avez besoin pour cette opération. Dans cet exemple, nous déterminons que nous pouvons nous permettre d'utiliser jusqu'à 2 Go de mémoire pour cette opération. Nous affectons donc 2 Go à `maintenance_work_mem` pour la session en cours.

   ```
   SET maintenance_work_mem='2 GB';
   SET
   ```
4. Dans la deuxième session, émettez une commande `vacuum freeze verbose` pour la table. Le paramètre de mode détaillé est utile, car il vous permet de voir l'activité bien qu'il n'existe actuellement aucun rapport d'avancement de cette opération dans PostgreSQL.

   ```
   \timing on
   Timing is on.
   vacuum freeze verbose pgbench_branches;
   ```

   ```
   INFO:  vacuuming "public.pgbench_branches"
   ```

```
INFO:  index "pgbench_branches_pkey" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  index "pgbench_branches_test_index" now contains 50 row versions in 2 pages
DETAIL:  0 index row versions were removed.
0 index pages have been deleted, 0 are currently reusable.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
INFO:  "pgbench_branches": found 0 removable, 50 nonremovable row versions
     in 43 out of 43 pages
DETAIL:  0 dead row versions cannot be removed yet.
There were 9347 unused item pointers.
0 pages are entirely empty.
CPU 0.00s/0.00u sec elapsed 0.00 sec.
VACUUM
Time: 2.765 ms
```

5. Dans la première session, si l'autovacuum provoquait un blocage de la session de vacuum, `pg_stat_activity` indique que l'attente a la valeur T pour votre session de vacuum. Dans ce cas, mettez fin au processus d'autovacuum comme suit.

```
SELECT pg_terminate_backend('the_pid');
```

> **Note**
>
> Certaines versions antérieures d'Amazon Aurora ne peuvent pas mettre fin à un processus d'autovacuum à l'aide de la commande précédente et échouent avec l'erreur suivante : `ERROR: 42501: must be a superuser to terminate superuser process LOCATION: pg_terminate_backend, signalfuncs.c:227`. Pour trouver les versions de PostgreSQL auxquelles des correctifs ont été appliqués, recherchez le point suivant dans les mises à jour d'Amazon Aurora PostgreSQL :
>
> ```
> Allow rds_superuser to terminate backends which are not explicitly
>  associated with a role
> ```

À ce stade, votre session commence. L'autovacuum redémarre immédiatement parce que cette table figure probablement tout en haut de sa liste de tâches.

6. Lancez votre commande `vacuum freeze verbose` dans la session 2, puis terminez le processus autovacuum de la session 1.

Réindexation d'une table pendant l'exécution du processus autovacuum

Si un index a été corrompu, la fonction d'autovacuum continue à traiter la table et échoue. Si vous essayez d'effectuer un processus vacuum manuel dans cette situation, vous recevez un message d'erreur similaire à ce qui suit.

```
postgres=>  vacuum freeze pgbench_branches;
ERROR: index "pgbench_branches_test_index" contains unexpected
   zero page at block 30521
HINT: Please REINDEX it.
```

Lorsque l'index est corrompu et que la fonction d'autovacuum tente de s'exécuter sur la table, vous vous heurtez à une session autovacuum déjà en cours d'exécution. Lorsque vous émettez une commande REINDEX, vous retirez un verrou exclusif sur la table. Les opérations d'écriture sont bloquées, ainsi que les opérations de lecture qui utilisent cet index spécifique.

Pour réindexer une table lorsque la fonction d'autovacuum est en cours d'exécution sur la table

1. Ouvrez les deux sessions de la base de données contenant la table sur laquelle vous voulez effectuer le processus vacuum. Pour la seconde session, utilisez « écran » ou un autre utilitaire qui gère la session si votre connexion est abandonnée.
2. Dans la première session, obtenez le PID de la session autovacuum en cours d'exécution sur la table.

   Exécutez la requête suivante pour obtenir le PID de la session autovacuum.

   ```
   SELECT datname, usename, pid, current_timestamp - xact_start
   AS xact_runtime, query
   FROM pg_stat_activity WHERE upper(query) like '%VACUUM%' ORDER BY
   xact_start;
   ```

3. Dans la deuxième session, émettez la commande reindex.

   ```
   \timing on
   Timing is on.
   reindex index pgbench_branches_test_index;
   ```

```
REINDEX
  Time: 9.966 ms
```

4. Dans la première session, si autovacuum provoquait un blocage du processus, vous voyez dans `pg_stat_activity` que l'attente a la valeur « T » pour votre session vacuum. Dans ce cas, vous mettrez fin au processus autovacuum.

   ```
   SELECT pg_terminate_backend('the_pid');
   ```

   À ce stade, votre session commence. Il est important de noter que la fonction d'autovacuum redémarre immédiatement parce que cette table figure probablement tout en haut de sa liste de tâches.

5. Lancez votre commande dans la session 2, puis terminez le processus autovacuum de la session 1.

Gestion de la fonction autovacuum avec de grands index

Dans le cadre de son fonctionnement, la fonction autovacuum effectue plusieurs phases de mise à vide lorsqu'elle s'exécute sur une table. Avant que la table ne soit nettoyée, tous ses index sont d'abord vidés. Lorsque vous supprimez plusieurs grands index, cette phase consomme beaucoup de temps et de ressources. Par conséquent, il est recommandé de contrôler le nombre d'index d'une table et d'éliminer les index inutilisés.

Pour ce processus, vérifiez d'abord la taille globale de l'index. Déterminez ensuite s'il existe des index potentiellement inutilisés qui peuvent être supprimés comme le montrent les exemples suivants.

Pour vérifier la taille de la table et de ses index

```
postgres=> select pg_size_pretty(pg_relation_size('pgbench_accounts'));
pg_size_pretty
6404 MB
(1 row)
```

```
postgres=> select pg_size_pretty(pg_indexes_size('pgbench_accounts'));
pg_size_pretty
11 GB
(1 row)
```

Dans cet exemple, la taille des index est supérieure à celle de la table. Cette différence peut entraîner des problèmes de performances, car les index sont surchargés ou inutilisés, ce qui a une incidence sur la fonction autovacuum ainsi que sur les opérations d'insertion.

Pour vérifier la présence d'index inutilisés

À l'aide de la vue pg_stat_user_indexes, vous pouvez vérifier la fréquence d'utilisation d'un index avec la colonne `idx_scan`. Dans l'exemple suivant, les index non utilisés ont la valeur `idx_scan` définie sur `0`.

```
postgres=> select * from pg_stat_user_indexes where relname = 'pgbench_accounts' order
 by idx_scan desc;

relid  | indexrelid | schemaname | relname          | indexrelname          | idx_scan
 | idx_tup_read | idx_tup_fetch
-------+------------+------------+------------------+-----------------------+----------
+--------------+---------------
16433  | 16454      | public     | pgbench_accounts | index_f               | 6
 | 6            | 0
16433  | 16450      | public     | pgbench_accounts | index_b               | 3
 | 199999       | 0
16433  | 16447      | public     | pgbench_accounts | pgbench_accounts_pkey | 0
 | 0            | 0
16433  | 16452      | public     | pgbench_accounts | index_d               | 0
 | 0            | 0
16433  | 16453      | public     | pgbench_accounts | index_e               | 0
 | 0            | 0
16433  | 16451      | public     | pgbench_accounts | index_c               | 0
 | 0            | 0
16433  | 16449      | public     | pgbench_accounts | index_a               | 0
 | 0            | 0
(7 rows)
```

```
postgres=> select schemaname, relname, indexrelname, idx_scan from pg_stat_user_indexes
 where relname = 'pgbench_accounts' order by idx_scan desc;

schemaname  | relname          | indexrelname          | idx_scan
------------+------------------+-----------------------+----------
public      | pgbench_accounts | index_f               | 6
public      | pgbench_accounts | index_b               | 3
public      | pgbench_accounts | pgbench_accounts_pkey | 0
```

```
public      | pgbench_accounts | index_d              | 0
public      | pgbench_accounts | index_e              | 0
public      | pgbench_accounts | index_c              | 0
public      | pgbench_accounts | index_a              | 0
(7 rows)
```

> **Note**
>
> Ces statistiques sont incrémentielles à partir du moment où elles sont réinitialisées. Supposons que vous disposiez d'un index qui n'est utilisé qu'à la fin d'un trimestre ou uniquement pour un rapport spécifique. Il est possible que cet index n'ait pas été utilisé depuis la réinitialisation des statistiques. Pour plus d'informations, consultez Statistics Functions (Fonctions statistiques). Les index utilisés pour renforcer l'unicité ne seront pas analysés et ne devraient pas être identifiés comme des index inutilisés. Pour identifier les index inutilisés, vous devez avoir une connaissance approfondie de l'application et de ses requêtes.

Pour vérifier quand les statistiques ont été réinitialisées pour la dernière fois pour une base de données, utilisez `pg_stat_database`

```
postgres=> select datname, stats_reset from pg_stat_database where datname =
 'postgres';

datname   | stats_reset
----------+-------------------------------
postgres  | 2022-11-17 08:58:11.427224+00
(1 row)
```

Vidage d'une table le plus rapidement possible

RDS pour PostgreSQL versions 12 et ultérieures

Si vous avez trop d'index dans une grande table, il se peut que votre instance de base de données soit proche du bouclage de l'ID de transaction (XID), c'est-à-dire lorsque le compteur XID revient à zéro. Si elle n'est pas vérifiée, cette situation peut entraîner une perte de données. Toutefois, vous pouvez rapidement vider la table sans nettoyer les index. Dans RDS pour PostgreSQL versions 12 et ultérieures, vous pouvez utiliser VACUUM avec la clause `INDEX_CLEANUP`.

```
postgres=> VACUUM (INDEX_CLEANUP FALSE, VERBOSE TRUE) pgbench_accounts;

INFO: vacuuming "public.pgbench_accounts"
INFO: table "pgbench_accounts": found 0 removable, 8 nonremovable row versions in 1 out
 of 819673 pages
DETAIL: 0 dead row versions cannot be removed yet, oldest xmin: 7517
Skipped 0 pages due to buffer pins, 0 frozen pages.
CPU: user: 0.01 s, system: 0.00 s, elapsed: 0.01 s.
```

Si une session de mise à vide automatique est déjà en cours, vous devez y mettre fin pour démarrer le processus VACUUM manuel. Pour plus d'informations sur le gel manuel du processus vacuum, consultez Réalisation d'un gel manuel du processus vacuum.

> **Note**
>
> Ignorer régulièrement le nettoyage de l'index entraîne un gonflement de l'index, ce qui dégrade les performances des analyses. L'index conserve les lignes inactives et le tableau conserve les pointeurs de ligne inactive. Par conséquent, `pg_stat_all_tables.n_dead_tup` augmente jusqu'à l'exécution d'un autovacuum ou d'un VACUUM manuel avec nettoyage d'index. Une bonne pratique consiste à n'utiliser la procédure précédente que pour empêcher le bouclage de l'ID de transaction.

RDS pour PostgreSQL versions 11 et ultérieures

Toutefois, dans RDS pour PostgreSQL versions 11 et ultérieures, la seule façon de permettre au processus vacuum de se terminer plus rapidement est de réduire le nombre d'index sur une table. La suppression d'un index peut affecter les plans de requête. Nous vous recommandons de supprimer d'abord les index inutilisés, puis de les supprimer lorsque le bouclage de l'ID de transaction est très proche. Une fois le processus vacuum terminé, vous pouvez recréer ces index.

Autres paramètres qui affectent la fonction d'autovacuum

Cette requête affiche les valeurs de certains des paramètres qui ont un impact direct sur la fonction d'autovacuum et son comportement. Les paramètres d'autovacuum sont décrits en détails dans la documentation PostgreSQL.

```
SELECT name, setting, unit, short_desc
FROM pg_settings
```

```
WHERE name IN (
'autovacuum_max_workers',
'autovacuum_analyze_scale_factor',
'autovacuum_naptime',
'autovacuum_analyze_threshold',
'autovacuum_analyze_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_scale_factor',
'autovacuum_vacuum_threshold',
'autovacuum_vacuum_cost_delay',
'autovacuum_vacuum_cost_limit',
'vacuum_cost_limit',
'autovacuum_freeze_max_age',
'maintenance_work_mem',
'vacuum_freeze_min_age');
```

Tous ces paramètres affectent la fonction d'autovacuum, mais les plus importants sont :

- maintenance_work_mem
- autovacuum_freeze_max_age
- autovacuum_max_workers
- autovacuum_vacuum_cost_delay
- autovacuum_vacuum_cost_limit

Définition des paramètres d'autovacuum au niveau de la table

Les paramètres de stockage liés à la fonction d'autovacuum peuvent être définis au niveau de la table, ce qui peut être plus judicieux que de modifier le comportement de toute la base de données. Pour les grandes tables, vous devrez peut-être définir des paramètres agressifs et il est déconseillé de faire en sorte que la fonction d'autovacuum se comporte de cette manière pour toutes les tables.

La requête suivante affiche les tables qui ont actuellement des options en place au niveau de la table.

```
SELECT relname, reloptions
FROM pg_class
WHERE reloptions IS NOT null;
```

Par exemple, cela peut être utile sur les tables qui sont beaucoup plus grandes que le reste de vos tables. Supposez que vous avez une table de 300 Go et 30 autres tables de moins de 1 Go. Dans ce

cas, vous pouvez définir des paramètres spécifiques pour votre grande table afin de ne pas modifier le comportement de l'intégralité de votre système.

```
ALTER TABLE mytable set (autovacuum_vacuum_cost_delay=0);
```

Cela permet de désactiver le retard d'autovacuum basé sur les coûts pour cette table au détriment d'une plus grande utilisation des ressources sur votre système. Normalement, l'autovacuum s'arrête pour `autovacuum_vacuum_cost_delay` à chaque fois que `autovacuum_cost_limit` est atteinte. Pour plus d'informations, veuillez consulter la documentation PostgreSQL concernant le processus de vacuum basé sur les coûts.

Enregistrement des activités d'autovacuum et de vacuum

Les informations sur les activités d'autovacuum sont envoyées au `postgresql.log` en fonction du niveau spécifié dans le paramètre `rds.force_autovacuum_logging_level`. Voici les valeurs autorisées pour ce paramètre et les versions de PostgreSQL pour lesquelles cette valeur est le paramètre par défaut :

- `disabled` (PostgreSQL 10, PostgreSQL 9.6)
- `debug5`, `debug4`, `debug3`, `debug2`, `debug1`
- `info` (PostgreSQL 12, PostgreSQL 11)
- `notice`
- `warning` (PostgreSQL versions 13 et ultérieures)
- `error`, journal, `fatal`, `panic`

`rds.force_autovacuum_logging_level` fonctionne avec le paramètre `log_autovacuum_min_duration`. La valeur du paramètre `log_autovacuum_min_duration` est le seuil (en millisecondes) au-dessus duquel les actions autovacuum sont enregistrées. Une valeur de `-1` n'enregistre rien, tandis qu'une valeur de 0 enregistre toutes les actions. Comme avec `rds.force_autovacuum_logging_level`, valeurs par défaut pour `log_autovacuum_min_duration` dépendent de la version, comme suit :

- `10000 ms` : PostgreSQL 14, PostgreSQL 13, PostgreSQL 12 et PostgreSQL 11
- `(empty)` : aucune valeur par défaut pour PostgreSQL 10 et PostgreSQL 9.6

Nous vous recommandons de définir le `rds.force_autovacuum_logging_level` à la valeur `WARNING`. Nous vous recommandons également de définir `log_autovacuum_min_duration` à une valeur comprise entre 1000 et 5000. Un paramètre de 5000 journaux d'activité qui prend plus de 5000 millisecondes. Tout paramètre autre que -1 enregistre également les messages si l'action autovacuum est ignorée en raison d'un verrouillage en conflit ou d'une perte simultanée de relations. Pour plus d'informations, veuillez consulter Action Vacuum automatique dans la documentation PostgreSQL.

Pour résoudre les problèmes, vous pouvez modifier le paramètre `rds.force_autovacuum_logging_level` à l'un des niveaux de débogage, de `debug1` jusqu'à `debug5` pour obtenir les informations les plus détaillées. Nous vous recommandons d'utiliser les paramètres de débogage pendant de courtes périodes et à des fins de dépannage uniquement. Pour en savoir plus, veuillez consulter la rubrique Quand journaliser dans la documentation de PostgreSQL.

> **Note**
>
> PostgreSQL permet au compte `rds_superuser` d'afficher les sessions autovacuum dans `pg_stat_activity`. Par exemple, vous pouvez identifier et mettre fin à la session qui bloque l'exécution d'une commande ou empêche la commande de s'exécuter plus lentement qu'une commande vacuum exécutée manuellement.

Comprendre le comportement de l’autovacuum avec les bases de données non valides

Une nouvelle valeur `-2` est introduite dans la colonne `datconnlimit` du catalogue `pg_database` pour indiquer que les bases de données qui ont été interrompues au milieu de l’opération DROP DATABASE ne sont pas valides.

Cette nouvelle valeur est disponible dans les versions suivantes d’Aurora PostgreSQL :

- Version 15.4 et toutes les versions ultérieures
- 14.9 et versions ultérieures
- 13.12 et versions ultérieures
- 12.16 et versions ultérieures
- 11.21 et versions ultérieures

Les bases de données non valides n'affectent pas la capacité de la fonction d'autovacuum à geler la fonctionnalité des bases de données valides. Autovacuum ignore les bases de données non valides. Par conséquent, les opérations de vacuum régulières continueront à fonctionner correctement et efficacement pour toutes les bases de données valides de votre environnement PostgreSQL.

Rubriques

Surveillance de l'ID de transaction

Cette fonction `age(datfrozenxid)` est couramment utilisée pour surveiller l'âge des ID de transaction (XID) des bases de données afin d'empêcher le bouclage de l'ID de transaction.

Les bases de données non valides étant exclues de l'autovacuum, leur compteur d'ID de transaction (XID) peut atteindre la valeur maximale de `2 billion`, revenir à `- 2 billion` et poursuivre ce cycle indéfiniment. Une requête typique pour surveiller le bouclage de l'ID de transaction peut ressembler à ceci :

```
SELECT max(age(datfrozenxid)) FROM pg_database;
```

Cependant, avec l'introduction de la valeur -2 pour `datconnlimit`, les bases de données non valides peuvent fausser les résultats de cette requête. Étant donné que ces bases de données ne sont pas valides et ne doivent pas faire l'objet de contrôles de maintenance réguliers, elles peuvent générer des faux positifs, ce qui vous laisse penser que `age(datfrozenxid)` est supérieur à ce qu'il est réellement.

Ajustement de la requête de surveillance

Pour garantir une surveillance précise, vous devez ajuster votre requête de surveillance afin d'exclure les bases de données non valides. Suivez cette requête recommandée :

```
SELECT
    max(age(datfrozenxid))
FROM
    pg_database
```

```
WHERE
    datconnlimit <> -2;
```

Cette requête garantit que seules les bases de données valides sont prises en compte dans le calcul `age(datfrozenxid)`, ce qui reflète fidèlement l'âge des ID de transaction dans l'environnement PostgreSQL.

Résolution du problème de base de données non valide

Lorsque vous essayez de vous connecter à une base de données non valide, vous pouvez rencontrer un message d'erreur similaire à ce qui suit :

```
postgres=> \c db1
connection to server at "mydb.xxxxxxxxxx.us-west-2.rds.amazonaws.com" (xx.xx.xx.xxx),
 port xxxx failed: FATAL:  cannot connect to invalid database "db1"
HINT:  Use DROP DATABASE to drop invalid databases.
Previous connection kept
```

En outre, si le paramètre `log_min_messages` est défini sur `DEBUG2` ou supérieur, vous remarquerez peut-être que les entrées de journal suivantes indiquent que le processus autovacuum ignore la base de données non valide :

```
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db6"
2024-07-30 05:59:00 UTC::@:[32000]:DEBUG:  autovacuum: skipping invalid database "db1"
```

Pour résoudre le problème, suivez le `HINT` fourni lors de la tentative de connexion. Connectez-vous à n'importe quelle base de données valide à l'aide de votre compte principal RDS ou d'un compte de base de données doté du rôle `rds_superuser`, puis supprimez les bases de données non valides.

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
```

```
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

Identification et résolution des bloqueurs de vacuum agressifs dans Aurora PostgreSQL

Dans PostgreSQL, l'opération de vacuum est essentielle pour garantir l'intégrité de la base de données, car elle permet de récupérer de l'espace de stockage et d'éviter les problèmes de bouclage de l'ID de transaction. Cependant, il peut arriver que l'opération de vacuum ne fonctionne pas comme vous le souhaitez, ce qui peut entraîner une dégradation des performances ou une surcharge de l'espace de stockage et même avoir un impact sur la disponibilité de votre instance de base de données par bouclage de l'ID de transaction. Il est donc essentiel d'identifier et de résoudre ces problèmes pour optimiser les performances et la disponibilité des bases de données. Lisez Présentation d'autovacuum dans les environnements Amazon RDS pour PostgreSQL pour en savoir plus sur l'autovacuum.

La fonction `postgres_get_av_diag()` permet d'identifier les problèmes qui empêchent ou retardent la progression du vacuum agressif. Des suggestions sont fournies, qui peuvent inclure des commandes pour résoudre le problème lorsqu'il est identifiable ou des conseils pour des diagnostics supplémentaires lorsque le problème n'est pas identifiable. Les bloqueurs de vacuum agressif sont signalés lorsque leur âge dépasse le seuil d'autovacuum adaptatif de 500 millions d'ID de transaction dans RDS.

Quel est l'âge de l'ID de transaction ?

La fonction `age()` pour les ID de transaction calcule le nombre de transactions survenues depuis le plus ancien ID de transaction dégelé pour une base de données (`pg_database.datfrozenxid`) ou une table (`pg_class.relfrozenxid`). Cette valeur indique l'activité de la base de données depuis la dernière opération de vacuum agressif et met en évidence la charge de travail probable pour les prochains processus VACUUM.

Qu'est-ce qu'un vacuum agressif ?

Une opération de vacuum agressif effectue une analyse complète de toutes les pages d'une table, y compris celles qui sont généralement ignorées lors d'un vacuum standard. Cette analyse approfondie vise à « geler » les ID de transaction approchant de leur âge maximum, empêchant ainsi une situation connue sous le nom de bouclage de l'ID de transaction.

Pour que `postgres_get_av_diag()` signale un bloqueur, celui-ci doit dater d'au moins 500 millions de transactions.

Rubriques

Installation d'outils de surveillance et de diagnostic d'autovacuum dans Aurora PostgreSQL

La fonction `postgres_get_av_diag()` est actuellement disponible dans les versions Aurora PostgreSQL suivantes :

- 17.4 et versions 17 ultérieures
- 16.7 et versions 16 ultérieures
- 15.11 et versions 15 ultérieures
- 14.16 et versions 14 ultérieures
- 13.19 et versions 13 ultérieures

Pour pouvoir utiliser `postgres_get_av_diag()`, créez l'extension `rds_tools`.

```
postgres=> CREATE EXTENSION rds_tools ;
CREATE EXTENSION
```

Vérifiez que l'extension est installée.

```
postgres=> \dx rds_tools
             List of installed extensions
   Name    | Version |  Schema   |                    Description
 ----------+---------+-----------
+-------------------------------------------------------------
```

```
 rds_tools |   1.9   | rds_tools | miscellaneous administrative functions for RDS
 PostgreSQL
 1 row
```

Vérifiez que la fonction est créée.

```
postgres=> SELECT
    proname function_name,
    pronamespace::regnamespace function_schema,
    proowner::regrole function_owner
FROM
    pg_proc
WHERE
    proname = 'postgres_get_av_diag';
    function_name     | function_schema | function_owner
----------------------+-----------------+----------------
 postgres_get_av_diag | rds_tools       | rds_superuser
(1 row)
```

Fonctions de postgres_get_av_diag() dans Aurora PostgreSQL

La fonction `postgres_get_av_diag()` récupère des informations de diagnostic sur les processus autovacuum qui bloquent ou sont en retard dans une base de données Aurora PostgreSQL. Pour obtenir des résultats précis, la requête doit être exécutée dans la base de données dont l'ID de transaction est le plus ancien. Pour plus d'informations sur l'utilisation de la base de données ayant l'ID de transaction le plus ancien, consultez Non connecté à la base de données dont l'ID de transaction est le plus ancien.

```
SELECT
    blocker,
    DATABASE,
    blocker_identifier,
    wait_event,
    TO_CHAR(autovacuum_lagging_by, 'FM9,999,999,999') AS autovacuum_lagging_by,
    suggestion,
    suggested_action
FROM (
    SELECT
        *
    FROM
        rds_tools.postgres_get_av_diag ()
    ORDER BY
```

```
        autovacuum_lagging_by DESC) q;
```

La fonction `postgres_get_av_diag()` retourne un tableau avec les informations suivantes :

blocker

Spécifie la catégorie d'activité de base de données qui bloque le vacuum.

- Instruction active
- État Idle in transaction (Transaction inactive)
- Transaction préparée
- Emplacement de réplication logique
- Instances de lecteur
- Tables temporaires

database

Spécifie le nom de la base de données, le cas échéant. Il s'agit de la base de données dans laquelle l'activité est en cours et bloque ou bloquera l'autovacuum. Il s'agit de la base de données à laquelle vous devez vous connecter et sur laquelle vous devez agir.

blocker_identifier

Spécifie l'identifiant de l'activité qui bloque ou bloquera l'autovacuum. Il peut s'agir d'un ID de processus accompagné d'une instruction SQL, d'une transaction préparée, de l'adresse IP d'un réplica en lecture et du nom de l'emplacement de réplication, logique ou physique.

wait_event

Spécifie l'événement d'attente de la session de blocage et s'applique aux bloqueurs suivants :

- Instruction active
- État Idle in transaction (Transaction inactive)

autovacum_lagging_by

Spécifie le nombre de transactions pour lesquelles l'autovacuum a pris du retard dans le traitement des éléments en attente par catégorie.

suggestion

Spécifie des suggestions pour résoudre le bloqueur. Ces instructions incluent le nom de la base de données dans laquelle l'activité existe, le cas échéant, l'ID de processus (PID) de la session, le cas échéant, et les actions à effectuer.

suggested_action

Suggère l'action à effectuer pour résoudre le bloqueur.

Résolution des bloqueurs de vacuum identifiables dans Aurora PostgreSQL

L'autovacuum effectue des opérations vacuum agressives et abaisse l'âge des ID de transaction en dessous du seuil spécifié par le paramètre `autovacuum_freeze_max_age` de votre instance RDS. Vous pouvez suivre cet âge à l'aide de la métrique `MaximumUsedTransactionIDs` d'Amazon CloudWatch.

Pour trouver le paramètre `autovacuum_freeze_max_age` (dont la valeur par défaut est de 200 millions d'ID de transaction) pour votre instance Amazon RDS, vous pouvez utiliser la requête suivante :

```
SELECT
    TO_CHAR(setting::bigint, 'FM9,999,999,999') autovacuum_freeze_max_age
FROM
    pg_settings
WHERE
    name = 'autovacuum_freeze_max_age';
```

Notez que `postgres_get_av_diag()` recherche uniquement les bloqueurs de vacuum agressifs quand l'âge dépasse le seuil de 500 millions d'ID de transaction pour l'autovacuum adaptatif d'Amazon RDS. Pour que `postgres_get_av_diag()` détecte un bloqueur, celui-ci doit dater d'au moins 500 millions de transactions.

La fonction `postgres_get_av_diag()` identifie les types de bloqueurs suivants :

Rubriques

- Instruction active
- État Idle in transaction (Transaction inactive)
- Transaction préparée
- Emplacement de réplication logique
- Instances de lecteur
- Tables temporaires

Instruction active

Dans PostgreSQL, une instruction active est une instruction SQL qui est exécutée par la base de données. Cela inclut les requêtes, les transactions ou les opérations en cours. Lors de la surveillance via `pg_stat_activity`, la colonne d'état indique que le processus avec le PID correspondant est actif.

La fonction `postgres_get_av_diag()` affiche une sortie similaire à la suivante lorsqu'elle identifie une instruction active.

```
blocker               | Active statement
database              | my_database
blocker_identifier    | SELECT pg_sleep(20000);
wait_event            | Timeout:PgSleep
autovacuum_lagging_by | 568,600,871
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (29621);"}
```

Action suggérée

En suivant les instructions de la colonne `suggestion`, l'utilisateur peut se connecter à la base de données dans laquelle l'instruction active se trouve et, comme indiqué dans la colonne `suggested_action`, il est conseillé de vérifier attentivement l'option permettant de mettre fin à la session. Si la résiliation est sûre, vous pouvez utiliser la fonction `pg_terminate_backend()` pour résilier la session. Cette action peut être effectuée par un administrateur (tel que le compte principal RDS) ou par un utilisateur disposant du privilège `pg_terminate_backend()` requis.

> **Warning**
>
> Une session résiliée annule (`ROLLBACK`) les modifications apportées. En fonction de vos besoins, vous souhaiterez peut-être réexécuter l'instruction. Cependant, il est recommandé de ne le faire qu'une fois que le processus d'autovacuum a terminé son opération de vacuum agressif.

État Idle in transaction (Transaction inactive)

Une instruction idle in transaction fait référence à toute session qui a ouvert une transaction explicite (par exemple en émettant une instruction `BEGIN`), qui a effectué un certain travail et qui attend maintenant que le client transmettre plus de travail ou signale la fin de la transaction en émettant une instruction `COMMIT`, `ROLLBACK`, ou `END` (ce qui se traduirait par un `COMMIT` implicite).

La fonction `postgres_get_av_diag()` affiche une sortie similaire à la suivante lorsqu'elle identifie une instruction `idle in transaction` comme un bloqueur.

```
blocker               | idle in transaction
database              | my_database
blocker_identifier    | INSERT INTO tt SELECT * FROM tt;
wait_event            | Client:ClientRead
autovacuum_lagging_by | 1,237,201,759
suggestion            | Connect to database "my_database", review carefully and you
 may consider terminating the process using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"SELECT pg_terminate_backend (28438);"}
```

Action suggérée

Comme indiqué dans la colonne `suggestion`, vous pouvez vous connecter à la base de données dans laquelle la session idle in transaction est présente et mettre fin à la session à l'aide de la fonction `pg_terminate_backend()`. L'utilisateur peut être votre administrateur (compte principal RDS) ou un utilisateur disposant du privilège `pg_terminate_backend()`.

> **Warning**
>
> Une session résiliée annule (`ROLLBACK`) les modifications apportées. En fonction de vos besoins, vous souhaiterez peut-être réexécuter l'instruction. Cependant, il est recommandé de ne le faire qu'une fois que le processus d'autovacuum a terminé son opération de vacuum agressif.

Transaction préparée

PostgreSQL autorise les transactions qui font partie d'une stratégie de validation en deux phases appelée transactions préparées. Elles sont activées en définissant une valeur différente de zéro pour le paramètre `max_prepared_transactions`. Les transactions préparées sont conçues

pour garantir qu'une transaction est durable et reste disponible même après une panne de base de données, un redémarrage ou une déconnexion du client. Comme les transactions régulières, elles reçoivent un ID de transaction et peuvent affecter l'autovacuum. Si elles restent à l'état préparé, l'autovacuum ne peut pas les geler et cela peut entraîner un bouclage de l'ID de transaction.

Lorsque les transactions restent à l'état préparé indéfiniment sans être résolues par un gestionnaire de transactions, elles deviennent des transactions préparées orphelines. La seule façon de résoudre ce problème est de valider ou d'annuler la transaction à l'aide des commandes `COMMIT PREPARED` ou `ROLLBACK PREPARED`, respectivement.

> **Note**
>
> Sachez qu'une sauvegarde effectuée lors d'une transaction préparée contiendra toujours cette transaction après restauration. Reportez-vous aux informations suivantes pour savoir comment localiser et clôturer de telles transactions.

La fonction `postgres_get_av_diag()` affiche le résultat suivant lorsqu'elle identifie un bloqueur qui est une transaction préparée.

```
blocker               | Prepared transaction
database              | my_database
blocker_identifier    | myptx
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database" and consider either COMMIT
 or ROLLBACK the prepared transaction using suggested_action. For more information, see
 Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"COMMIT PREPARED 'myptx';",[OR],"ROLLBACK PREPARED 'myptx';"}
```

Action suggérée

Comme indiqué dans la colonne de suggestions, connectez-vous à la base de données où se trouve la transaction préparée. Sur la base de la colonne `suggested_action`, examinez attentivement s'il faut exécuter `COMMIT` ou `ROLLBACK`, et quelle est l'action appropriée.

Pour surveiller les transactions préparées en général, PostgreSQL propose une vue de catalogue appelée `pg_prepared_xacts`. Vous pouvez utiliser la requête suivante pour rechercher les transactions préparées.

```
SELECT
    gid,
    prepared,
    owner,
    database,
    transaction AS oldest_xmin
FROM
    pg_prepared_xacts
ORDER BY
    age(transaction) DESC;
```

Emplacement de réplication logique

L'objectif d'un emplacement de réplication est de conserver les modifications non consommées jusqu'à ce qu'elles soient répliquées sur un serveur cible. Pour plus d'informations, consultez Réplication logique de PostgreSQL.

Il existe deux types d'emplacement de réplication logique.

Emplacements de réplication logique inactifs

Lorsque la réplication est interrompue, les journaux de transactions non consommés ne peuvent pas être supprimés et l'emplacement de réplication devient inactif. Même si un emplacement de réplication logique inactif n'est pas utilisé par un abonné, il reste sur le serveur, ce qui entraîne la conservation des fichiers WAL et empêche la suppression des anciens journaux de transactions. Cela peut augmenter l'utilisation du disque et empêcher spécifiquement l'autovacuum de nettoyer les tables de catalogue internes, car le système doit empêcher le remplacement des informations LSN. Si rien n'est fait, cela peut entraîner un gonflement du catalogue, une dégradation des performances et un risque accru de vacuum de bouclage, ce qui peut entraîner une durée d'indisponibilité des transactions.

Emplacements de réplication logique actifs mais lents

Parfois, la suppression des tuples inactifs du catalogue est retardée en raison de la dégradation des performances de la réplication logique. Ce retard de réplication ralentit la mise à jour de `catalog_xmin` et peut entraîner un gonflement du catalogue et un vacuum de bouclage.

La fonction `postgres_get_av_diag()` affiche une sortie similaire à la suivante lorsqu'elle trouve un emplacement de réplication logique en tant que bloqueur.

```
blocker               | Logical replication slot
```

```
database              | my_database
blocker_identifier    | slot1
wait_event            | Not applicable
autovacuum_lagging_by | 1,940,103,068
suggestion            | Ensure replication is active and resolve any lag for the slot
 if active. If inactive, consider dropping it using the command in suggested_action.
 For more information, see Working with PostgreSQL autovacuum in the Amazon RDS User
 Guide.
suggested_action      | {"SELECT pg_drop_replication_slot('slot1') FROM
 pg_replication_slots WHERE active = 'f';"}
```

Action suggérée

Pour résoudre ce problème, vérifiez la configuration de réplication afin de détecter tout problème lié au schéma cible ou aux données susceptibles de mettre fin au processus d'application. Voici les raisons les plus courantes de ce problème :

- Colonnes manquantes
- Types de données incompatibles
- Non-correspondance des données
- Table manquante

Si le problème est lié à des problèmes d'infrastructure :

- Problèmes de réseau : comment résoudre les problèmes liés à une base de données Amazon RDS dans un état réseau incompatible.
- La base de données ou l'instance de base de données n'est pas disponible pour les raisons suivantes :
  - L'instance de réplica n'a plus d'espace de stockage : consultez Instances de base de données Amazon RDS à court de stockage pour en savoir plus sur l'ajout d'espace de stockage.
  - Paramètres incompatibles : consultez Comment puis-je réparer une instance de base de données Amazon RDS bloquée avec le statut de paramètres incompatibles ? pour plus d'informations sur la façon dont vous pouvez résoudre le problème.

Si votre instance se trouve en dehors du réseau AWS ou sur AWS EC2, demandez à votre administrateur comment résoudre les problèmes liés à la disponibilité ou à l'infrastructure.

Suppression de l'emplacement inactif

> **Warning**
>
> Attention : avant de supprimer un emplacement de réplication, assurez-vous qu'il n'est pas en cours de réplication, qu'il est inactif et qu'il est dans un état irrécupérable. La suppression prématurée d'un emplacement peut perturber la réplication ou entraîner une perte de données.

Après avoir confirmé que l'emplacement de réplication n'est plus nécessaire, supprimez-le pour permettre à l'autovacuum de continuer. La condition `active = 'f'` garantit que seul un emplacement inactif est supprimé.

```
SELECT pg_drop_replication_slot('slot1') WHERE active ='f'
```

Instances de lecteur

Lorsque le paramètre hot_standby_feedback est activé, il empêche l'autovacuum sur l'instance d'enregistreur de supprimer les lignes inactives qui pourraient encore être nécessaires aux requêtes exécutées sur l'instance de lecteur. Ce comportement est nécessaire, car les requêtes exécutées sur l'instance du lecteur (également applicable aux instances du lecteur dans Aurora Global Database) nécessitent que ces lignes restent disponibles sur l'instance d'enregistreur, ce qui évite les conflits et les annulations de requêtes.

> **Note**
>
> `hot_standby_feedback` est activé par défaut et non modifiable dans Aurora PostgreSQL.

La fonction `postgres_get_av_diag()` affiche une sortie similaire à la suivante lorsqu'elle trouve un réplica en lecture avec un emplacement de réplication logique en tant que bloqueur.

```
blocker               | Oldest query running on aurora reader
database              | Not applicable
blocker_identifier    | my-aurora-reader-2
wait_event            | Not applicable
autovacuum_lagging_by | 540,122,859
suggestion            | Run the following query on the reader "my-aurora-reader-2" to
 find the long running query:
```

```
                      | SELECT * FROM pg_catalog.pg_stat_activity WHERE
 backend_xmin::text::bigint = 523476310;

                      | Review carefully and you may consider terminating the query on
 reader using suggested_action.
suggested_action      | {"SELECT pg_terminate_backend(pid) FROM
 pg_catalog.pg_stat_activity WHERE backend_xmin::text::bigint = 523476310;","

                      | [OR]


                      | ","Delete the reader if not needed"}
```

Comme recommandé dans la colonne `suggested_action`, examinez attentivement ces options pour débloquer l'autovacuum.

- Résilier la requête : en suivant les instructions de la colonne de suggestions, vous pouvez vous connecter au réplica en lecture, comme indiqué dans la colonne suggested_action. Il est conseillé de vérifier attentivement l'option permettant de résilier la session. Si la résiliation est considérée comme sûre, vous pouvez utiliser la fonction `pg_terminate_backend()` pour résilier la session. Cette action peut être effectuée par un administrateur (tel que le compte principal RDS) ou par un utilisateur disposant du privilège pg_terminate_backend() requis.

  Vous pouvez exécuter la commande SQL suivante sur le réplica en lecture pour résilier la requête qui empêche le vacuum sur la base de données principale de nettoyer les anciennes lignes. La valeur de `backend_xmin` est indiquée dans la sortie de la fonction :

  ```
  SELECT
      pg_terminate_backend(pid)
  FROM
      pg_catalog.pg_stat_activity
  WHERE
      backend_xmin::text::bigint = backend_xmin;
  ```

- Supprimer les instances de lecteur si elles ne sont pas nécessaires : si l'instance de lecteur n'est plus nécessaire, vous pouvez la supprimer. Cela supprime la surcharge de réplication associée et permet au système principal de recycler les journaux de transactions sans être freiné par l'instance.

Tables temporaires

Les tables temporaires, créées à l'aide du mot-clé `TEMPORARY`, résident dans le schéma temporaire, par exemple pg_temp_xxx, et ne sont accessibles qu'à la session qui les a créées. Les tables temporaires sont supprimées à la fin de la session. Cependant, ces tables sont invisibles pour le processus autovacuum de PostgreSQL et doivent être nettoyées manuellement par la session qui les a créées. Essayer d'effectuer un vacuum sur la table temporaire d'une autre session n'a aucun effet.

Dans des circonstances exceptionnelles, une table temporaire peut exister sans qu'elle appartienne à une session active. Si la session liée à une table temporaire se termine de façon inattendue en raison d'un incident fatal, d'un problème réseau ou d'un événement similaire, il est possible que cette table ne soit pas nettoyée et qu'elle devienne ainsi une table « orpheline ». Lorsque le processus autovacuum de PostgreSQL détecte une table temporaire orpheline, il consigne le message suivant :

```
LOG: autovacuum: found orphan temp table \"%s\".\"%s\" in database \"%s\"
```

La fonction `postgres_get_av_diag()` affiche une sortie similaire à la suivante lorsqu'elle identifie une table temporaire comme un bloqueur. Pour que cette fonction affiche correctement la sortie relative aux tables temporaires, elle doit être exécutée dans la base de données dans laquelle se trouvent ces tables.

```
blocker               | Temporary table
database              | my_database
blocker_identifier    | pg_temp_14.ttemp
wait_event            | Not applicable
autovacuum_lagging_by | 1,805,802,632
suggestion            | Connect to database "my_database". Review carefully, you
 may consider dropping temporary table using command in suggested_action. For more
 information, see Working with PostgreSQL autovacuum in the Amazon RDS User Guide.
suggested_action      | {"DROP TABLE ttemp;"}
```

Action suggérée

Suivez les instructions fournies dans la colonne `suggestion` de sortie pour identifier et supprimer la table temporaire qui empêche l'exécution de l'autovacuum. Utilisez la commande suivante pour supprimer la table temporaire signalée par `postgres_get_av_diag()`. Remplacez le nom de la table en fonction de la sortie fournie par la fonction `postgres_get_av_diag()`.

```
DROP TABLE my_temp_schema.my_temp_table;
```

La requête suivante peut être utilisée pour identifier les tables temporaires :

```
SELECT
    oid,
    relname,
    relnamespace::regnamespace,
    age(relfrozenxid)
FROM
    pg_class
WHERE
relpersistence = 't'
ORDER BY
    age(relfrozenxid) DESC;
```

Résolution des bloqueurs de vacuum non identifiables dans Aurora PostgreSQL

Cette section explore d'autres raisons qui peuvent empêcher l'opération de vacuum de progresser. Ces problèmes ne sont actuellement pas directement identifiables par la fonction `postgres_get_av_diag()`.

Rubriques

- Incohérence de l'index
- Taux de transaction exceptionnellement élevé

Incohérence de l'index

Un index dont la logique est incohérente peut empêcher l'autovacuum de progresser. Les erreurs suivantes ou des erreurs similaires sont journalisées pendant la phase de vacuum de l'index ou lorsque des instructions SQL accèdent à l'index.

```
ERROR: right sibling's left-link doesn't match:block 5 links to 10 instead of expected
 2 in index ix_name
```

```
ERROR: failed to re-find parent key in index "XXXXXXXXXX" for deletion target page XXX
CONTEXT:  while vacuuming index index_name of relation schema.table
```

Conseils

Reconstruisez l'index ou ignorez les index en utilisant `INDEX_CLEANUP` lors d'une opération `VACUUM FREEZE` manuelle.

- Utilisation de l'option CONCURRENTLY : avant PostgreSQL version 12, la reconstruction d'un index nécessitait un verrou de table exclusif, limitant ainsi l'accès à la table. Avec PostgreSQL versions 12 et ultérieures, l'option CONCURRENTLY permet le verrouillage au niveau des lignes, ce qui améliore considérablement la disponibilité de la table. Voici la commande :

  ```
  REINDEX INDEX ix_name CONCURRENTLY;
  ```

  Bien que l'option CONCURRENTLY soit moins perturbatrice, elle peut être plus lente sur les tables très occupées. Envisagez de générer l'index pendant les périodes de faible trafic si possible. Pour plus d'informations, consultez REINDEX dans la documentation de PostgreSQL.
- Utilisation de l'option INDEX_CLEANUP FALSE : si les index sont volumineux et que le temps estimé pour les finaliser est significatif, vous pouvez débloquer l'autovacuum en exécutant une opération VACUUM FREEZE manuelle tout en excluant les index. Cette fonctionnalité est disponible dans PostgreSQL versions 12 et ultérieures.

  Ignorer les index vous permet d'éviter le processus de vacuum de l'index incohérent et d'atténuer le problème de bouclage. Toutefois, cela ne résout pas le problème sous-jacent de page non valide. Pour résoudre complètement le problème de page non valide, vous devrez tout de même reconstruire l'index.

Taux de transaction exceptionnellement élevé

Dans PostgreSQL, les taux de transactions élevés peuvent avoir un impact significatif sur les performances de l'autovacuum, ce qui ralentit le nettoyage des tuples inactifs et augmente le risque de bouclage des ID de transaction. Vous pouvez surveiller le taux de transaction en mesurant la différence de `max(age(datfrozenxid))` entre deux périodes, généralement par seconde. En outre, vous pouvez utiliser les métriques de compteur suivantes de RDS Performance Insights pour mesurer le taux de transaction (somme de xact_commit et xact_rollback), qui est le nombre total de transactions.

Compteur	Type	Unité	Métrique
xact_commit	Transactions	Validations par seconde	db.Transactions.xact_commit
xact_rollback	Transactions	Restaurations par seconde	db.Transactions.xact_rollback

Une augmentation rapide indique une charge de transactions élevée, qui peut surcharger l'autovacuum, provoquant un gonflement, des conflits de verrouillage et des problèmes de performances potentiels. Cela peut avoir un impact négatif sur le processus d'autovacuum de plusieurs manières :

- Activité liée à la table : la table en question peut faire l'objet d'un volume élevé de transactions, ce qui peut entraîner des retards.
- Ressources système : l'ensemble du système est peut-être surchargé, ce qui rend difficile pour l'autovacuum d'accéder aux ressources nécessaires pour fonctionner efficacement.

Envisagez les stratégies suivantes pour permettre à l'autovacuum de fonctionner plus efficacement et de mener à bien ses tâches dans le délai imparti :

1. Réduisez le taux de transaction si possible. Envisagez de traiter par lots ou de regrouper les transactions similaires dans la mesure du possible.
2. Ciblez les tables fréquemment mises à jour avec une opération `VACUUM FREEZE` manuelle tous les soirs, toutes les semaines ou toutes les deux semaines pendant les heures creuses.
3. Envisagez d'augmenter verticalement votre classe d'instance pour allouer davantage de ressources système afin de gérer le volume élevé de transactions et l'autovacuum.

Résolution des problèmes de performance de vacuum dans Aurora PostgreSQL

Cette section décrit les facteurs qui contribuent souvent au ralentissement des performances de vacuum et explique comment résoudre ces problèmes.

Rubriques

- Vacuum des index de grande taille
- Trop de tables ou de bases de données pour le vacuum
- Un vacuum agressif (pour éviter tout bouclage) est en cours d'exécution

Vacuum des index de grande taille

VACUUM fonctionne selon des phases séquentielles : initialisation, analyse des tas, vacuum des index et des tas, nettoyage des index, troncature des tas et nettoyage final. Pendant l'analyse des tas, le processus élague les pages, les défragmente et les gèle. Une fois l'analyse des tas terminée, VACUUM nettoie les index, renvoie les pages vides au système d'exploitation et effectue

les dernières tâches de nettoyage, telles que le vacuum de la carte de l'espace libre et la mise à jour des statistiques.

L'opération de vacuum de l'index peut nécessiter plusieurs passages lorsque la capacité `maintenance_work_mem` (ou `autovacuum_work_mem`) est insuffisante pour traiter l'index. Dans PostgreSQL 16 et versions antérieures, une limite de mémoire de 1 Go pour le stockage des ID de tuple inactifs imposait souvent plusieurs passages sur les index volumineux. PostgreSQL 17 introduit `TidStore`, qui alloue de la mémoire de manière dynamique au lieu d'utiliser un tableau à allocation unique. Cela permet de supprimer la contrainte de 1 Go, d'utiliser la mémoire de manière plus efficace et d'éviter d'avoir à analyser plusieurs fois chaque index.

Les index de grande taille peuvent toujours nécessiter plusieurs passages dans PostgreSQL 17 si la mémoire disponible ne peut pas prendre en charge l'intégralité du traitement des index en une seule fois. Généralement, les index de grande taille contiennent davantage de tuples inactifs qui nécessitent plusieurs passages.

Détection des opérations de vacuum lentes

La fonction `postgres_get_av_diag()` peut détecter les opérations de vacuum qui sont lentes en raison d'une mémoire insuffisante. Pour plus d'informations sur cette fonction, consultez Installation d'outils de surveillance et de diagnostic d'autovacuum dans Aurora PostgreSQL.

La fonction `postgres_get_av_diag()` émet les notifications suivantes lorsque la mémoire disponible n'est pas suffisante pour terminer l'opération de vacuum de l'index en un seul passage.

`rds_tools` 1.9

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_work_mem is XX might not be sufficient.
 Consider increasing the setting to XXX, and if necessary, scaling up the RDS instance
 class for more
       memory. The suggested value is an estimate based on the current number of dead
 tuples for the table being vacuumed, which might not fully reflect the latest state.
 Additionally, review the possibility of manual
       vacuum with exclusion of indexes using (VACUUM (INDEX_CLEANUP FALSE, VERBOSE
 TRUE) table_name;). For more information, see

       Working with PostgreSQL autovacuum in the Amazon Aurora User Guide.
```

> **Note**
>
> La fonction `postgres_get_av_diag()` repose sur `pg_stat_all_tables.n_dead_tup` pour l'estimation de la quantité de mémoire requise pour l'opération de vacuum de l'index.

Lorsque la fonction `postgres_get_av_diag()` identifie une opération de vacuum lente qui nécessite plusieurs analyses d'index en raison d'une capacité `autovacuum_work_mem` insuffisante, elle génère le message suivant :

```
NOTICE: Your vacuum is performing multiple index scans due to insufficient
 autovacuum_work_mem:XXX for index vacuuming.
       For more information, see Working with PostgreSQL autovacuum in the Amazon
 Amazon RDS User Guide.
```

Conseils

Vous pouvez appliquer les solutions de contournement suivantes à l'aide d'une opération `VACUUM FREEZE` manuelle pour accélérer le gel de la table.

Augmentation de la mémoire pour l'opération de vacuum

Comme suggéré par la fonction `postgres_get_av_diag()`, il est conseillé d'augmenter le paramètre `autovacuum_work_mem` pour faire face aux contraintes de mémoire potentielles au niveau de l'instance. Même si `autovacuum_work_mem` est un paramètre dynamique, il est important de noter que pour que le nouveau paramètre de mémoire prenne effet, le démon autovacuum doit redémarrer ses applications de travail. Pour ce faire :

1. Vérifiez que le nouveau paramètre est en place.
2. Arrêtez les processus qui exécutent une opération d'autovacuum.

Cette approche garantit que l'allocation de mémoire ajustée est appliquée aux nouvelles opérations d'autovacuum.

Pour des résultats plus immédiats, pensez à effectuer une opération `VACUUM FREEZE` manuelle avec un paramètre `maintenance_work_mem` augmenté au cours de votre session :

```
SET maintenance_work_mem TO '1GB';
VACUUM FREEZE VERBOSE table_name;
```

Si vous utilisez Amazon RDS et que vous constatez que vous avez besoin de mémoire supplémentaire pour prendre en charge des valeurs plus élevées pour `maintenance_work_mem` ou `autovacuum_work_mem`, envisagez de passer à une classe d'instance avec plus de mémoire. Cela peut fournir les ressources nécessaires pour améliorer les opérations de vacuum manuelles et automatiques, ce qui se traduit par une amélioration des performances globales du vacuum et des bases de données.

Désactivation d'INDEX_CLEANUP

L'opération `VACUUM` manuelle de PostgreSQL 12 et versions ultérieures permet d'ignorer la phase de nettoyage de l'index, tandis que l'autovacuum d'urgence dans PostgreSQL 14 et versions ultérieures le fait automatiquement en fonction du paramètre `vacuum_failsafe_age`.

> **Warning**
>
> Ignorer le nettoyage de l'index peut entraîner un gonflement de l'index et avoir un impact négatif sur les performances des requêtes. Pour pallier ce problème, pensez à réindexer les index concernés ou à en effectuer le vacuum pendant une période de maintenance.

Pour obtenir des conseils supplémentaires sur la gestion des index de grande taille, reportez-vous à la documentation sur Gestion de la fonction autovacuum avec de grands index .

Opérations de vacuum d'index en parallèle

À partir de PostgreSQL 13, les index peuvent faire l'objet d'un vacuum et d'un nettoyage en parallèle par défaut avec une opération `VACUUM` manuelle. Un processus de travail de vacuum est alors assigné à chaque index. Cependant, pour que PostgreSQL puisse déterminer si une opération de vacuum peut être exécutée en parallèle, des critères spécifiques doivent être remplis :

- Il doit y avoir au moins deux index.
- Le paramètre `max_parallel_maintenance_workers` doit être défini sur au moins 2.
- La taille de l'index doit dépasser la limite de `min_parallel_index_scan_size`, qui est par défaut de 512 Ko.

Vous pouvez ajuster le paramètre `max_parallel_maintenance_workers` en fonction du nombre de processeurs virtuels disponibles sur votre instance Amazon RDS et du nombre d'index sur la table afin d'optimiser le délai d'exécution de l'opération de vacuum.

Pour plus d'informations, consultez Opérations de vacuum en parallèle dans Amazon RDS pour PostgreSQL et Amazon Aurora PostgreSQL.

Trop de tables ou de bases de données pour le vacuum

Comme indiqué dans la documentation The Autovacuum Daemon de PostgreSQL, le démon autovacuum fonctionne par le biais de plusieurs processus. Cela inclut un lanceur d'autovacuum permanent chargé de démarrer les processus de travail d'autovacuum pour chaque base de données du système. Le lanceur programme ces applications de travail pour qu'elles soient lancées environ toutes les `autovacuum_naptime` secondes par base de données.

Avec « N » bases de données, une nouvelle application de travail commence environ toutes les [`autovacuum_naptime`/N secondes]. Cependant, le nombre total d'applications de travail simultanés est limité par le paramètre `autovacuum_max_workers`. Si le nombre de bases de données ou de tables qui nécessitent un vacuum dépasse cette limite, la base de données ou table suivante est traitée dès qu'une application de travail est disponible.

Lorsque de nombreuses tables ou bases de données volumineuses nécessitent un vacuum simultané, toutes les applications de travail d'autovacuum disponibles peuvent être occupées pendant une période prolongée, ce qui retarde la maintenance des autres tables et bases de données. Dans les environnements où les taux de transactions sont élevés, ce goulot d'étranglement peut rapidement s'aggraver et potentiellement entraîner des problèmes de vacuum de bouclage au sein de votre instance Amazon RDS.

Lorsque `postgres_get_av_diag()` détecte un nombre élevé de tables ou de bases de données, il fournit la recommandation suivante :

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound and
 it might be slow.
```

```
NOTICE: The current setting of autovacuum_max_workers:3 might not be sufficient.
 Consider increasing the setting and, if necessary, consider scaling up the Amazon RDS
 instance class for more workers.
```

Conseils

Augmentation d'autovacuum_max_workers

Pour accélérer l'opération de vacuum, nous vous recommandons de régler le paramètre `autovacuum_max_workers` afin d'autoriser un plus grand nombre d'applications de travail

d'autovacuum simultanées. Si les problèmes de performances persistent, envisagez d'augmenter verticalement votre instance Amazon RDS vers une classe comportant davantage de processeurs virtuels, ce qui peut améliorer encore les capacités de traitement parallèle.

Un vacuum agressif (pour éviter tout bouclage) est en cours d'exécution

L'âge de la base de données (MaximumUsedTransactionIDs) dans PostgreSQL ne diminue que lorsqu'un vacuum agressif (pour empêcher tout bouclage) s'est terminé avec succès. Jusqu'à ce que ce vacuum se termine, l'âge continue d'augmenter en fonction du taux de transaction.

La fonction `postgres_get_av_diag()` génère le message `NOTICE` suivant lorsqu'elle détecte un vacuum agressif. Cependant, elle ne déclenche cette sortie qu'après que le vacuum a été actif pendant au moins deux minutes.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

Pour plus d'informations sur le vacuum agressif, consultez Lorsqu'un vacuum agressif est déjà en cours d'exécution.

Vous pouvez vérifier si un vacuum agressif est en cours à l'aide de la requête suivante :

```
SELECT
    a.xact_start AS start_time,
    v.datname "database",
    a.query,
    a.wait_event,
    v.pid,
    v.phase,
    v.relid::regclass,
    pg_size_pretty(pg_relation_size(v.relid)) AS heap_size,
    (
        SELECT
            string_agg(pg_size_pretty(pg_relation_size(i.indexrelid)) || ':' ||
 i.indexrelid::regclass || chr(10), ', ')
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS index_sizes,
    trunc(v.heap_blks_scanned * 100 / NULLIF(v.heap_blks_total, 0)) AS step1_scan_pct,
```

```
    v.index_vacuum_count || '/' || (
        SELECT
            count(*)
        FROM
            pg_index i
        WHERE
            i.indrelid = v.relid
    ) AS step2_vacuum_indexes,
    trunc(v.heap_blks_vacuumed * 100 / NULLIF(v.heap_blks_total, 0)) AS
 step3_vacuum_pct,
    age(CURRENT_TIMESTAMP, a.xact_start) AS total_time_spent_sofar
FROM
    pg_stat_activity a
    INNER JOIN pg_stat_progress_vacuum v ON v.pid = a.pid;
```

Vous pouvez déterminer s'il s'agit d'un vacuum agressif (pour éviter tout bouclage) en vérifiant la colonne de requête dans la sortie. L'expression « pour éviter tout bouclage » indique qu'il s'agit d'un vacuum agressif.

```
query                 | autovacuum: VACUUM public.t3 (to prevent wraparound)
```

Supposons, par exemple, que vous disposiez d'un bloqueur à l'âge d'un milliard de transactions et d'une table nécessitant un vacuum agressif pour éviter tout bouclage au même âge de transaction. De plus, il existe un autre bloqueur à l'âge de 750 millions de transactions. Après avoir éliminé le bloqueur à l'âge d'un milliard de transactions, l'âge des transactions ne tombera pas immédiatement à 750 millions. Il restera élevé jusqu'à ce que la table nécessitant un vacuum agressif ou toute transaction dont l'âge est supérieur à 750 millions soit finalisée. Au cours de cette période, l'âge des transactions de votre cluster PostgreSQL continuera d'augmenter. Une fois le processus de vacuum terminé, l'âge des transactions tombe à 750 millions, mais recommence à augmenter jusqu'à ce que l'opération de vacuum soit terminée. Ce cycle se poursuit tant que ces conditions persistent, jusqu'à ce que l'âge des transactions atteigne finalement le niveau configuré pour votre instance Amazon RDS, spécifié par `autovacuum_freeze_max_age`.

Explication des messages NOTICE dans Aurora PostgreSQL

La fonction `postgres_get_av_diag()` fournit les messages NOTICE suivants :

Lorsque l’âge n’a pas encore atteint le seuil de surveillance

Le seuil de surveillance `postgres_get_av_diag()` pour identifier les bloqueurs est de 500 millions de transactions par défaut. Si `postgres_get_av_diag()` génère le message NOTICE suivant, l’âge de la transaction n’a pas encore atteint ce seuil.

```
NOTICE: postgres_get_av_diag() checks for blockers that prevent aggressive vacuums
 only, it does so only after exceeding dvb_threshold which is 500,000,000 and age of
 this PostgreSQL cluster is currently at 2.
```

Non connecté à la base de données dont l’ID de transaction est le plus ancien

La fonction `postgres_get_av_diag()` fournit le résultat le plus précis lorsqu’elle est connectée à la base de données dont l’ID de transaction est le plus ancien. La base de données dont l’âge de l’ID de transaction est le plus ancien indiqué par `postgres_get_av_diag()` sera différente de « my_database » dans votre cas. Si vous n’êtes pas connecté à la bonne base de données, le message NOTICE suivant est généré :

```
NOTICE: You are not connected to the database with the age of oldest transaction
 ID. Connect to my_database database and run postgres_get_av_diag() for accurate
 reporting.
```

Il est important de se connecter à la base de données dont l’âge de la transaction est le plus ancien pour les raisons suivantes :

- Identification des bloqueurs de tables temporaires : les métadonnées des tables temporaires étant spécifiques à chaque base de données, elles se trouvent généralement dans la base de données dans laquelle elles ont été créées. Toutefois, si une table temporaire se trouve être le principal bloqueur et qu’elle se trouve dans la base de données contenant la transaction la plus ancienne, cela peut prêter à confusion. La connexion à la base de données appropriée garantit l’identification précise du bloqueur de table temporaire.
- Diagnostic des vacuums lents : les métadonnées d’index et les informations relatives au nombre de tables sont spécifiques à chaque base de données et sont nécessaires pour diagnostiquer les problèmes de lenteur du vacuum.

La base de données dont l’âge de la transaction est le plus ancien se trouve sur une base de données rdsadmin ou template0

Dans certains cas, les bases de données `rdsadmin` ou `template0` peuvent être identifiées comme étant celles dont l’ID de transaction est le plus ancien. Dans ce cas, `postgres_get_av_diag()` émet le message NOTICE suivant :

```
NOTICE: The database with the age of oldest transaction ID is rdsadmin or template0,
 reach out to support if the reported blocker is in rdsadmin or template0.
```

Vérifiez que le bloqueur répertorié ne provient d'aucune de ces deux bases de données. Si la présence du bloqueur est signalée dans `rdsadmin` ou `template0`, contactez le support, car ces bases de données ne sont pas accessibles aux utilisateurs et nécessitent une intervention.

Il est très peu probable que la base de données `rdsadmin` ou `template0` contienne un bloqueur principal.

Lorsqu'un vacuum agressif est déjà en cours d'exécution

La fonction `postgres_get_av_diag()` est conçue pour signaler lorsqu'un processus de vacuum agressif est en cours d'exécution, mais elle ne déclenche cette sortie que lorsque le vacuum est actif depuis au moins 1 minute. Ce délai intentionnel contribue à réduire les risques de faux positifs. En attendant, la fonction garantit que seuls les vacuums efficaces et significatifs sont signalés, ce qui permet une surveillance plus précise et plus fiable de l'activité du vacuum.

La fonction `postgres_get_av_diag()` génère le message NOTICE suivant lorsqu'elle détecte un ou plusieurs vacuums agressifs en cours.

```
NOTICE: Your database is currently running aggressive vacuum to prevent wraparound,
 monitor autovacuum performance.
```

Comme indiqué dans le message NOTICE, continuez à surveiller les performances du vacuum. Pour plus d'informations sur le vacuum agressif, consultez Un vacuum agressif (pour éviter tout bouclage) est en cours d'exécution

Lorsque l'autovacuum est désactivé

La fonction `postgres_get_av_diag()` génère le message NOTICE suivant si l'autovacuum est désactivé sur votre instance de base de données :

```
NOTICE: Autovacuum is OFF, we strongly recommend to enable it, no restart is
 necessary.
```

L'autovacuum est une fonctionnalité essentielle de votre instance de base de données Aurora PostgreSQL. Elle garantit le bon fonctionnement de la base de données. Elle supprime automatiquement les anciennes versions de lignes, récupère de l'espace de stockage et évite le gonflement des tables, ce qui contribue à garantir l'efficacité des tables et des index pour

des performances optimales. En outre, elle protège contre le bouclage des ID de transaction, qui peut interrompre les transactions sur votre instance Amazon RDS. La désactivation de l'autovacuum peut entraîner une baisse à long terme des performances et de la stabilité de la base de données. Nous vous conseillons de laisser cette fonctionnalité activée en permanence. Pour plus d'informations, consultez Présentation d'autovacuum dans les environnements Aurora PostgreSQL.

> **Note**
>
> La désactivation de l'autovacuum n'arrête pas les vacuums agressifs. Ils se produiront toujours une fois que vos tables auront atteint le seuil `autovacuum_freeze_max_age`.

Le nombre de transactions restantes est extrêmement faible

La fonction `postgres_get_av_diag()` génère le message NOTICE suivant lorsqu'un vacuum de bouclage est imminent. Ce message NOTICE est émis lorsque votre instance Amazon RDS est à 100 millions de transactions avant que de nouvelles transactions ne soient potentiellement rejetées.

```
WARNING: Number of transactions remaining is critically low, resolve issues with
 autovacuum or perform manual VACUUM FREEZE before your instance stops accepting
 transactions.
```

Une action immédiate est nécessaire pour éviter toute durée d'indisponibilité de la base de données. Vous devez surveiller de près vos opérations de vacuum et envisager de lancer manuellement une opération `VACUUM FREEZE` sur la base de données concernée afin d'éviter les échecs de transaction.

Gestion de la contention des OID TOAST dans

TOAST (The Oversized-Attribute Storage Technique) est une fonctionnalité de PostgreSQL conçue pour gérer de grandes valeurs de données dépassant la taille de bloc de base de données habituelle de 8 Ko. PostgreSQL n'autorise pas les lignes physiques à s'étendre sur plusieurs blocs. La taille du bloc agit comme une limite supérieure de la taille des lignes. TOAST surmonte cette restriction en divisant les grandes valeurs de champs en plus petits morceaux. Il les stocke séparément dans une table TOAST dédiée liée à la table principale. Pour plus d'informations, consultez le mécanisme de stockage TOAST de PostgreSQL et la documentation d'implémentation.

Rubriques

Comprendre les opérations de TOAST

TOAST effectue la compression et stocke de grandes valeurs de champ hors ligne. TOAST attribue un OID (identifiant d'objet) unique à chaque bloc de données surdimensionné stocké dans la table TOAST. La table principale stocke l'ID de valeur TOAST et l'ID de relation sur la page pour référencer la ligne correspondante dans la table TOAST. Cela permet à PostgreSQL de localiser et de gérer efficacement ces fragments TOAST. Cependant, au fur et à mesure que la table TOAST s'allonge, le système risque d'épuiser le stock disponible OIDs, ce qui entraîne à la fois une dégradation des performances et des temps d'arrêt potentiels dus à l'épuisement de l'OID.

Identifiants d'objets dans TOAST

Un identifiant d'objet (OID) est un identifiant unique à l'échelle du système utilisé par PostgreSQL pour référencer des objets de base de données tels que des tables, des index et des fonctions. Ces identifiants jouent un rôle essentiel dans les opérations internes de PostgreSQL, car ils permettent à la base de données de localiser et de gérer efficacement les objets.

Pour les tables contenant des ensembles de données éligibles au toasting, PostgreSQL OIDs attribue une identification unique à chaque bloc de données surdimensionné stocké dans la table TOAST associée. Le système associe chaque fragment à `unchunk_id`, ce qui permet à PostgreSQL d'organiser et de localiser efficacement ces fragments dans la table TOAST.

Identifier les défis en matière de performance

La gestion des OID de PostgreSQL repose sur un compteur global de 32 bits, de sorte qu'il fonctionne après avoir généré 4 milliards de valeurs uniques. Bien que le cluster de base de données partage ce compteur, l'allocation d'OID implique deux étapes lors des opérations TOAST :

- Compteur global pour l'allocation : le compteur global attribue un nouvel OID à l'ensemble du cluster.

- Recherche locale de conflits — La table TOAST garantit que le nouvel OID n'entre pas en conflit avec l'OID OIDs déjà utilisé dans cette table spécifique.

Une dégradation des performances peut se produire lorsque :

- La table TOAST présente une fragmentation élevée ou une utilisation dense des OID, ce qui entraîne des retards dans l'attribution de l'OID.
- Le système alloue et réutilise fréquemment des données OIDs dans des environnements présentant un taux de perte de données élevé ou de larges tables utilisant largement TOAST.

Pour plus d'informations, consultez la documentation relative aux limites de taille des tables TOAST de PostgreSQL et à l'allocation d'OID :

Un compteur global génère OIDs et enveloppe environ 4 milliards de valeurs, de sorte que le système génère à nouveau de temps en temps une valeur déjà utilisée. PostgreSQL le détecte et réessaie avec l'OID suivant. Un INSERT lent peut se produire s'il y a une très longue série de valeurs OID utilisées sans interruption dans la table TOAST. Ces difficultés s'accentuent à mesure que l'espace OID se remplit, ce qui ralentit les insertions et les mises à jour.

Identifier le problème

- `INSERT`Les déclarations simples prennent beaucoup plus de temps que d'habitude de manière incohérente et aléatoire.
- Les retards se produisent uniquement pour `INSERT` les `UPDATE` déclarations impliquant des opérations TOAST.
- Les entrées de journal suivantes apparaissent dans les journaux PostgreSQL lorsque le système peine à les trouver dans les tables TOAST OIDs :

```
LOG: still searching for an unused OID in relation "pg_toast_20815"
DETAIL: OID candidates have been checked 1000000 times, but no unused OID has been
 found yet.
```

- Performance Insights indique un nombre élevé de sessions actives (AAS) moyennes associées à des événements `LWLock:buffer_io` et à des événements `LWLock:OidGenLock` d'attente.

  Vous pouvez exécuter la requête SQL suivante pour identifier les transactions INSERT de longue durée associées à des événements d'attente :

```
SELECT
    datname AS database_name,
    usename AS database_user,
    pid,
    now() - pg_stat_activity.xact_start AS transaction_duration,
    concat(wait_event_type, ':', wait_event) AS wait_event,
    substr(query, 1, 30) AS TRANSACTION,
    state
FROM
    pg_stat_activity
WHERE (now() - pg_stat_activity.xact_start) > INTERVAL '60 seconds'
    AND state IN ('active', 'idle in transaction', 'idle in transaction (aborted)',
 'fastpath function call', 'disabled')
    AND pid <> pg_backend_pid()
AND lower(query) LIKE '%insert%'
ORDER BY
    transaction_duration DESC;
```

Exemples de résultats de requête affichant des opérations INSERT avec des temps d'attente prolongés :

```
 database_name |  database_user  |  pid  | transaction_duration |     wait_event
 |          transaction           | state
---------------+-----------------+-------+----------------------
+---------------------+---------------------------------+--------
 postgres       | db_admin_user| 70965 | 00:10:19.484061      | LWLock:buffer_io    |
 INSERT INTO "products" (......... | active
 postgres       | db_admin_user| 69878 | 00:06:14.976037      | LWLock:buffer_io    |
 INSERT INTO "products" (......... | active
 postgres       | db_admin_user| 68937 | 00:05:13.942847      | :                   |
 INSERT INTO "products" (......... | active
```

Isoler le problème

- Test small insert — Insérez un enregistrement inférieur au `toast_tuple_target` seuil. N'oubliez pas que la compression est appliquée avant le stockage de TOAST. Si cela fonctionne sans problèmes de performances, le problème est lié aux opérations TOAST.

- Tester une nouvelle table — Créez une nouvelle table avec la même structure et insérez un enregistrement supérieur à`toast_tuple_target`. Si cela fonctionne sans problème, le problème est localisé dans l'allocation OID de la table d'origine.

Recommandations

Les approches suivantes peuvent aider à résoudre les problèmes de contention des OID TOAST.

- Nettoyage et archivage des données : passez en revue et supprimez toutes les données obsolètes ou inutiles afin de OIDs les libérer pour une utilisation future, ou archivez-les. Prenez en compte les restrictions suivantes :
  - Évolutivité limitée, car le nettoyage futur ne sera peut-être pas toujours possible.
  - Possibilité d'une opération VACUUM de longue durée pour éliminer les tuples morts qui en résultent.
- Écrire dans une nouvelle table : créez une nouvelle table pour les futures insertions et utilisez une `UNION ALL` vue pour combiner les anciennes et les nouvelles données pour les requêtes. Cette vue présente les données combinées des anciennes et des nouvelles tables, ce qui permet aux requêtes d'y accéder sous la forme d'une seule table. Prenez en compte les restrictions suivantes :
  - Les mises à jour de l'ancienne table risquent tout de même d'entraîner l'épuisement de l'OID.
- Partition ou partition : partitionnez les données de la table ou de la partition pour améliorer l'évolutivité et les performances. Prenez en compte les restrictions suivantes :
  - Complexité accrue de la logique des requêtes et de la maintenance, nécessité potentielle de modifier les applications pour gérer correctement les données partitionnées.

Contrôle

Utilisation des tables du système

Vous pouvez utiliser les tables système de PostgreSQL pour surveiller l'augmentation de l'utilisation des OID.

> **Warning**
>
> Selon le nombre de personnes figurant OIDs dans le tableau TOAST, le processus peut prendre un certain temps. Nous vous recommandons de planifier la surveillance en dehors des heures de bureau afin de minimiser l'impact.

Le bloc anonyme suivant compte le nombre de caractères distincts OIDs utilisés dans chaque table TOAST et affiche les informations de la table parent :

```
DO $$
DECLARE
    r record;
    o bigint;
    parent_table text;
    parent_schema text;
BEGIN
    SET LOCAL client_min_messages TO notice;
    FOR r IN
    SELECT
        c.oid,
        c.oid::regclass AS toast_table
    FROM
        pg_class c
    WHERE
        c.relkind = 't'
        AND c.relowner != 10 LOOP
            -- Fetch the number of distinct used OIDs (chunk IDs) from the TOAST table
            EXECUTE 'SELECT COUNT(DISTINCT chunk_id) FROM ' || r.toast_table INTO o;
            -- If there are used OIDs, find the associated parent table and its schema
            IF o <> 0 THEN
                SELECT
                    n.nspname,
                    c.relname INTO parent_schema,
                    parent_table
                FROM
                    pg_class c
                    JOIN pg_namespace n ON c.relnamespace = n.oid
                WHERE
                    c.reltoastrelid = r.oid;
                -- Raise a concise NOTICE message
```

```
                RAISE NOTICE 'Parent schema: % | Parent table: % | Toast table: %
 | Number of used OIDs: %', parent_schema, parent_table, r.toast_table, TO_CHAR(o,
 'FM9,999,999,999,999');
            END IF;
        END LOOP;
END
$$;
```

Exemple de sortie affichant les statistiques d'utilisation des OID par table TOAST :

```
NOTICE:  Parent schema: public | Parent table: my_table | Toast table:
 pg_toast.pg_toast_16559 | Number of used OIDs: 45,623,317
NOTICE:  Parent schema: public | Parent table: my_table1 | Toast table:
 pg_toast.pg_toast_45639925 | Number of used OIDs: 10,000
NOTICE:  Parent schema: public | Parent table: my_table2 | Toast table:
 pg_toast.pg_toast_45649931 | Number of used OIDs: 1,000,000
DO
```

Le bloc anonyme suivant récupère l'OID maximal attribué pour chaque table TOAST non vide :

```
DO $$
DECLARE
    r record;
    o bigint;
    parent_table text;
    parent_schema text;
BEGIN
    SET LOCAL client_min_messages TO notice;
    FOR r IN
    SELECT
        c.oid,
        c.oid::regclass AS toast_table
    FROM
        pg_class c
    WHERE
        c.relkind = 't'
        AND c.relowner != 10 LOOP
            -- Fetch the max(chunk_id) from the TOAST table
            EXECUTE 'SELECT max(chunk_id) FROM ' || r.toast_table INTO o;
            -- If there's at least one TOASTed chunk, find the associated parent table
 and its schema
            IF o IS NOT NULL THEN
                SELECT
```

```
                    n.nspname,
                    c.relname INTO parent_schema,
                    parent_table
                FROM
                    pg_class c
                    JOIN pg_namespace n ON c.relnamespace = n.oid
                WHERE
                    c.reltoastrelid = r.oid;
                -- Raise a concise NOTICE message
                RAISE NOTICE 'Parent schema: % | Parent table: % | Toast table:
 % | Max chunk_id: %', parent_schema, parent_table, r.toast_table, TO_CHAR(o,
 'FM9,999,999,999,999');
            END IF;
        END LOOP;
END
$$;
```

Exemple de sortie affichant la partie maximale IDs pour les tables TOAST :

```
NOTICE:  Parent schema: public | Parent table: my_table | Toast table:
 pg_toast.pg_toast_16559 | Max chunk_id: 45,639,907
NOTICE:  Parent schema: public | Parent table: my_table1 | Toast table:
 pg_toast.pg_toast_45639925 | Max chunk_id: 45,649,929
NOTICE:  Parent schema: public | Parent table: my_table2 | Toast table:
 pg_toast.pg_toast_45649931 | Max chunk_id: 46,649,935
DO
```

Utilisation de l'Analyse des performances

Les événements `LWLock:buffer_io` d'attente `LWLock:OidGenLock` apparaissent dans Performance Insights lors d'opérations nécessitant l'attribution de nouveaux identifiants d'objet ()OIDs. Les sessions actives à moyenne élevée (AAS) associées à ces événements indiquent généralement des conflits lors de l'attribution des OID et de la gestion des ressources. Cela est particulièrement courant dans les environnements caractérisés par une perte de données élevée, une utilisation importante de données ou une création d'objets fréquente.

LWLock:buffer_io

`LWLock:buffer_io`est un événement d'attente qui se produit lorsqu'une session PostgreSQL attend la fin I/O des opérations sur une mémoire tampon partagée. Cela se produit généralement

lorsque la base de données lit les données du disque dans la mémoire ou écrit des pages modifiées de la mémoire sur le disque. L'événement `BufferIO` d'attente garantit la cohérence en empêchant plusieurs processus d'accéder à la même mémoire tampon ou de la modifier pendant que les I/O opérations sont en cours. La fréquence élevée de cet événement d'attente peut indiquer un engorgement du disque ou une I/O activité excessive de la charge de travail de la base de données.

Pendant les opérations TOAST :

- PostgreSQL OIDs alloue des fonds aux objets volumineux et garantit leur unicité en analysant l'index de la table TOAST.
- Les grands index TOAST peuvent nécessiter l'accès à plusieurs pages pour vérifier l'unicité de l'OID. Cela entraîne une augmentation des E/S sur le disque, en particulier lorsque le pool de mémoire tampon ne peut pas mettre en cache toutes les pages requises.

La taille de l'index influe directement sur le nombre de pages tampon auxquelles il est nécessaire d'accéder au cours de ces opérations. Même si l'indice n'est pas gonflé, sa taille même peut augmenter les E/S de la mémoire tampon, en particulier dans les environnements à forte concurrence ou à taux de désabonnement élevé. Pour plus d'informations, consultez : Guide de résolution des problèmes liés aux événements d'LWLockattente BufferIO.

LWLock:OidGenLock

`OidGenLock`est un événement d'attente qui se produit lorsqu'une session PostgreSQL attend d'allouer un nouvel identifiant d'objet (OID). Ce verrou garantit qu' OIDs ils sont générés de manière séquentielle et sûre, en ne permettant de générer qu'un seul processus OIDs à la fois.

Pendant les opérations TOAST :

- Allocation d'OID pour les segments dans les tables TOAST : PostgreSQL les OIDs affecte aux segments des tables TOAST lors de la gestion d'enregistrements de données volumineux. Chaque OID doit être unique pour éviter les conflits dans le catalogue système.
- Haute simultanéité : l'accès au générateur d'OID étant séquentiel, des conflits peuvent survenir lorsque plusieurs sessions créent simultanément des objets qui en ont besoin OIDs. `OidGenLock` Cela augmente la probabilité que les sessions attendent la fin de l'allocation des OID.
- Dépendance à l'égard de l'accès au catalogue du système : l'allocation OIDs nécessite des mises à jour des tables de catalogue système partagées, telles que `pg_class` et`pg_type`. Si ces tables sont soumises à une activité intense (en raison de fréquentes opérations DDL), cela peut augmenter le nombre de conflits de verrouillage pour. `OidGenLock`

- Demande d'allocation d'OID élevée — Les charges de travail lourdes de TOAST associées à des enregistrements de données volumineux nécessitent une allocation d'OID constante, ce qui augmente les contentions.

Facteurs supplémentaires qui augmentent la contention des OID :

- Création fréquente d'objets : les charges de travail qui créent et suppriment fréquemment des objets, tels que des tables temporaires, amplifient la contention sur le compteur d'OID global.
- Verrouillage global du compteur : le compteur OID global est accessible en série pour garantir l'unicité, créant ainsi un point de conflit unique dans les environnements à forte simultanéité.

Utilisation de Babelfish for Aurora PostgreSQL

Babelfish pour Aurora PostgreSQL étend votre cluster de bases de données Aurora PostgreSQL en permettant d'accepter les connexions de base de données à partir de clients SQL Server. Avec Babelfish, les applications qui étaient initialement conçues pour SQL Server peuvent être utilisées directement avec Aurora PostgreSQL, sans apporter de changements significatifs au code par rapport à une migration traditionnelle et sans modifier les pilotes de base de données. Pour en savoir plus sur la migration, consultez Migration d'une base de données SQL Server vers Babelfish pour Aurora PostgreSQL.

Babelfish fournit un point de terminaison supplémentaire pour un cluster de bases de données Aurora PostgreSQL, ce qui lui permet de comprendre le protocole de niveau filaire SQL Server et les instructions SQL Server couramment utilisées. Les applications clientes qui utilisent le protocole filaire TDS (Tabular Data Stream) peuvent se connecter de manière native au port d'écoute TDS sur Aurora PostgreSQL. Pour en savoir plus sur TDS, consultez [MS-TDS]: Tabular Data Stream Protocol ([MS-TDS] : protocole de flux de données tabulaires) sur le site Web de Microsoft.

> **Note**
>
> Babelfish for Aurora PostgreSQL prend en charge les versions 7.1 à 7.4 de TDS.

Babelfish permet également d'accéder aux données à l'aide de la connexion PostgreSQL. Par défaut, les deux langages SQL pris en charge par Babelfish sont disponibles via leurs protocoles filaires natifs sur les ports suivants :

- Langage SQL Server (T-SQL) : les clients se connectent au port 1433.
- Langage PostgreSQL (PL/pgSQL) : les clients se connectent au port 5432.

Babelfish exécute le langage Transact-SQL (T-SQL) avec quelques différences. Pour de plus amples informations, consultez Différences entre Babelfish pour Aurora PostgreSQL et SQL Server.

Les sections suivantes fournissent des informations sur la configuration et l'utilisation d'un cluster de bases de données Babelfish for Aurora PostgreSQL.

Rubriques

- Limitations Babelfish
- Analyse de l'architecture et de la configuration de Babelfish

Limitations Babelfish

Les limitations suivantes s’appliquent actuellement à Babelfish pour Aurora PostgreSQL :

- Babelfish ne prend pas en charge les fonctionnalités Aurora suivantes :
  - Gestion des identités et des accès AWS
  - Flux d’activité des bases de données (DAS)
  - API de données RDS avec Aurora PostgreSQL sans serveur v2 et provisionné
  - RDS Proxy avec RDS for SQL Server
  - Mécanisme d’authentification SCRAM
  - Éditeur de requête
- Babelfish ne fournit pas le support d’API de pilote client suivant :
  - Les requêtes d’API ayant des attributs de connexion liés à Microsoft Distributed Transaction Coordinator (MSDTC) ne sont pas prises en charge. Il s'agit notamment des appels XA effectués par la SQLServer XAResource classe dans le pilote JDBC de SQL Server.
- Actuellement, Babelfish ne prend pas en charge les extensions Aurora PostgreSQL suivantes :
  - `bloom`
  - `btree_gin`
  - `btree_gist`
  - `citext`
  - `cube`
  - `hstore`
  - `hypopg`
  - Réplication logique avec `pglogical`
  - `ltree`
  - `pgcrypto`
  - Gestion des plans de requêtes avec `apg_plan_mgmt`

  Pour en savoir plus sur les extensions PostgreSQL, consultez Utilisation d'extensions avec encapsuleurs de données externes.
- Le pilote open source jTDS, conçu comme une alternative au pilote JDBC de Microsoft, n’est pas pris en charge.

Analyse de l'architecture et de la configuration de Babelfish

Vous gérez le cluster de bases de données Aurora PostgreSQL-Compatible Edition exécutant Babelfish comme n'importe quel cluster de bases de données Aurora. En d'autres termes, vous bénéficiez de la capacité de mise à l'échelle, de la haute disponibilité avec prise en charge du basculement et de la réplication intégrée d'un cluster de bases de données Aurora. Pour en savoir plus sur ces fonctionnalités, consultez Gestion des performances et dimensionnement des clusters de bases de données Aurora, Haute disponibilité pour Amazon Aurora et Réplication avec Amazon Aurora. Vous avez également accès à de nombreux autres outils et utilitaires AWS, y compris les suivants :

- Amazon CloudWatch est un service de surveillance et d'observabilité qui vous fournit des données et des informations exploitables. Pour de plus amples informations, consultez Surveillance des métriques Amazon Aurora avec Amazon CloudWatch.
- Performance Insights est une fonction de réglage et de surveillance des performances de la base de données qui vous aide à évaluer rapidement la charge sur votre base de données. Pour en savoir plus, veuillez consulter la section Surveillance de la charge de la base de données avec Performance Insights sur .
- Les bases de données globales Aurora couvrent plusieurs Régions AWS, permettant des lectures globales à faible latence, et assurent une reprise rapide de toute interruption susceptible d'affecter une Région AWS entière. Pour de plus amples informations, consultez Utilisation d'Amazon Aurora Global Database.
- L'application automatique de correctifs logiciels maintient votre base de données à jour avec les derniers correctifs de sécurité et de fonctions dès qu'ils sont disponibles.
- Les événements Amazon RDS vous informent par e-mail ou par SMS des événements importants relatifs à la base de données, comme les basculements automatiques. Pour de plus amples informations, consultez Surveillance des événements Amazon Aurora.

Vous trouverez ci-dessous des informations sur l'architecture de Babelfish et sur la façon dont les bases de données SQL Server que vous migrez sont gérées par Babelfish. Lorsque vous créez votre cluster de bases de données Babelfish, vous devez prendre des décisions à l'avance concernant une base de données unique ou plusieurs bases de données, les classements et d'autres détails.

Rubriques

- Architecture Babelfish
- Paramètres du groupe de paramètres de cluster de bases de données pour Babelfish

- Comprendre les classements dans Babelfish pour Aurora PostgreSQL.
- Gestion du traitement des erreurs Babelfish avec des trappes de secours

Architecture Babelfish

Lorsque vous créez un cluster Aurora PostgreSQL dans lequel Babelfish est activé, Aurora approvisionne ce cluster avec une base de données PostgreSQL nommée `babelfish_db`. Cette base de données héberge l'ensemble des objets et structures SQL Server migrés.

> **Note**
>
> Dans un cluster Aurora PostgreSQL, le nom de base de données `babelfish_db` est réservé à Babelfish. La création de votre propre base de données « babelfish_db » sur un cluster de bases de données Babelfish empêche Aurora d'allouer Babelfish.

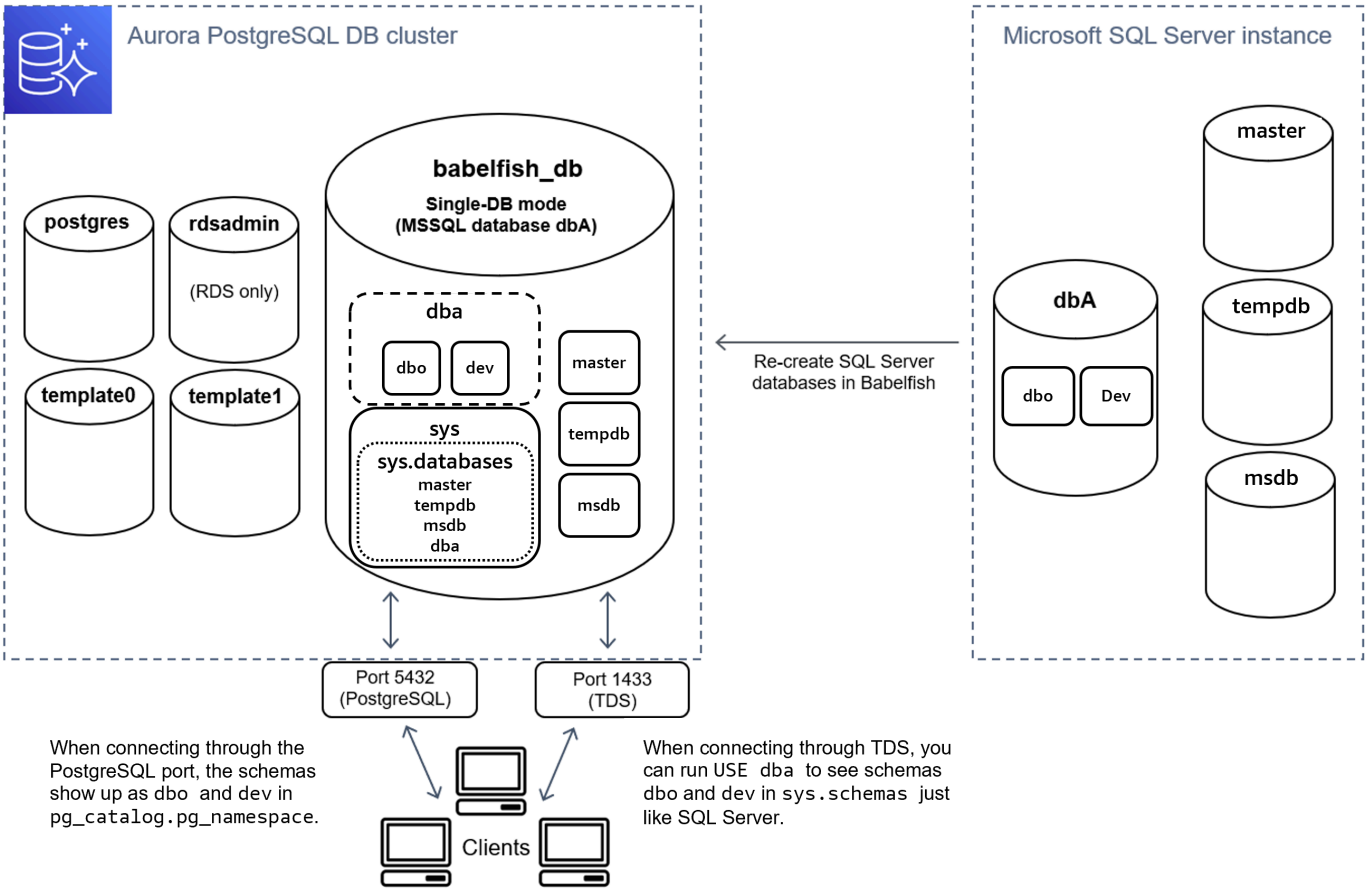
Lorsque vous vous connectez au port TDS, la session est placée dans la base de données `babelfish_db`. Depuis T-SQL, la structure ressemble à celle d'une connexion à une instance SQL Server. Vous pouvez consulter les bases de données `master`, `msdb` et `tempdb` ainsi que le catalogue `sys.databases`. Vous pouvez créer des bases de données utilisateur supplémentaires et passer d'une base de données à l'autre à l'aide de l'instruction USE. Lorsque vous créez une base de données utilisateur SQL Server, celle-ci est mise à plat dans la base de données PostgreSQL `babelfish_db`. Votre base de données conserve une syntaxe et une sémantique inter-bases de données identiques ou semblables à celles fournies par SQL Server.

Utilisation de Babelfish avec une ou plusieurs bases de données

Lorsque vous créez un cluster Aurora PostgreSQL à utiliser avec Babelfish, vous pouvez choisir d'utiliser une base de données SQL Server individuellement ou plusieurs bases de données SQL Server ensemble. Votre choix détermine la façon dont les noms des schémas SQL Server contenus dans la base de données `babelfish_db` apparaissent à partir d'Aurora PostgreSQL. Le mode de migration est stocké dans le paramètre `migration_mode`. Vous ne devez pas modifier ce paramètre après avoir créé votre cluster car vous pourriez perdre l'accès à tous vos objets SQL précédemment créés.

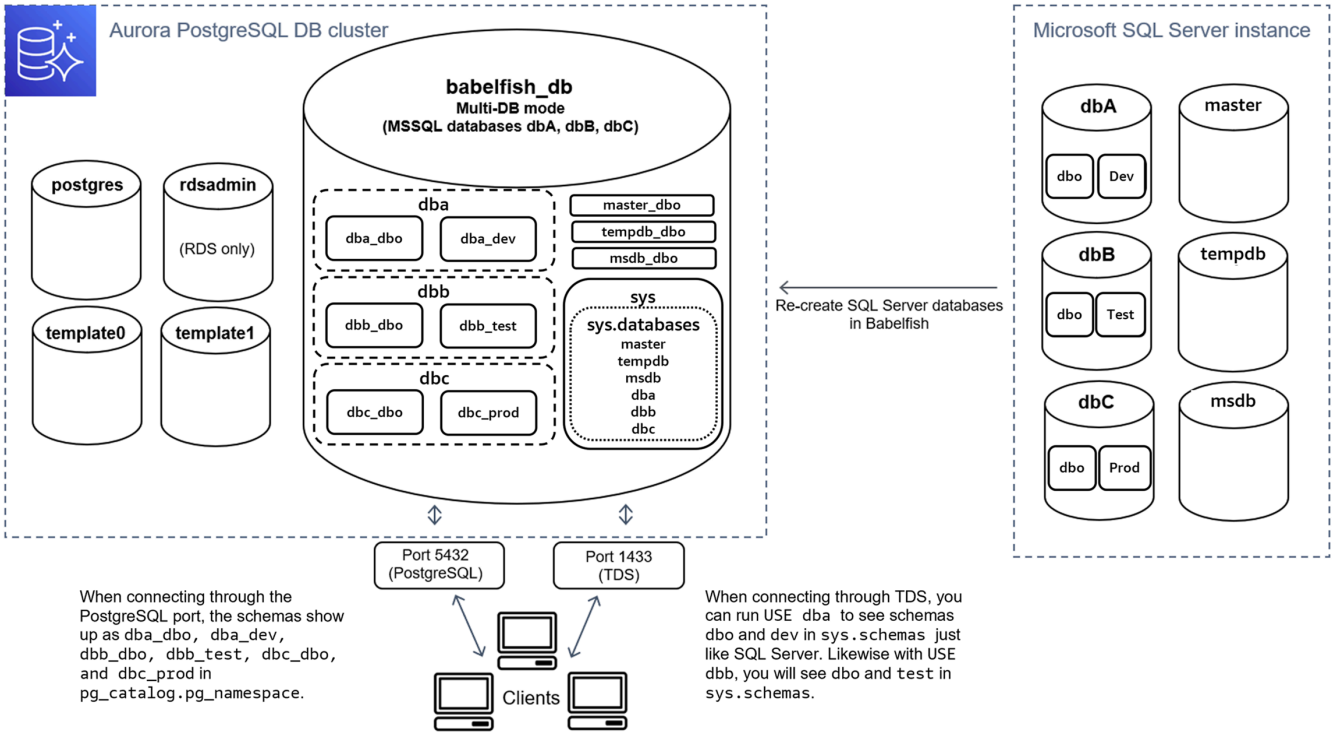
En mode single-db, les noms des schémas de la base de données SQL Server restent les mêmes dans la base de données `babelfish_db` de PostgreSQL. Si vous choisissez de ne migrer qu'une

seule base de données, les noms de schémas de la base de données utilisateur migrée peuvent être référencés dans PostgreSQL avec les mêmes noms que ceux utilisés dans SQL Server. Par exemple, les schémas dbo et smith résident dans la base de données dbA.



Lorsque vous vous connectez via TDS, vous pouvez exécuter USE dba pour voir les schémas dbo et dev depuis T-SQL, comme vous le feriez dans SQL Server. Les noms de schémas inchangés sont également visibles depuis PostgreSQL.

En mode à plusieurs bases de données, les noms de schémas des bases de données utilisateur deviennent dbname_schemaname lorsqu'ils font l'objet d'un accès depuis PostgreSQL. Les noms de schémas restent les mêmes lorsqu'ils font l'objet d'un accès depuis T-SQL.



Comme le montre l'image, le mode à une base de données et le mode à plusieurs bases de données sont identiques à la procédure de connexion via le port TDS et d'utilisation de T-SQL dans SQL Server. Par exemple, `USE dbA` répertorie les schémas dbo et `dev` comme dans SQL Server. Les noms de schémas mappés, tels que `dba_dbo` et `dba_dev`, sont visibles depuis PostgreSQL.

Vos schémas sont toujours contenus dans chacune des bases de données. Le nom de chaque base de données précède le nom du schéma SQL Server, avec un trait de soulignement comme délimiteur. Par exemple :

- `dba` contient `dba_dbo` et `dba_dev`.
- `dbb` contient `dbb_dbo` et `dbb_test`.
- `dbc` contient `dbc_dbo` et `dbc_prod`.

Dans la base de données `babelfish_db`, l'utilisateur T-SQL doit toujours exécuter `USE dbname` pour modifier le contexte de la base de données, afin que la présentation reste semblable à celle de SQL Server.

Choix d'un mode de migration

Chaque mode de migration présente des avantages et des inconvénients. Choisissez votre mode de migration en fonction du nombre de bases de données utilisateur dont vous disposez et de vos

plans de migration. Après avoir créé un cluster à utiliser avec Babelfish, vous ne devez pas changer le mode de migration car vous pourriez perdre l'accès à tous vos objets SQL précédemment créés. Lorsque vous choisissez un mode de migration, tenez compte des exigences de vos bases de données utilisateur et de vos clients.

Lorsque vous créez un cluster à utiliser avec Babelfish, Aurora PostgreSQL crée les bases de données système, `master` et `tempdb`. Si vous avez créé ou modifié des objets dans les bases de données système (`master` ou `tempdb`), veillez à recréer ces objets dans votre nouveau cluster. Contrairement à SQL Server, Babelfish ne se réinitialise pas `tempdb` après le redémarrage d'un cluster.

Utilisez le mode de migration d'une base de données individuelle dans les cas suivants :

- Si vous migrez une base de données SQL Server individuelle. En mode Base de données individuelle, les noms des schémas migrés, lorsqu'ils font l'objet d'un accès depuis PostgreSQL, sont identiques aux noms des schémas SQL Server d'origine. Cela permet de réduire les modifications de code apportées aux requêtes SQL existantes si vous souhaitez les optimiser pour les exécuter avec une connexion PostgreSQL.
- Si votre objectif final est une migration complète vers l'instance native d'Aurora PostgreSQL. Avant toute migration, regroupez vos schémas en un seul (`dbo`), puis procédez à une migration vers un seul cluster pour limiter les modifications nécessaires.

Utilisez le mode de migration de plusieurs bases de données dans les cas suivants :

- Si vous souhaitez bénéficier de l'expérience SQL Server par défaut avec plusieurs bases de données utilisateur dans la même instance.
- Si plusieurs bases de données utilisateur doivent être migrées ensemble.

Paramètres du groupe de paramètres de cluster de bases de données pour Babelfish

Lorsque vous créez un cluster de bases de données Aurora PostgreSQL et choisissez Turn on Babelfish (Activer Babelfish), un groupe de paramètres de cluster de bases de données est créé automatiquement pour vous si vous choisissez Create new (Créer un nouveau). Ce groupe de paramètres de cluster de bases de données est basé sur le groupe de paramètres de cluster de bases de données Aurora PostgreSQL pour la version d'Aurora PostgreSQL choisie pour l'installation, par exemple Aurora PostgreSQL version 14. Il est nommé en utilisant le modèle général suivant :

```
custom-aurora-postgresql14-babelfish-compat-3
```

Vous pouvez modifier les paramètres suivants pendant le processus de création du cluster, mais certains d'entre eux ne peuvent pas être modifiés une fois qu'ils sont stockés dans le groupe de paramètres personnalisés. Choisissez donc soigneusement :

- Base de données unique ou plusieurs bases de données
- Paramètres régionaux de classement par défaut
- Nom du classement
- Groupe de paramètres de base de données

Pour utiliser un cluster de bases de données Aurora PostgreSQL version 13 ou un groupe de paramètres ultérieur, modifiez le groupe et définissez le paramètre `babelfish_status` sur on. Spécifiez toutes les options Babelfish avant de créer votre cluster Aurora PostgreSQL. Pour en savoir plus, consultez Groupes de paramètres pour Amazon Aurora.

Les paramètres suivants contrôlent les préférences Babelfish. Sauf indication contraire dans la description, les paramètres sont modifiables. La valeur par défaut est incluse dans la description. Pour voir les valeurs autorisées pour n'importe quel paramètre, procédez comme suit :

> **Note**
>
> Lorsque vous associez un nouveau groupe de paramètres de base de données à une instance de base de données, les paramètres statiques et dynamiques modifiés sont appliqués uniquement après que l'instance de base de données est redémarrée. Toutefois, si vous modifiez des paramètres dynamiques dans le groupe de paramètres de base de

données après l'avoir associé à l'instance de base de données, ces modifications sont appliquées immédiatement sans redémarrage.

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Parameter groups (Groupes de paramètres) dans le menu de navigation.
3. Choisissez le groupe de paramètres du cluster de bases de données `default.aurora-postgresql14` dans la liste.
4. Saisissez le nom d'un paramètre dans le champ de recherche. Par exemple, saisissez `babelfishpg_tsql.default_locale` dans le champ de recherche pour afficher ce paramètre ainsi que sa valeur par défaut et ses valeurs autorisées.

> **Note**
>
> Les bases de données globales Babelfish pour Aurora PostgreSQL ne fonctionnent dans les régions secondaires que si les paramètres suivants sont activés dans ces régions.

Paramètre	Description	Type d'application	Modifiable
babelfishpg_tsql.apg_enable_correlated_scalar_transform	Permet au planificateur de transformer une sous-requête scalaire corrélée dans Babelfish. (Par défaut : on) (Autorisé : on, off)	dynamic	true
babelfishpg_tsql.apg_enable_subquery_cache	Active l'utilisation du cache pour les sous-requêtes scalaires corrélées dans Babelfish. (Par défaut : on) (Autorisé : on, off)	dynamic	true

Paramètre	Description	Type d’application	Modifiable
babelfishpg_tds.tds_default_numeric_scale	Définit l’échelle par défaut du type numérique à envoyer dans les métadonnées de la colonne TDS si le moteur n’en spécifie pas. (Par défaut : 8) (Autorisée : 0 à 38)	dynamic	true
babelfishpg_tds.tds_default_numeric_precision	Entier qui définit la précision par défaut du type numérique à envoyer dans les métadonnées de la colonne TDS si le moteur n’en spécifie pas. (Par défaut : 38) (Autorisée : 1 à 38)	dynamic	true
babelfishpg_tds.tds_default_packet_size	Entier qui définit la taille par défaut des paquets pour la connexion des clients SQL Server. (Par défaut : 4 096) (Autorisée : 512 à 32 767)	dynamic	true

Paramètre	Description	Type d’application	Modifiable
babelfishpg_tds.tds_default_protocol_version	Entier qui définit une version de protocole TDS par défaut pour la connexion des clients. (Par défaut : DEFAULT) (Autorisée : TDSv7.0, TDSv7.1, TDSv7.1.1, TDSv7.2, TDSv7.3A, TDSv7.3B, TDSv7.4, DEFAULT)	dynamic	true
babelfishpg_tds.default_server_name	Chaîne qui identifie le nom par défaut du serveur Babelfish. (Par défaut : Microsoft SQL Server) (Autorisée : null)	dynamic	true
babelfishpg_tds.tds_debug_log_level	Entier qui définit le niveau de journalisation dans TDS ; 0 désactive la journalisation. (Par défaut : 1) (Autorisée : 0, 1, 2, 3)	dynamic	true

Paramètre	Description	Type d’application	Modifiable
babelfishpg_tds.listen_addresses	Chaîne qui définit le nom d’hôte ou les adresses IP sur lesquelles écouter TDS. Ce paramètre ne peut pas être modifié une fois que le cluster de bases de données Babelfish a été créé. (Par défaut : *) (Autorisée : null)	–	false
babelfishpg_tds.port	Entier qui spécifie le port TCP utilisé pour les requêtes dans la syntaxe SQL Server. (Par défaut : 1 433) (Autorisée : 1 à 65 535)	statique	true
babelfishpg_tds.tds_ssl_encrypt	Booléen qui active (0) ou désactive (1) le chiffrement des données traversant le port d’écoute TDS. Pour obtenir des informations détaillées sur l’utilisation de SSL pour les connexions client, consultez Paramètres SSL Babelfish et connexions client. (Par défaut : 0) (Autorisée : 0, 1)	dynamic	true

Paramètre	Description	Type d’application	Modifiable
babelfishpg_tds.tds_ssl_max_protocol_version	Chaîne qui spécifie la version la plus élevée du protocole SSL/TLS à utiliser pour la session TDS. (Par défaut : « TLSv1.2 ») (Autorisée : « TLSv1 », « TLSv1.1 », « TLSv1.2 »)	dynamic	true
babelfishpg_tds.tds_ssl_min_protocol_version	Chaîne qui spécifie la version minimale du protocole SSL/TLS à utiliser pour la session TDS. (Par défaut : ’TLSv1.2’ à partir d’Aurora PostgreSQL version 16, ’TLSv1’ pour les versions antérieures à Aurora PostgreSQL version 16) (Autorisé : ’TLSv1’, ’TLSv1.1’, ’TLSv1.2’)	dynamic	true

Paramètre	Description	Type d’application	Modifiable
babelfishpg_tds.unix_socket_directories	Chaîne qui identifie le répertoire des sockets Unix du serveur TDS. Ce paramètre ne peut pas être modifié une fois que le cluster de bases de données Babelfish a été créé. (Par défaut : /tmp) (Autorisée : null)	–	false
babelfishpg_tds.unix_socket_group	Chaîne qui identifie le groupe de sockets Unix du serveur TDS. Ce paramètre ne peut pas être modifié une fois que le cluster de bases de données Babelfish a été créé. (Par défaut : rdsdb) (Autorisée : null)	–	false

Paramètre	Description	Type d’application	Modifiable
babelfishpg_tsql.default_locale	Chaîne qui spécifie les paramètres régionaux par défaut utilisés pour les classements Babelfish. Les paramètres régionaux par défaut se limitent aux paramètres régionaux ; ils n’incluent aucun qualificatif. Définissez ce paramètre au moment de l’approvisionnement d’un cluster de bases de données Babelfish. Une fois le cluster de bases de données approvisionné, les modifications apportées à ce paramètre sont ignorées. (Par défaut : en_US) (Autorisée : voir tables)	statique	true

Paramètre	Description	Type d’application	Modifiable
babelfishpg_tsql.migration_mode	Liste non modifiable qui spécifie la prise en charge des bases de données utilisateur uniques ou multiples. Définissez ce paramètre au moment de l’approvisionnement d’un cluster de bases de données Babelfish. Une fois le cluster de bases de données approvisionné, vous ne pouvez pas modifier la valeur de ce paramètre. (Par défaut : multi-db à partir d’Aurora PostgreSQL version 16, single-db pour les versions antérieures à Aurora PostgreSQL version 16) (Autorisé : single-db, multi-db, null)	statique	true

Paramètre	Description	Type d’application	Modifiable
babelfishpg_tsql.server_collation_name	Chaîne qui spécifie le nom du classement utilisé pour les actions au niveau du serveur. Définissez ce paramètre au moment de l’approvisionnement d’un cluster de bases de données Babelfish. Une fois le cluster de bases de données approvisionné, ne modifiez pas la valeur de ce paramètre. (Par défaut : bbf_unicode_general_ci_as) (Autorisée : voir tables)	statique	true
babelfishpg_tsql.version	Chaîne qui définit la sortie de la variable @@VERSION. Ne modifiez pas cette valeur pour les clusters de bases de données Aurora PostgreSQL. (Par défaut : null) (Autorisée : default)	dynamic	true

Paramètre	Description	Type d’application	Modifiable
rds.babelfish_status	Chaîne qui définit l’état de la fonctionnalité Babelfish . Lorsque ce paramètre est défini sur `datatypes only` , Babelfish est désactivé, mais les types de données SQL Server sont toujours disponibles. (Par défaut : off) (Autorisée : on, off, datatypesonly)	statique	true
unix_socket_permissions	Entier qui définit les autorisations des sockets Unix du serveur TDS. Ce paramètre ne peut pas être modifié une fois que le cluster de bases de données Babelfish a été créé. (Par défaut : 0700) (Autorisée : 0 à 511)	–	false

Paramètres SSL Babelfish et connexions client

Pour exiger que les connexions à votre cluster de bases de données Babelfish pour Aurora PostgreSQL utilisent SSL/TLS, appliquez le paramètre `rds.force_ssl`.

- Pour que les connexions utilisent SSL/TLS, définissez la valeur du paramètre `rds.force_ssl` sur 1 (activé).
- Pour que les connexions n'utilisent plus SSL/TLS, définissez la valeur du paramètre `rds.force_ssl` sur 0 (désactivé).

La valeur par défaut de ce paramètre dépend de la version d'Aurora PostgreSQL :

- Pour Aurora PostgreSQL 17 et les versions ultérieures : la valeur par défaut est 1 (activé).
- Pour Aurora PostgreSQL 16 et les versions antérieures : la valeur par défaut est 0 (désactivé).

> **Note**
>
> Lorsque vous effectuez une mise à niveau de version majeure d'Aurora PostgreSQL 16 ou version antérieure vers la version 17 ou une version ultérieure, la valeur par défaut du paramètre passe de 0 (désactivé) à 1 (activé). Ce changement peut entraîner des défaillances de connectivité pour les applications qui ne sont pas configurées pour le protocole SSL. Pour revenir au comportement par défaut précédent, définissez ce paramètre sur 0 (désactivé).

Pour des informations spécifiques au pilote, consultez Connexion à un cluster de bases de données Babelfish.

Lorsqu'un client se connecte au port TDS (`1433` par défaut), Babelfish compare le paramètre SSL (Secure Sockets Layer) envoyé lors de l'établissement de la liaison client avec le paramètre Babelfish SSL (`tds_ssl_encrypt`). Babelfish détermine alors si une connexion est autorisée. Si une connexion est autorisée, le comportement de chiffrement est appliqué ou non, en fonction de vos paramètres et de la prise en charge du chiffrement offerte par le client.

Le tableau suivant montre comment Babelfish se comporte pour chaque combinaison.

Paramètre SSL du client	Paramètre SSL de Babelfish	rds.force_ssl	Connexion autorisée ?	Valeur renvoyée au client
ENCRYPT_ON	N'importe quel compte	N'importe quel compte	Autorisée, la connexion est intégralement chiffrée	ENCRYPT_ON
ENCRYPT_OFF	tds_ssl_encrypt=1	N'importe quel compte	Autorisée, la connexion est intégralement chiffrée	ENCRYPT_REQ
ENCRYPT_OFF	tds_ssl_encrypt=0	rds.force_ssl=0	Autorisée, le paquet de connexion est chiffré	ENCRYPT_OFF
ENCRYPT_OFF	tds_ssl_encrypt=0	rds.force_ssl=1	Non, connexion fermée	ENCRYPT_OFF
ENCRYPT_NOT_SUP	tds_ssl_encrypt=0	rds.force_ssl=0	Oui	ENCRYPT_NOT_SUP
ENCRYPT_NOT_SUP	tds_ssl_encrypt=1	N'importe quel compte	Non, connexion fermée	ENCRYPT_REQ
ENCRYPT_NOT_SUP	tds_ssl_encrypt=0	rds.force_ssl=1	Non, connexion fermée	ENCRYPT_NOT_SUP
ENCRYPT_CLIENT_CERT	N'importe quel compte	N'importe quel compte	Non, connexion fermée	Non pris en charge

Comprendre les classements dans Babelfish pour Aurora PostgreSQL.

Lorsque vous créez un cluster de bases de données Aurora PostgreSQL avec Babelfish, vous choisissez un classement pour vos données. Un classement spécifie l'ordre de tri et les modèles de bits qui génèrent le texte ou les caractères dans un langage humain écrit donné. Un classement inclut des règles de comparaison des données pour un ensemble donné de modèles de bits. Le classement est lié à la localisation. Les différents paramètres régionaux affectent le mappage des caractères, l'ordre de tri, etc. Les attributs de classement sont reflétés dans les noms des différents classements. Pour plus d'informations sur les attributs, consultez Babelfish collation attributes table.

Babelfish mappe les classements SQL Server à des classements comparables fournis par Babelfish. Babelfish prédéfinit les classements Unicode avec des comparaisons de chaînes et des ordres de tri sensibles aux particularités culturelles. Babelfish permet également de traduire les classements de votre base de données SQL Server vers le classement Babelfish le plus proche. Des classements spécifiques aux paramètres régionaux sont fournis pour différentes langues et régions.

Certains classements spécifient une page de code correspondant à un encodage côté client. Babelfish traduit automatiquement l'encodage du serveur vers l'encodage du client en fonction du classement de chaque colonne de sortie.

Babelfish prend en charge les classements répertoriés dans Babelfish supported collations table. Babelfish mappe les classements SQL Server à des classements comparables fournis par Babelfish.

Babelfish utilise la version 153.80 de la bibliothèque de classements ICU (International Components for Unicode. Pour plus d'informations sur les classements ICU, consultez Collation (Classement) dans la documentation ICU. Pour en savoir plus sur PostgreSQL et le classement, consultez Collation Support (Prise en charge des classements) dans la documentation PostgreSQL.

Rubriques

Paramètres de cluster de bases de données qui contrôlent le classement et les paramètres régionaux

Les paramètres suivants ont une incidence sur le comportement du classement.

babelfishpg_tsql.default_locale

Ce paramètre spécifie les paramètres régionaux par défaut utilisés par le classement. Ce paramètre est utilisé en combinaison avec les attributs répertoriés dans Babelfish collation attributes table pour personnaliser les classements liés à une langue et à une région spécifiques. La valeur par défaut de ce paramètre est `en-US`.

Les paramètres régionaux par défaut s'appliquent à tous les noms de classement Babelfish commençant par « BBF » et à tous les classements SQL Server mappés à des classements Babelfish. La modification de la valeur de ce paramètre sur un cluster de bases de données Babelfish existant n'affecte pas les paramètres régionaux des classements existants. Pour obtenir la liste des classements, consultez Babelfish supported collations table.

babelfishpg_tsql.server_collation_name

Ce paramètre spécifie le classement par défaut du serveur (instance de cluster de bases de données Aurora PostgreSQL) et de la base de données. La valeur par défaut est `sql_latin1_general_cp1_ci_as`. `server_collation_name` doit être un classement `CI_AS` car, dans T-SQL, le classement du serveur détermine la façon dont les identifiants sont comparés.

Lorsque vous créez votre cluster de bases de données Babelfish, vous choisissez le Collation name (Nom du classement) dans la liste sélectionnable. Il s'agit notamment des classements répertoriés dans Babelfish supported collations table. Ne modifiez pas `server_collation_name` après la création de la base de données Babelfish.

Les paramètres que vous choisissez lorsque vous créez votre cluster de bases de données Babelfish pour Aurora PostgreSQL sont stockés dans le groupe de paramètres de cluster de bases de données associé au cluster pour ces paramètres et définissent son comportement de classement.

Classements déterministes et non déterministes dans Babelfish

Babelfish prend en charge les classements déterministes et non déterministes :

- Un classement déterministe considère que les caractères dont les séquences d'octets sont identiques sont égaux. Cela signifie que `x` et `X` ne sont pas égaux dans un classement

déterministe. Les classements déterministes peuvent être sensibles à la casse (CS) et sensibles aux accents (AS).

- Un classement non déterministe ne nécessite pas de correspondance identique. Un classement non déterministe considère que `x` et `X` sont égaux. Les classements non déterministes sont insensibles à la casse (CI), insensibles aux accents (AI) ou les deux.

Le tableau suivant présente certaines différences de comportement entre Babelfish et PostgreSQL lorsque vous utilisez des classements non déterministes.

Babelfish	PostgreSQL
Prend en charge la clause LIKE pour les classements CI_AS.	Ne prend pas en charge la clause LIKE sur les classements non déterministes.
Ne prend en charge la clause LIKE que sur les classements AI suivants issus de la version 4.2.0 de Babelfish : • bbf_unicode_cp1250_ci_ai • bbf_unicode_cp1250_cs_ai • bbf_unicode_cp1257_ci_ai • bbf_unicode_cp1257_cs_ai • bbf_unicode_cp1_ci_ai • bbf_unicode_cp1_cs_ai • estonian_ci_ai • finnish_swedish_ci_ai • french_ci_ai • latin1_general_ci_ai • latin1_general_cs_ai • modern_spanish_ci_ai • polish_ci_ai • sql_latin1_general_cp1_ci_ai • sql_latin1_general_cp1_cs_ai	Ne prend pas en charge la clause LIKE sur les classements non déterministes.

Babelfish	PostgreSQL
• traditional_spanish_ci_ai	

Pour obtenir la liste des autres limites et différences de comportement pour Babelfish par rapport à SQL Server et PostgreSQL, consultez Limites et différences de comportement des classements.

Babelfish et SQL Server suivent une convention de dénomination pour les classements qui décrivent les attributs de classement, comme indiqué dans le tableau suivant.

Attribut	Description
AI	Insensible aux accents.
AS	Sensibles aux accents.
BIN2	BIN2 exige que les données soient triées dans l'ordre des points de code. L'ordre des points de code Unicode suit le même ordre de caractères pour les encodages UTF-8, UTF-16 et UCS-2. L'ordre des points de code est un classement déterministe rapide.
CI	Insensible à la casse.
CS	Sensible à la casse
PREF	Pour trier les majuscules avant les minuscules, utilisez un classement PREF. Si la comparaison est insensible à la casse, la version majuscule d'une lettre est triée avant la version minuscule, à condition qu'il n'y ait pas d'autre distinction. La bibliothèque ICU prend en charge les préférences pour les majuscules avec `colCaseFirst=upper` , mais pas pour les classements CI_AS. PREF ne peut être appliqué qu'aux classements déterministes `CS_AS`.

Classements pris en charge au niveau de la base de données dans Babelfish

Les classements suivants sont pris en charge au niveau de la base de données dans Babelfish :

- bbf_unicode_bin2

- bbf_unicode_cp1_ci_ai
- bbf_unicode_cp1_ci_as
- bbf_unicode_cp1250_ci_ai
- bbf_unicode_cp1250_ci_as
- bbf_unicode_cp1257_ci_ai
- bbf_unicode_cp1257_ci_as
- estonian_ci_ai
- estonian_ci_as
- finnish_swedish_ci_ai
- finnish_swedish_ci_as
- french_ci_ai
- french_ci_as
- latin1_general_bin2
- latin1_general_ci_ai
- latin1_general_ci_as
- latin1_general_90_bin2
- latin1_general_100_bin2
- latin1_general_140_bin2
- modern_spanish_ci_ai
- modern_spanish_ci_as
- polish_ci_ai
- polish_ci_as
- sql_latin1_general_cp1_ci_ai
- sql_latin1_general_cp1_ci_as
- sql_latin1_general_cp1250_ci_as
- sql_latin1_general_cp1251_ci_as
- sql_latin1_general_cp1257_ci_as
- traditional_spanish_ci_ai
- traditional_spanish_ci_as

> **Note**
>
> Pour utiliser un classement différent au niveau de la base de données, assurez-vous qu'il correspond au classement au niveau du serveur. Pour plus d'informations, consultez Classements de serveur et d'objet dans Babelfish.

Classements de serveur et d'objet dans Babelfish

Utilisez les classements suivants comme classement de serveur ou classement d'objet.

ID du classement	Remarques
bbf_unicode_general_ci_as	Prend en charge la comparaison insensible à la casse et l'opérateur LIKE.
bbf_unicode_cp1_ci_as	Classement non déterministe également connu sous le nom de CP1252.
bbf_unicode_CP1250_ci_as	Classement non déterministe permettant de représenter des textes dans les langues d'Europe centrale et d'Europe de l'Est qui utilisent l'alphabet latin.
bbf_unicode_CP1251_ci_as	Classement non déterministe pour les langues utilisant l'alphabe t cyrillique.
bbf_unicode_cp1253_ci_as	Classement non déterministe utilisé pour représenter le grec moderne.
bbf_unicode_cp1254_ci_as	Classement non déterministe qui prend en charge le turc.
bbf_unicode_cp1255_ci_as	Classement non déterministe qui prend en charge l'hébreu.
bbf_unicode_cp1256_ci_as	

ID du classement	Remarques
	Classement non déterministe pour les langues utilisant l'alphabet arabe.
bbf_unicode_cp1257_ci_as	Classement non déterministe permettant de prendre en charge les langues estonienne, lettone et lituanienne.
bbf_unicode_cp1258_ci_as	Classement non déterministe utilisé pour les caractères vietnamiens.
bbf_unicode_cp874_ci_as	Classement non déterministe utilisé pour les caractères thaïlandais.
sql_latin1_general_cp1250_ci_as	Encodage non déterministe de caractères sur un octet utilisé pour représenter les caractères latins.
sql_latin1_general_cp1251_ci_as	Classement non déterministe prenant en charge les caractères latins.
sql_latin1_general_cp1_ci_as	Classement non déterministe prenant en charge les caractères latins.
sql_latin1_general_cp1253_ci_as	Classement non déterministe prenant en charge les caractères latins.
sql_latin1_general_cp1254_ci_as	Classement non déterministe prenant en charge les caractères latins.
sql_latin1_general_cp1255_ci_as	Classement non déterministe prenant en charge les caractères latins.

ID du classement	Remarques
sql_latin1_general_cp1256_ci_as	Classement non déterministe prenant en charge les caractères latins.
sql_latin1_general_cp1257_ci_as	Classement non déterministe prenant en charge les caractères latins.
sql_latin1_general_cp1258_ci_as	Classement non déterministe prenant en charge les caractères latins.
chinese_prc_ci_as	Classement non déterministe prenant en charge le chinois (RPC).
cyrillic_general_ci_as	Classement non déterministe prenant en charge l'alphabet cyrillique.
finnish_swedish_ci_as	Classement non déterministe prenant en charge le finnois.
french_ci_as	Classement non déterministe prenant en charge le français.
japanese_ci_as	Classement non déterministe prenant en charge le japonais. Pris en charge dans Babelfish 2.1.0 et versions ultérieures.
korean_wansung_ci_as	Classement non déterministe prenant en charge le coréen (avec tri par dictionnaire).
latin1_general_ci_as	Classement non déterministe prenant en charge les caractères latins.
modern_spanish_ci_as	Classement non déterministe prenant en charge l'espagnol moderne.

ID du classement	Remarques
polish_ci_as	Classement non déterministe prenant en charge le polonais.
thai_ci_as	Classement non déterministe prenant en charge le thaï.
traditional_spanish_ci_as	Classement non déterministe prenant en charge l’espagnol (tri traditionnel).
turkish_ci_as	Classement non déterministe prenant en charge le turc.
ukrainian_ci_as	Classement non déterministe prenant en charge l’ukrainien.
vietnamese_ci_as	Classement non déterministe prenant en charge le vietnamien.

Vous pouvez utiliser les classements suivants comme classements d’objets.

Langage	Options déterministes	Options non déterministes
Arabe	Arabic_CS_AS	Arabic_CI_AS Arabic_CI_AI
Alphabet arabe	BBF_Unicode_CP1256_CS_AS BBF_Unicode_Pref_CP1256_CS_AS	BBF_Unicode_CP1256_CI_AI BBF_Unicode_CP1256_CS_AI
Binaire	latin1_general_bin2 BBF_Unicode_BIN2	–
Langues d’Europe	BBF_Unicode_CP1250_CS_AS	BBF_Unicode_CP1250_CI_AI

Langage	Options déterministes	Options non déterministes
centrale et d’Europe de l’Est qui utilisent l’alphabet latin	BBF_Unicode_Pref_CP1250_CS_AS	BBF_Unicode_CP1250_CS_AI
Chinois	Chinese_PRC_CS_AS	Chinese_PRC_CI_AS Chinese_PRC_CI_AI
Cyrillic_General	Cyrillic_General_CS_AS	Cyrillic_General_CI_AS Cyrillic_General_CI_AI
Alphabet cyrillique	BBF_Unicode_CP1251_CS_AS BBF_Unicode_Pref_CP1251_CS_AS	BBF_Unicode_CP1251_CI_AI BBF_Unicode_CP1251_CS_AI
Estonian	Estonian_CS_AS	Estonian_CI_AS Estonian_CI_AI
Estonien, letton et lituanien	BBF_Unicode_CP1257_CS_AS BBF_Unicode_Pref_CP1257_CS_AS	BBF_Unicode_CP1257_CI_AI BBF_Unicode_CP1257_CS_AI
Finnish_Swedish	Finnish_Swedish_CS_AS	Finnish_Swedish_CI_AS Finnish_Swedish_CI_AI
Français	French_CS_AS	French_CI_AS French_CI_AI

Langage	Options déterministes	Options non déterministes
Grec	Greek_CS_AS	Greek_CI_AS Greek_CI_AI
Hébreu	BBF_Unicode_CP1255_CS_AS BBF_Unicode_Pref_CP1255_CS_AS Hebrew_CS_AS	BBF_Unicode_CP1255_CI_AI BBF_Unicode_CP1255_CS_AI Hebrew_CI_AS Hebrew_CI_AI
Japonais (Babelfish 2.1.0 et versions ultérieures)	Japanese_CS_AS	Japanese_CI_AI Japanese_CI_AS
Korean_Wamsung	Korean_Wamsung_CS_AS	Korean_Wamsung_CI_AS Korean_Wamsung_CI_AI
Caractères latins pour la page de code CP1252	latin1_general_cs_as BBF_Unicode_General_CS_AS BBF_Unicode_General_Pref_CS_AS BBF_Unicode_Pref_CP1_CS_AS BBF_Unicode_CP1_CS_AS	latin1_general_ci_as latin1_general_ci_ai latin1_general_cs_ai BBF_Unicode_General_CI_AI BBF_Unicode_General_CS_AI BBF_Unicode_CP1_CI_AI BBF_Unicode_CP1_CS_AI

Langage	Options déterministes	Options non déterministes
Grec moderne	BBF_Unicode_CP1253_CS_AS BBF_Unicode_Pref_CP1253_CS_AS	BBF_Unicode_CP1253_CI_AI BBF_Unicode_CP1253_CS_AI
Modern_Spanish	Modern_Spanish_CS_AS	Modern_Spanish_CI_AS Modern_Spanish_CI_AI
Mongol	Mongolian_CS_AS	Mongolian_CI_AS Mongolian_CI_AI
Polonais	Polish_CS_AS	Polish_CI_AS Polish_CI_AI
Thaï	BBF_Unicode_CP874_CS_AS BBF_Unicode_Pref_CP874_CS_AS Thai_CS_AS	BBF_Unicode_CP874_CI_AI BBF_Unicode_CP874_CS_AI Thai_CI_AS, Thai_CI_AI
Traditional_Spanish	Traditional_Spanish_CS_AS	Traditional_Spanish_CI_AS Traditional_Spanish_CI_AI
Turc	BBF_Unicode_CP1254_CS_AS BBF_Unicode_Pref_CP1254_CS_AS Turkish_CS_AS	BBF_Unicode_CP1254_CI_AI BBF_Unicode_CP1254_CS_AI Turkish_CI_AS, Turkish_CI_AI

Langage	Options déterministes	Options non déterministes
Ukrainien	Ukrainian_CS_AS	Ukranian_CI_AS Ukranian_CI_AI
Vietnamien	BBF_Unicode_CP1258_CS_AS BBF_Unicode_Pref_CP1258_CS_AS Vietnamese_CS_AS	BBF_Unicode_CP1258_CI_AI BBF_Unicode_CP1258_CS_AI Vietnamese_CI_AS Vietnamese_CI_AI

Comportement de classement par défaut dans Babelfish

Auparavant, le classement par défaut des types de données pouvant être classées était `pg_catalog.default`. Les types de données et les objets qui dépendent de ces types de données sont classés selon le classement sensible à la casse. Cette condition peut avoir un impact sur les objets T-SQL du jeu de données avec un classement sensible à la casse. À partir de Babelfish 2.3.0, le classement par défaut pour les types de données pouvant être classées (à l'exception de TEXT et NTEXT) est le même que le classement du paramètre `babelfishpg_tsql.server_collation_name`. Lorsque vous passez à Babelfish 2.3.0, le classement par défaut est sélectionné automatiquement au moment de la création du cluster de bases de données, ce qui n'a aucun impact visible.

Gestion des classements

La bibliothèque ICU fournit un suivi des versions des classements afin que les index qui reposent sur les classements puissent être réindexés lorsqu'une nouvelle version d'ICU est disponible. Pour savoir si votre base de données actuelle comporte des classements qui doivent être actualisés, vous pouvez utiliser la requête suivante après vous être connecté avec `psql` ou `pgAdmin` :

```
SELECT pg_describe_object(refclassid, refobjid,
    refobjsubid) AS "Collation",
    pg_describe_object(classid, objid, objsubid) AS "Object"
    FROM pg_depend d JOIN pg_collation c ON refclassid = 'pg_collation'::regclass
    AND refobjid = c.oid WHERE c.collversion <> pg_collation_actual_version(c.oid)
```

```
    ORDER BY 1, 2;
```

Cette requête renvoie la sortie suivante :

```
 Collation | Object
-----------+--------
(0 rows)
```

Dans cet exemple, aucun classement n'a besoin d'être mis à jour.

Pour obtenir la liste des classements prédéfinis dans votre base de données Babelfish, vous pouvez utiliser `psql` ou `pgAdmin` avec la requête suivante :

```
SELECT * FROM pg_collation;
```

Les classements prédéfinis sont stockés dans la table `sys.fn_helpcollations`. Vous pouvez utiliser la commande suivante pour afficher des informations sur un classement (comme ses indicateurs lcid, style et collate). Pour obtenir la liste de tous les classements à l'aide de `sqlcmd`, connectez-vous au port T-SQL (1433, par défaut) et exécutez la requête suivante :

```
1> :setvar SQLCMDMAXVARTYPEWIDTH 40
2> :setvar SQLCMDMAXFIXEDTYPEWIDTH 40
3> SELECT * FROM fn_helpcollations()
4> GO
name                                 description
------------------------------------ ----------------------------------------
arabic_cs_as                         Arabic, case-sensitive, accent-sensitive
arabic_ci_ai                         Arabic, case-insensitive, accent-insensi
arabic_ci_as                         Arabic, case-insensitive, accent-sensiti
bbf_unicode_bin2                     Unicode-General, case-sensitive, accent-
bbf_unicode_cp1250_ci_ai             Default locale, code page 1250, case-ins
bbf_unicode_cp1250_ci_as             Default locale, code page 1250, case-ins
bbf_unicode_cp1250_cs_ai             Default locale, code page 1250, case-sen
bbf_unicode_cp1250_cs_as             Default locale, code page 1250, case-sen
bbf_unicode_pref_cp1250_cs_as        Default locale, code page 1250, case-sen
bbf_unicode_cp1251_ci_ai             Default locale, code page 1251, case-ins
bbf_unicode_cp1251_ci_as             Default locale, code page 1251, case-ins
bbf_unicode_cp1254_ci_ai             Default locale, code page 1254, case-ins
...
(124 rows affected)
```

Les lignes 1 et 2 de l'exemple limitent la sortie à des fins de lisibilité de la documentation uniquement.

```
1> SELECT SERVERPROPERTY('COLLATION')
2> GO
serverproperty
--------------------------------------------------------------------------
sql_latin1_general_cp1_ci_as

(1 rows affected)
1>
```

Limites et différences de comportement des classements

Babelfish utilise la bibliothèque ICU pour la prise en charge des classements. PostgreSQL repose sur une version spécifique d'ICU et ne peut correspondre qu'à une seule version d'un classement. Les variations d'une version à l'autre sont inévitables, de même que les variations mineures dans le temps à mesure que les langues évoluent. La liste suivante répertorie les limites et variations de comportement connues pour les classements Babelfish :

- Index et dépendance de type de classement : un index figurant sur un type défini par l'utilisateur qui dépend de la bibliothèque de classements International Components for Unicode (ICU – la bibliothèque utilisée par Babelfish) n'est pas invalidé lorsque la version de la bibliothèque est modifiée.
- Fonction COLLATIONPROPERTY : les propriétés de classement ne sont implémentées que pour les classements Babelfish BBF pris en charge. Pour plus d'informations, consultez le Babelfish supported collations table.
- Différences entre les règles de tri Unicode : les classements SQL pour SQL Server trient les données codées en Unicode (`nchar` et `nvarchar`) différemment des données qui ne sont pas codées en Unicode (`char` et `varchar`). Les bases de données Babelfish sont toujours codées en UTF-8 et appliquent toujours les règles de tri Unicode de manière cohérente, quel que soit le type de données. L'ordre de tri de `char` ou `varchar` est donc identique à celui de `nchar` ou `nvarchar`.
- Classements à égalité secondaire et comportement de tri : le classement ICU Unicode secondaire par défaut (`CI_AS`) trie les signes de ponctuation et les autres caractères non alphanumériques avant les caractères numériques, et les caractères numériques avant les caractères alphabétiques. Toutefois, l'ordre des signes de ponctuation et des autres caractères spéciaux est différent.
- Classements tertiaires, solution de contournement pour ORDER BY : les classements SQL, tels que `SQL_Latin1_General_Pref_CP1_CI_AS`, prennent en charge la fonction

TERTIARY_WEIGHTS et la capacité à trier les chaînes considérées comme égales au sein d'un classement CI_AS afin que le tri s'effectue d'abord sur les majuscules : ABC, ABc, AbC, Abc, aBC, aBc, abC et enfin abc. Ainsi, la fonction analytique DENSE_RANK OVER (ORDER BY column) évalue ces chaînes comme ayant le même rang, mais les classe en commençant par les majuscules au sein d'une partition.

Vous pouvez obtenir un résultat similaire avec Babelfish en ajoutant une clause COLLATE à la clause ORDER BY qui spécifie un classement CS_AS tertiaire indiquant @colCaseFirst=upper. Toutefois, le modificateur colCaseFirst s'applique uniquement aux chaînes qui présentent une égalité tertiaire (plutôt qu'une égalité secondaire comme avec le classement CI_AS). Par conséquent, vous ne pouvez pas émuler des classements SQL tertiaires à l'aide d'un seul classement ICU.

Pour contourner ce problème, nous vous recommandons de modifier les applications qui utilisent le classement SQL_Latin1_General_Pref_CP1_CI_AS afin d'utiliser le classement BBF_SQL_Latin1_General_CP1_CI_AS en premier. Ajoutez ensuite COLLATE BBF_SQL_Latin1_General_Pref_CP1_CS_AS à n'importe quelle clause ORDER BY de cette colonne.

- Extension de caractère : une extension de caractère traite un caractère comme étant égal à une séquence de caractères au niveau primaire. Le classement CI_AS par défaut de SQL Server prend en charge l'extension de caractère. Les classements ICU prennent en charge l'extension de caractère uniquement pour les classements insensibles aux accents.

  Lorsqu'une extension de caractère est nécessaire, utilisez un classement AI pour les comparaisons. Toutefois, ces classements ne sont actuellement pas pris en charge par l'opérateur LIKE.

- Codage char et varchar : lorsque des classements SQL sont utilisés pour les types de données char ou varchar, l'ordre de tri des caractères précédant le code ASCII 127 est déterminé par la page de code spécifique de ce classement SQL. Pour les classements SQL, les chaînes déclarées comme char ou varchar peuvent être triées différemment des chaînes déclarées comme nchar ou nvarchar.

  PostgreSQL code toutes les chaînes avec le codage de la base de données afin de convertir tous les caractères en UTF-8 et de les trier à l'aide des règles Unicode.

  Étant donné que les classements SQL trient les types de données nchar et nvarchar à l'aide de règles Unicode, Babelfish encode toutes les chaînes du serveur en utilisant UTF-8. Babelfish trie

les chaînes nchar et nvarchar de la même manière que les chaînes char et varchar, en utilisant les règles Unicode.

- Caractère supplémentaire : les fonctions SQL Server `NCHAR`, `UNICODE` et `LEN` prennent en charge les caractères des points de code en dehors du plan multilingue de base (BMP) Unicode. En revanche, les classements non SC utilisent des caractères de paire de substitution pour gérer les caractères supplémentaires. Pour les types de données Unicode, SQL Server peut représenter jusqu'à 65 535 caractères en utilisant la norme UCS-2, ou toute la gamme Unicode (1 114 111 caractères) si des caractères supplémentaires sont utilisés.
- Classements sensibles aux kanas (KS) : un classement sensible aux kanas (KS) traite les caractères japonais (kanas) `Hiragana` et `Katakana` différemment. ICU prend en charge la norme de classement japonaise `JIS X 4061`. Le modificateur de paramètres locaux `colhiraganaQ [on | off]`, désormais obsolète, peut fournir la même fonctionnalité que les classements KS. Toutefois, les classements KS du même nom que ceux de SQL Server ne sont actuellement pas pris en charge par Babelfish.
- Classements sensibles à la largeur (WS) : lorsqu'un caractère d'un seul octet (demi-largeur) et le même caractère représenté par un caractère de deux octets (pleine largeur) sont traités différemment, le classement est dit sensible à la largeur (WS). Les classements WS portant le même nom que ceux de SQL Server ne sont actuellement pas pris en charge par Babelfish.
- Classements VSS (sensibilité au sélecteur de variante) : les classements VSS (sensibilité au sélecteur de variante) font la distinction entre les sélecteurs de variantes idéographiques dans les classements japonais `Japanese_Bushu_Kakusu_140` et `Japanese_XJIS_140`. Une séquence de variantes est constituée d'un caractère de base et d'un sélecteur de variante supplémentaire. Si vous ne sélectionnez pas le champ `_VSS`, le sélecteur de variante n'est pas pris en compte dans la comparaison.

  Les classements VSS ne sont actuellement pas pris en charge par Babelfish.
- Classements BIN et BIN2 : un classement BIN2 trie les caractères en fonction de l'ordre des points de code. L'ordre binaire octet par octet du format UTF-8 préserve l'ordre des points de code Unicode, ce qui en fait probablement le classement le plus performant. Si l'ordre des points de code Unicode fonctionne pour une application, n'hésitez pas à utiliser un classement BIN2. Toutefois, l'utilisation d'un classement BIN2 peut entraîner l'affichage des données dans un ordre culturellement inattendu sur le client. De nouveaux mappages vers des caractères minuscules sont ajoutés à Unicode au fil du temps. `LOWER` peut donc fonctionner différemment selon les versions d'ICU. Il s'agit d'un cas particulier du problème plus général de gestion des versions du classement plutôt que d'un problème spécifique au classement BIN2.

Babelfish fournit le classement `BBF_Latin1_General_BIN2` avec la distribution Babelfish pour assembler dans l'ordre les points de code Unicode. Dans un classement BIN, seul le premier caractère est trié en tant que wchar. Les autres caractères sont triés octet par octet, dans l'ordre des points de code en fonction de leur encodage. Cette approche ne suit pas les règles de classement Unicode et n'est pas prise en charge par Babelfish.

- Classements non déterministes et limite CHARINDEX : pour les versions de Babelfish antérieures à la version 2.1.0, vous ne pouvez pas utiliser CHARINDEX avec des classements non déterministes. Par défaut, Babelfish utilise un classement insensible à la casse (non déterministe). L'utilisation de CHARINDEX pour les versions antérieures de Babelfish génère l'erreur d'exécution suivante :

```
nondeterministic collations are not supported for substring searches
```

> **Note**
>
> Cette limite et cette solution de contournement s'appliquent uniquement à Babelfish version 1.x (Aurora PostgreSQL versions 13.x). Babelfish 2.1.0 et versions ultérieures ne présentent pas ce problème.

Vous pouvez contourner ce problème de l'une des manières suivantes :

- Convertir explicitement l'expression en une collation sensible à la casse et respecter la casse des deux arguments en appliquant les instructions LOWER ou UPPER. Par exemple, la commande `SELECT charindex('x', a) FROM t1` deviendrait ce qui suit :

```
SELECT charindex(LOWER('x'), LOWER(a COLLATE sql_latin1_general_cp1_cs_as)) FROM t1
```

- Créez une fonction SQL f_charindex, et remplacez les appels à CHARINDEX par des appels à la fonction suivante :

```
CREATE function f_charindex(@s1 varchar(max), @s2 varchar(max)) RETURNS int
AS
BEGIN
declare @i int = 1
WHILE len(@s2) >= len(@s1)
BEGIN
  if LOWER(@s1) = LOWER(substring(@s2,1,len(@s1))) return @i
```

```
  set @i += 1
  set @s2 = substring(@s2,2,999999999)
END
return 0
END
go
```

Gestion du traitement des erreurs Babelfish avec des trappes de secours

Dans la mesure du possible, Babelfish imite le comportement de SQL en ce qui concerne le flux de contrôle et l'état de transaction. Lorsque Babelfish rencontre une erreur, il renvoie un code d'erreur semblable au code d'erreur SQL Server. Si Babelfish ne peut pas mapper l'erreur à un code SQL Server, il renvoie un code d'erreur fixe (`33557097`) et prend des mesures spécifiques en fonction du type d'erreur, comme suit :

- Pour les erreurs de compilation, Babelfish annule la transaction.
- Pour les erreurs d'exécution, Babelfish termine le lot et annule la transaction.
- Pour une erreur de protocole entre le client et le serveur, la transaction n'est pas annulée.

Si un code d'erreur ne peut pas être mappé à un code équivalent et que le code d'une erreur similaire est disponible, il est mappé au code alternatif. Par exemple, les comportements à l'origine des codes SQL Server `8143` et `8144`sont tous deux mappés au code `8143`.

Les erreurs qui ne peuvent pas être mappées ne respectent pas une construction `TRY... CATCH`.

Vous pouvez utiliser `@@ERROR` pour renvoyer un code d'erreur SQL Server, ou la fonction `@@PGERROR` pour renvoyer un code d'erreur PostgreSQL. Vous pouvez également utiliser la fonction `fn_mapped_system_error_list` pour renvoyer une liste de codes d'erreur mappés. Pour en savoir plus sur les codes d'erreur PostgreSQL, consultez le site Internet PostgreSQL.

Modification des paramètres de la trappe de secours de Babelfish

Pour gérer les déclarations susceptibles d'échouer, Babelfish définit certaines options appelées trappes de secours. Une trappe de secours est une option qui spécifie le comportement de Babelfish lorsqu'il rencontre une fonction ou une syntaxe non prise en charge.

Vous pouvez utiliser la procédure stockée `sp_babelfish_configure` pour contrôler les paramètres d'une trappe de secours. Utilisez le script pour définir la trappe de secours sur `ignore` ou `strict`. Si elle est définie sur `strict`, Babelfish renvoie une erreur que vous devez corriger avant de continuer.

Pour appliquer les changements à la session en cours et au niveau du cluster, incluez le mot-clé `server`.

Procédez comme suit :

- Pour répertorier toutes les trappes de secours et leur statut, ainsi que des informations relatives à leur utilisation, exécutez `sp_babelfish_configure`.
- Pour répertorier les trappes de secours nommées et leurs valeurs, pour la session en cours ou à l'échelle du cluster, exécutez la commande `sp_babelfish_configure '`*`hatch_name`*`'` où *`hatch_name`* correspond à l'identifiant d'une ou de plusieurs trappes de secours. Le nom d'une trappes de secours (*`hatch_name`*) peut utiliser des caractères génériques SQL, tels que « % ».
- Pour définir une ou plusieurs trappes de secours sur la valeur spécifiée, exécutez `sp_babelfish_configure ['`*`hatch_name`*`' [, 'strict'|'ignore' [, 'server']]]`. Pour rendre les paramètres permanents à l'échelle d'un cluster, ajoutez le mot-clé `server`, comme indiqué ci-après :

  ```
  EXECUTE sp_babelfish_configure 'escape_hatch_unique_constraint', 'ignore', 'server'
  ```

  Pour les appliquer à la session en cours uniquement, n'utilisez pas `server`.
- Pour remettre toutes les trappes de secours à leur valeur par défaut, lancez `sp_babelfish_configure 'default'` (Babelfish 1.2.0 et versions ultérieures).

La chaîne identifiant la (ou les) trappe(s) peut inclure des caractères génériques SQL. L'exemple suivant définit toutes les trappes de secours syntaxiques à ignorer (`ignore`) pour le cluster Aurora PostgreSQL

```
EXECUTE sp_babelfish_configure '%', 'ignore', 'server'
```

Dans le tableau suivant, vous trouverez les descriptions et les valeurs par défaut des trappes de secours prédéfinies par Babelfish.

Trappe de secours	Description	Par défaut
`escape_hatch_checkpoint`	Autorise l'utilisation de l'instruction CHECKPOINT dans le code procédural, mais l'instruction CHECKPOINT n'est actuellement pas implémentée.	ignore

Trappe de secours	Description	Par défaut
`escape_hatch_constraint_name_for_default`	Détermine le comportement de Babelfish pour les noms de contraintes par défaut.	ignore
`escape_hatch_database_misc_options`	Détermine le comportement de Babelfish pour les options suivantes sur CREATE DATABASE: CONTAINMENT, DB_CHAINING, TRUSTWORTHY, PERSISTENT_LOG_BUFFER.	ignore
`escape_hatch_for_replication`	Détermine le comportement de Babelfish pour la clause [NOT] FOR REPLICATION lors de la création ou de la modification d'une table.	strict
`escape_hatch_fulltext`	Détermine le comportement de Babelfish pour les fonctions FULLTEXT telles que DEFAULT_FULLTEXT_LANGUAGE dans CREATE/ALTER DATABASE, CREATE FULLTEXT INDEX ou sp_fulltext_database.	ignore

Trappe de secours	Description	Par défaut
`escape_hatch_ignore_dup_key`	Contrôle le comportement de Babelfish lié à CREATE/ALTER TABLE et CREATE INDEX. Lorsque la valeur IGNORE_DUP_KEY=ON, provoque une erreur lorsqu'elle est définie sur `strict` (la valeur par défaut) ou ignore l'erreur lorsqu'elle est définie sur `ignore` (Babelfish version 1.2.0 et ultérieures).	strict
`escape_hatch_index_clustering`	Détermine le comportement de Babelfish pour les mots-clés CLUSTERED ou NONCLUSTERED liés aux index et aux contraintes PRIMARY KEY ou UNIQUE. Lorsque CLUSTERED est ignoré, l'index ou la contrainte est créé comme si NONCLUSTERED avait été spécifié.	ignore
`escape_hatch_index_columnstore`	Détermine le comportement de Babelfish pour la clause COLUMNSTORE. Si vous spécifiez `ignore`, Babelfish crée un index B-Tree classique.	strict

Trappe de secours	Description	Par défaut
`escape_hatch_join_hints`	Détermine le comportement des mots-clés dans un opérateur JOIN : LOOP, HASH, MERGE, REMOTE, REDUCE, REDISTRIBUTE, REPLICATE.	ignore
`escape_hatch_language_non_english`	Détermine le comportement de Babelfish pour les langues autres que l'anglais dans les messages affichés à l'écran. Babelfish ne prend actuellement en charge que `us_english` pour les messages affichés à l'écran. SET LANGUAGE peut utiliser une variable contenant le nom de la langue, de sorte que la langue définie ne peut être détectée qu'au moment de l'exécution.	strict
`escape_hatch_login_hashed_password`	En cas de définition sur ignore, l'erreur liée au mot-clé `HASHED` est supprimée pour `CREATE LOGIN` et `ALTER LOGIN`.	strict
`escape_hatch_login_misc_options`	En cas de définition sur ignore, l'erreur liée à des mots-clés autres que `HASHED`, `MUST_CHANGE` , `OLD_PASSWORD` et `UNLOCK` est supprimée pour `CREATE LOGIN` et `ALTER LOGIN`.	strict

Trappe de secours	Description	Par défaut
`escape_hatch_login_old_password`	En cas de définition sur ignore, l'erreur liée au mot-clé `OLD_PASSWORD` est supprimée pour `CREATE LOGIN` et `ALTER LOGIN`.	strict
`escape_hatch_login_password_must_change`	En cas de définition sur ignore, l'erreur liée au mot-clé `MUST_CHANGE` est supprimée pour `CREATE LOGIN` et `ALTER LOGIN`.	strict
`escape_hatch_login_password_unlock`	En cas de définition sur ignore, l'erreur liée au mot-clé `UNLOCK` est supprimée pour `CREATE LOGIN` et `ALTER LOGIN`.	strict
`escape_hatch_nocheck_add_constraint`	Détermine le comportement de Babelfish pour la clause WITH CHECK ou NOCHECK liée aux contraintes.	strict
`escape_hatch_nocheck_existing_constraint`	Détermine le comportement de Babelfish pour les contraintes FOREIGN KEY ou CHECK.	strict

Trappe de secours	Description	Par défaut
`escape_hatch_query_hints`	Détermine le comportement de Babelfish pour les indicateurs de requête. Lorsque cette option est définie sur ignore, le serveur ignore les indicateurs qui utilisent la clause OPTION (...) pour spécifier les aspects de traitement des requêtes. Les exemples incluent SELECT FROM ... OPTION(MERGE JOIN HASH, MAXRECURSION 10)).	ignore
`escape_hatch_rowversion`	Contrôle le comportement des types de données ROWVERSION et TIMESTAMP. Pour plus d'informations, consultez Utilisation des fonctionnalités Babelfish dont la mise en œuvre est limitée.	strict
`escape_hatch_schemabinding_function`	Détermine le comportement de Babelfish pour la clause WITH SCHEMABINDING. Par défaut, la clause WITH SCHEMABINDING est ignorée lorsqu'elle est spécifiée avec la commande CREATE ou ALTER FUNCTION.	ignore

Trappe de secours	Description	Par défaut
`escape_hatch_schemabinding_procedure`	Détermine le comportement de Babelfish pour la clause WITH SCHEMABINDING. Par défaut, la clause WITH SCHEMABINDING est ignorée lorsqu'elle est spécifiée avec la commande CREATE ou ALTER PROCEDURE.	ignore
`escape_hatch_rowguidcol_column`	Détermine le comportement de Babelfish pour la clause ROWGUIDCOL lors de la création ou de la modification d'une table.	strict
`escape_hatch_schemabinding_trigger`	Détermine le comportement de Babelfish pour la clause WITH SCHEMABINDING. Par défaut, la clause WITH SCHEMABINDING est ignorée lorsqu'elle est spécifiée avec la commande CREATE ou ALTER TRIGGER.	ignore
`escape_hatch_schemabinding_view`	Détermine le comportement de Babelfish pour la clause WITH SCHEMABINDING. Par défaut, la clause WITH SCHEMABINDING est ignorée lorsqu'elle est spécifiée avec la commande CREATE ou ALTER VIEW.	ignore

Trappe de secours	Description	Par défaut
`escape_hatch_session_settings`	Détermine le comportement de Babelfish à l'égard des instructions SET non prises en charge au niveau de la session.	ignore
`escape_hatch_showplan_all`	Contrôle le comportement de Babelfish lié à SET SHOWPLAN_ALL et SET STATISTICS PROFILE. Lorsqu'ils sont définis sur ignore (ignorer), ils se comportent comme SET BABELFISH_SHOWPLAN_ALL et SET BABELFISH_STATISTICS PROFILE ; lorsqu'ils sont définis sur strict, ils sont ignorés en silence.	strict
`escape_hatch_storage_on_partition`	Détermine le comportement de Babelfish pour la clause `ON partition_scheme column` lors de la définition du partitionnement. Babelfish n'implémente actuellement pas le partitionnement.	strict

Trappe de secours	Description	Par défaut
`escape_hatch_storage_options`	Trappe de secours associée à une option de stockage utilisée dans CREATE, ALTER DATABASE, TABLE, INDEX. Cela inclut les clauses (LOG) ON, TEXTIMAGE_ON, FILESTREAM_ON qui définissent les emplacements de stockage (partitions, groupes de fichiers) des tables, index et contraintes, ainsi que d'une base de données. Ce paramètre de trappe de secours s'applique à toutes ces clauses (y compris ON [PRIMARY] et ON "DEFAULT"). Une exception s'applique lorsqu'une partition est spécifiée pour une table ou un index accompagné de ON partition_scheme (column).	ignore
`escape_hatch_table_hints`	Détermine le comportement des indicateurs de table spécifiés à l'aide de la clause WITH (...).	ignore

Trappe de secours	Description	Par défaut
`escape_hatch_unique_constraint`	Lorsqu'elle est définie comme stricte, une obscure différence sémantique entre SQL Server et PostgreSQL dans le traitement des valeurs NULL sur les colonnes indexées peut entraîner des erreurs. La différence sémantiqu e n'apparaît que dans des cas d'utilisation irréalistes. Vous pouvez donc définir cette trappe de secours sur « ignorer » pour ne pas afficher l'erreur. Obsolète à partir des versions suivantes 3.6.0 et ultérieures et des versions 4.2.0 et ultérieures	strict

Création d'un cluster de bases de données Babelfish pour Aurora PostgreSQL

Babelfish pour Aurora PostgreSQL est pris en charge sur Aurora PostgreSQL version 13.4 et versions ultérieures.

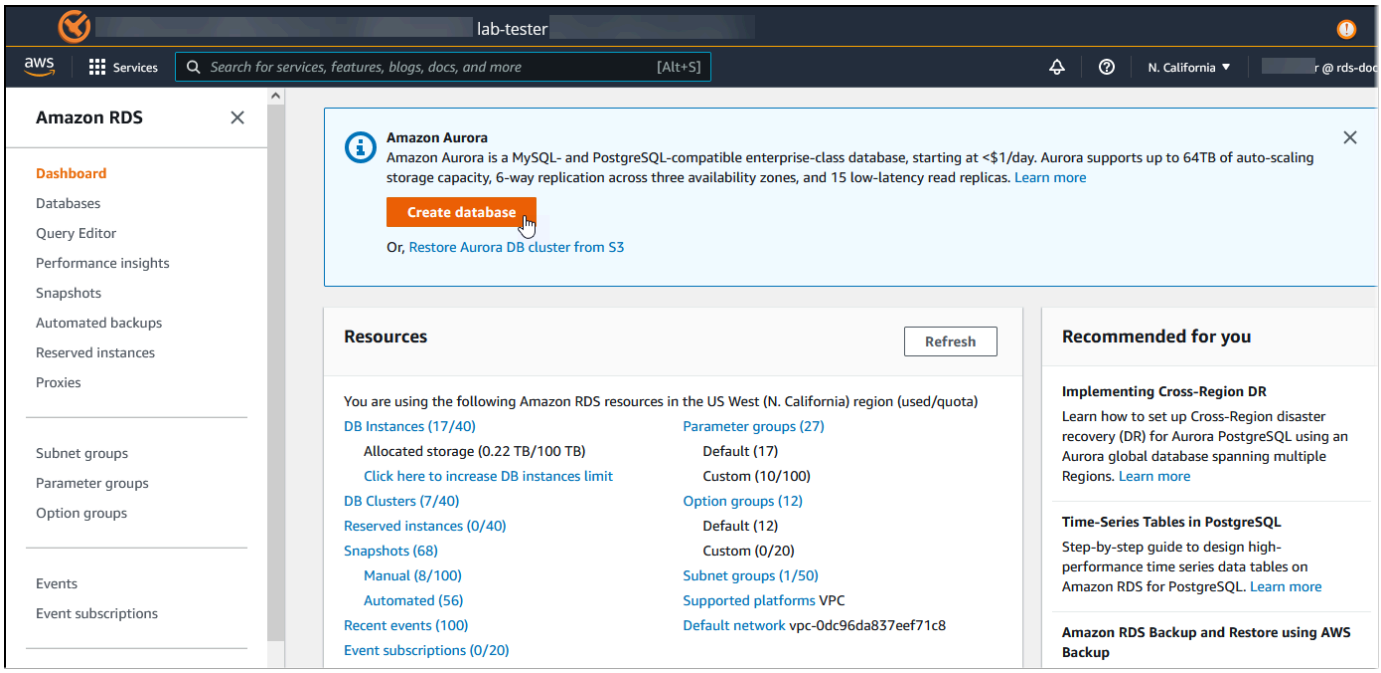
Vous pouvez utiliser la AWS Management Console ou l'interface AWS CLI pour créer un cluster Aurora PostgreSQL doté de Babelfish.

> **Note**
>
> Dans un cluster Aurora PostgreSQL, le nom de base de données `babelfish_db` est réservé à Babelfish. La création de votre propre base de données « babelfish_db » sur Babelfish pour Aurora PostgreSQL empêche Aurora d'allouer Babelfish.

Console

Pour créer un cluster doté de Babelfish à partir de la AWS Management Console

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/ et choisissez Create database (Créer une base de données).



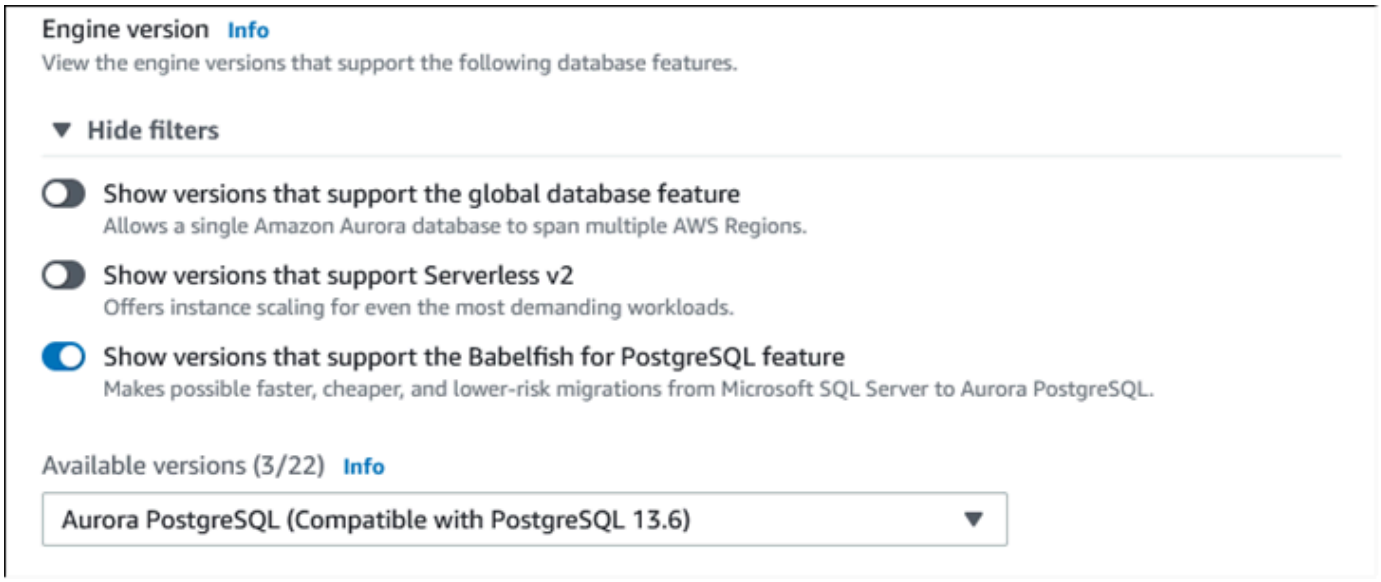
2. Dans Choose a database creation method (Choisir une méthode de création de base de données), effectuez l'une des actions suivantes :

   - Pour spécifier des options de moteur détaillées, choisissez Standard create (Création standard).
   - Pour utiliser des options préconfigurées prenant en charge les bonnes pratiques relatives à un cluster Aurora, choisissez Easy create (Création facile).

3. Pour Type de moteur, choisissez Aurora (compatible avec PostgreSQL).

4. Choisissez Show filters (Afficher les filtres), puis Show versions that support the Babelfish for PostgreSQL feature (Afficher les versions prenant en charge la fonction Babelfish for PostgreSQL) pour répertorier les types de moteurs qui prennent en charge Babelfish. Babelfish est actuellement pris en charge sur Aurora PostgreSQL 13.4 et versions ultérieures.

5. Dans le champ Available versions (Versions disponibles), choisissez une version d'Aurora PostgreSQL. Pour obtenir les dernières fonctionnalités de Babelfish, choisissez la version majeure d'Aurora PostgreSQL la plus élevée.



6. Dans le champ Templates (Modèles), sélectionnez le modèle qui correspond à votre cas d'utilisation.

7. Dans le champ DB cluster identifier (Identifiant du cluster de bases de données), saisissez un nom facile à retrouver plus tard dans la liste des clusters de bases de données.

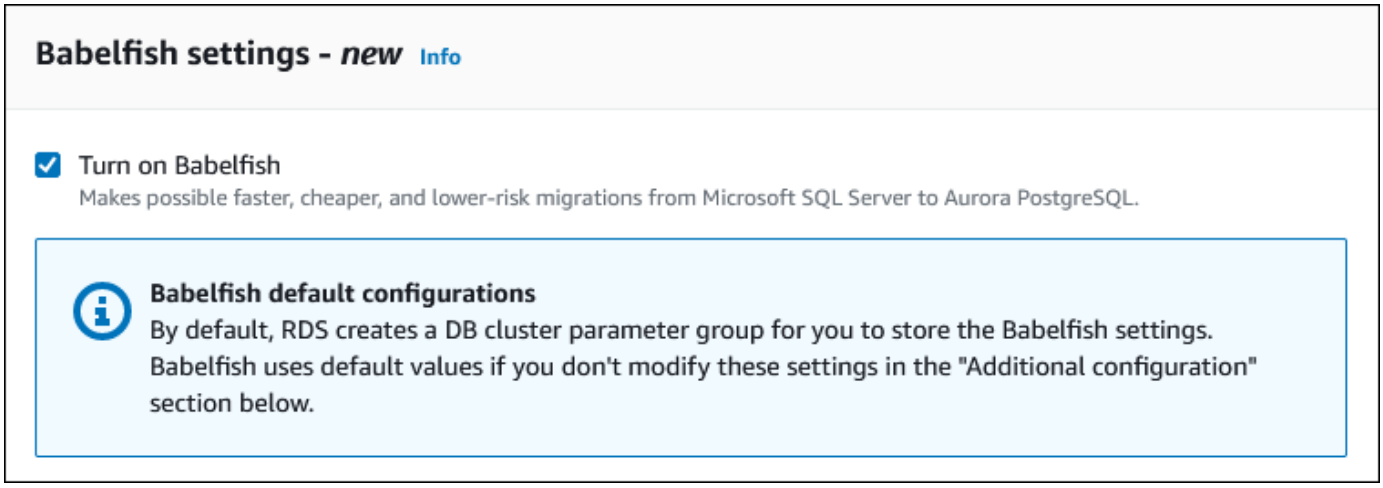
8. Dans le champ Master username (Nom d'utilisateur principal), saisissez un nom d'utilisateur administrateur. La valeur par défaut pour Aurora PostgreSQL est `postgres`. Vous pouvez accepter la valeur par défaut ou choisir un autre nom. Par exemple, pour respecter la convention

de dénomination utilisée sur vos bases de données SQL Server, vous pouvez saisir `sa` (administrateur système) pour le nom d'utilisateur principal.

Si vous ne créez pas d'utilisateur nommé `sa` pour le moment, vous pourrez en créer un plus tard à l'aide du client de votre choix. Après avoir créé l'utilisateur, utilisez la commande `ALTER SERVER ROLE` pour l'ajouter au groupe (rôle) `sysadmin` pour le cluster.

> **Warning**
>
> Le nom d'utilisateur principal doit toujours utiliser des caractères minuscules, faute de quoi le cluster de bases de données ne pourra pas se connecter à Babelfish via le port TDS.

9. Pour Mot de passe principal, créez un mot de passe fort et confirmez le mot de passe.
10. Pour les options suivantes, jusqu'à la section Babelfish settings (Paramètres Babelfish), spécifiez les paramètres de votre cluster de bases de données. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour les clusters de bases de données Aurora.
11. Pour rendre la fonctionnalité Babelfish disponible, cochez la case Turn on Babelfish (Activer Babelfish).



12. Dans le champ DB cluster parameter group (Groupe de paramètres du cluster de bases de données), effectuez l'une des actions suivantes :

    - Choisissez Create new (Créer) pour créer un nouveau groupe de paramètres avec Babelfish activé.
    - Choisissez Choose existing (Choisir un groupe existant) pour utiliser un groupe de paramètres existant. Si vous utilisez un groupe existant, veillez à le modifier avant de créer le cluster et

à attribuer des valeurs aux paramètres Babelfish. Pour en savoir plus sur les paramètres Babelfish, consultez Paramètres du groupe de paramètres de cluster de bases de données pour Babelfish.

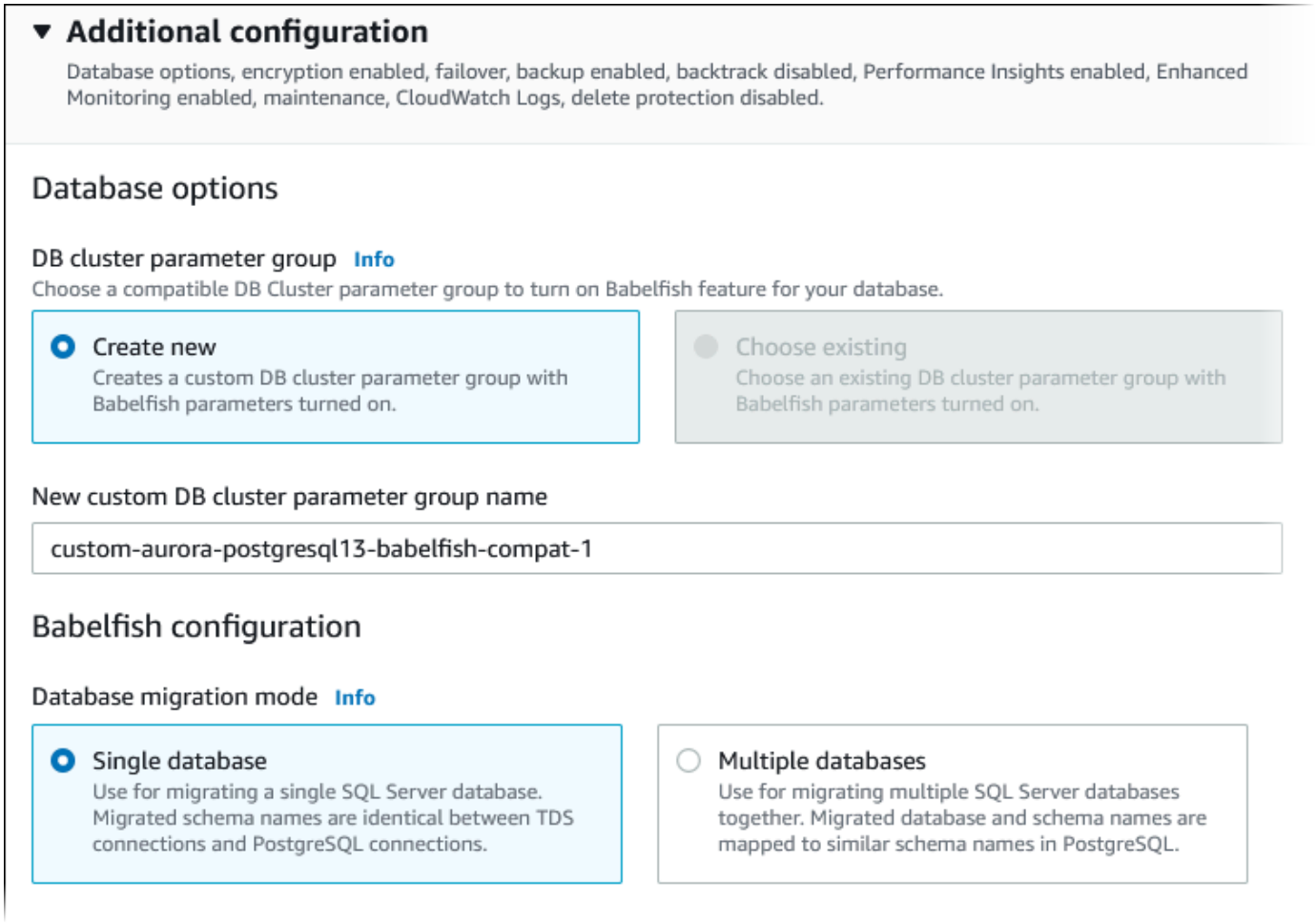
Si vous utilisez un groupe existant, indiquez son nom dans la zone qui suit.

13. Dans le champ Database migration mode (Mode de migration de base de données), choisissez l'une des options suivantes :

    - Single database (Base de données individuelle) pour migrer une base de données SQL Server individuelle.

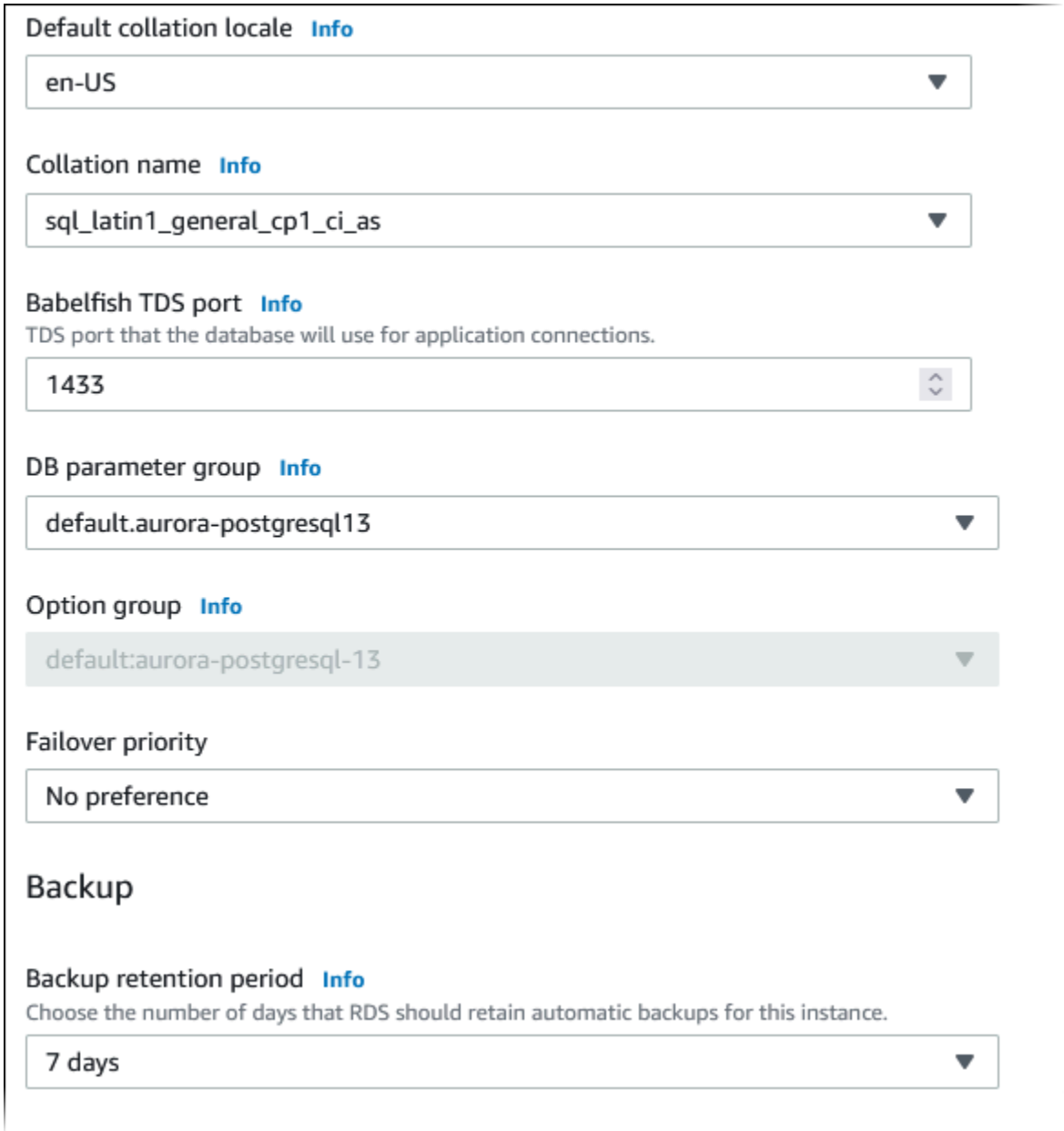
      Dans certains cas, vous pouvez migrer plusieurs bases de données utilisateur ensemble, avec pour objectif final une migration complète vers l'instance native d'Aurora PostgreSQL sans Babelfish. Si les applications finales nécessitent un regroupement des schémas (un seul schéma dbo), commencez par regrouper vos bases de données SQL Server en une seule base de données SQL Server. Migrez ensuite vers Babelfish en utilisant le mode Single database (Base de données individuelle).
    - Multiple databases (Plusieurs bases de données) pour migrer plusieurs bases de données SQL Server (issues d'un même environnement SQL Server). Le mode Plusieurs bases de données ne permet pas de regrouper plusieurs bases de données issues d'environnements SQL Server différents. Pour en savoir plus sur la migration de plusieurs bases de données, consultez Utilisation de Babelfish avec une ou plusieurs bases de données.

> **Note**
>
> À partir de la version Aurora PostgreSQL 16, le mode Plusieurs bases de données est choisi par défaut comme mode de migration de base de données.



14. Dans le champ Default collation locale (Paramètres régionaux du classement par défaut), saisissez les paramètres régionaux de votre serveur. La valeur par défaut est `en-US`. Pour en savoir plus sur les classements, consultez Comprendre les classements dans Babelfish pour Aurora PostgreSQL..
15. Dans le champ Collation name (Nom du classement), saisissez le classement par défaut. La valeur par défaut est `sql_latin1_general_cp1_ci_as`. Pour plus d'informations, consultez Comprendre les classements dans Babelfish pour Aurora PostgreSQL..
16. Pour Port TDS de Babelfish, entrez le port par défaut `1433`. Actuellement, Babelfish prend en charge uniquement le port `1433` pour votre cluster de bases de données.
17. Dans le champ DB parameter group (Groupe de paramètres de base de données), choisissez un groupe de paramètres ou demandez à Aurora de créer un groupe doté des paramètres par défaut.

18. Dans le champ Failover priority (Priorité de basculement), choisissez une priorité de basculement pour l'instance. Si vous ne choisissez aucune valeur, la valeur par défaut est `tier-1`. Cette priorité détermine l'ordre dans lequel les réplicas sont promus lors de la reprise après une défaillance de l'instance principale. Pour plus d'informations, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.
19. Dans le champ Backup retention period (Période de rétention des sauvegardes), choisissez la durée (comprise entre 1 et 35 jours) pendant laquelle Aurora doit conserver les copies de sauvegarde de la base de données. Vous pouvez utiliser les copies de sauvegarde pour les restaurations à un instant dans le passé de votre base de données à la seconde. La période de rétention par défaut est de sept jours.



20. Choisissez Copy tags to snapshots (Copier les identifications dans les instantanés) pour copier toutes les identifications de l'instance de base de données dans un instantané de bases de données lors de la création d'un instantané.

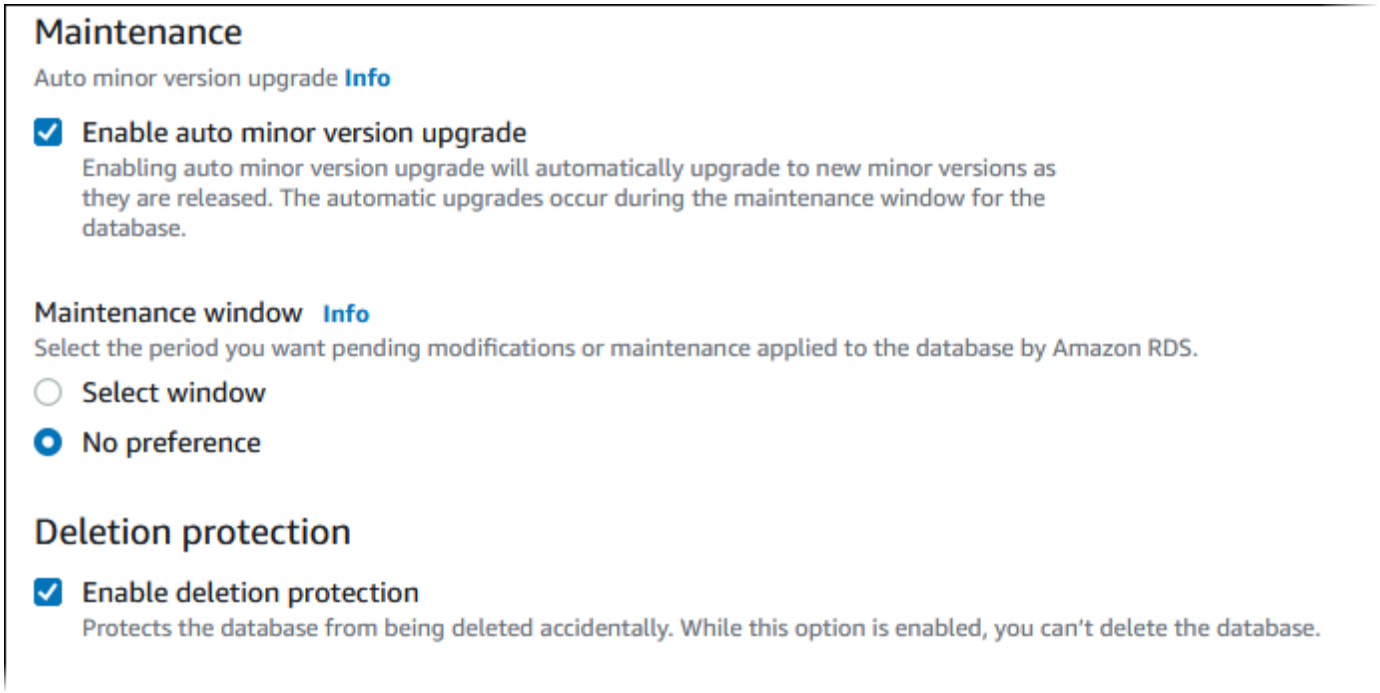
> **Note**
>
> Lorsque vous restaurez un cluster de bases de données à partir d'un instantané, il n'est pas restauré en tant que cluster de bases de données Babelfish pour Aurora PostgreSQL. Vous devez activer les paramètres qui contrôlent les préférences de Babelfish dans le groupe de paramètres du cluster de bases de données pour réactiver Babelfish. Pour plus d'informations sur les paramètres de Babelfish, consultez Paramètres du groupe de paramètres de cluster de bases de données pour Babelfish.

21. Choisissez Enable encryption (Activer le chiffrement) afin d'activer le chiffrement au repos (chiffrement du stockage Aurora) pour ce cluster de bases de données.
22. Choisissez Enable Performance Insights (Activer Performance Insights) pour activer Amazon RDS Performance Insights.
23. Choisissez Enable Enhanced monitoring (Activer la surveillance améliorée) afin de commencer à collecter des métriques en temps réel pour le système d'exploitation sur lequel le cluster de bases de données est exécuté.
24. Choisissez PostgreSQL log (Journal PostgreSQL) pour publier les fichiers journaux dans Amazon CloudWatch Logs.
25. Choisissez Enable auto minor version upgrade (Activer la mise à niveau automatique des versions mineures) pour mettre automatiquement à jour votre cluster de bases de données Aurora lorsqu'une mise à niveau de version mineure est disponible.
26. Dans le champ Maintenance window (Fenêtre de maintenance), procédez comme suit :

    - Pour choisir quand Amazon RDS doit apporter des modifications ou effectuer des opérations de maintenance, choisissez Select window (Sélectionner la fenêtre).
    - Pour effectuer la maintenance d'Amazon RDS à une heure non planifiée, choisissez No preference (Aucune préférence).

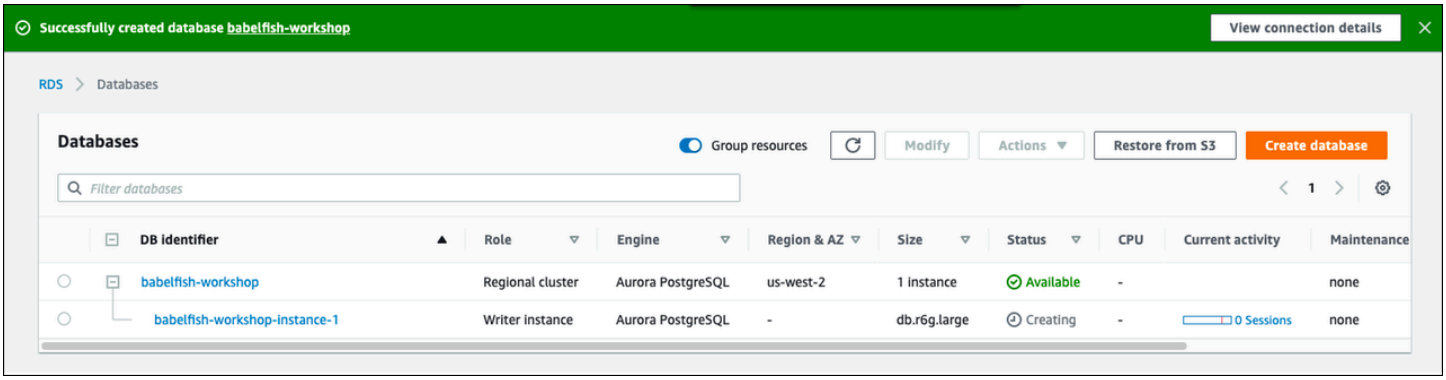
27. Cochez la case Enable deletion protection (Activer la protection contre la suppression) pour protéger votre base de données d'une suppression accidentelle.

Si vous activez cette fonction, il vous est impossible d'effectuer une suppression directe de la base de données. Pour supprimer la base de données, vous devez modifier le cluster de bases de données et désactiver cette fonction.



28. Choisissez Create database (Créer une base de données).

La nouvelle base de données configurée pour Babelfish figure dans la liste Databases (Bases de données). La colonne Status (Statut) indique Available (Disponible) une fois le déploiement terminé.



AWS CLI

Lorsque vous créez un cluster de bases de données Babelfish pour Aurora PostgreSQL à l'aide d'AWS CLI, vous devez transmettre à la commande le nom du groupe de paramètres de cluster de

bases de données à utiliser pour le cluster. Pour plus d'informations, consultez Prérequis des clusters de bases de données.

Avant de pouvoir utiliser l'interface AWS CLI pour créer un cluster Aurora PostgreSQL doté de Babelfish, vous devez effectuer ce qui suit :

- Choisissez l'URL de votre point de terminaison dans la liste des services disponible dans Points de terminaison et quotas Amazon Aurora.
- Créez un groupe de paramètres pour le cluster. Pour plus d'informations sur les groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora.
- Modifiez le groupe de paramètres en ajoutant le paramètre permettant d'activer Babelfish.

Pour créer un cluster de bases de données Aurora PostgreSQL doté de Babelfish à l'aide de l'interface AWS CLI

Les exemples suivants utilisent le nom d'utilisateur principal par défaut, `postgres`. Remplacez-le si nécessaire par le nom d'utilisateur que vous avez créé pour votre cluster de bases de données, tel que `sa`, ou par le nom d'utilisateur que vous avez choisi si vous n'avez pas accepté la valeur par défaut.

1. Créez un groupe de paramètres.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-db-cluster-parameter-group \
   --endpoint-url endpoint-url \
   --db-cluster-parameter-group-name parameter-group \
   --db-parameter-group-family aurora-postgresql14 \
   --description "description"
   ```

   Pour Windows :

   ```
   aws rds create-db-cluster-parameter-group ^
   --endpoint-url endpoint-URL ^
   --db-cluster-parameter-group-name parameter-group ^
   --db-parameter-group-family aurora-postgresql14 ^
   --description "description"
   ```

2. Modifiez votre groupe de paramètres pour activer Babelfish.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
--endpoint-url endpoint-url \
--db-cluster-parameter-group-name parameter-group \
--parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot"
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
--endpoint-url endpoint-url ^
--db-cluster-parameter-group-name paramater-group ^
--parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot"
```

3. Identifiez le groupe de sous-réseaux de bases de données et l'ID du groupe de sécurité du cloud privé virtuel (VPC) de votre nouveau cluster de bases de données, puis appelez la commande create-db-cluster.

   Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
--db-cluster-identifier  cluster-name\
--master-username postgres \
--manage-master-user-password \
--engine aurora-postgresql \
--engine-version 14.3            \
--vpc-security-group-ids security-group \
--db-subnet-group-name subnet-group-name \
--db-cluster-parameter-group-name parameter-group
```

   Pour Windows :

```
aws rds create-db-cluster ^
--db-cluster-identifier cluster-name ^
--master-username postgres ^
--manage-master-user-password ^
--engine aurora-postgresql ^
--engine-version 14.3 ^
--vpc-security-group-ids security-group ^
```

```
--db-subnet-group-name subnet-group ^
--db-cluster-parameter-group-name parameter-group
```

Cet exemple spécifie l'option `--manage-master-user-password` permettant de générer le mot de passe de l'utilisateur principal et de le gérer dans Secrets Manager. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager. Vous pouvez également utiliser l'option `--master-password` pour spécifier et gérer vous-même le mot de passe.

4. Créer de manière explicite l'instance principale pour votre cluster de bases de données. Utilisez le nom du cluster que vous avez créé à l'étape 3 pour l'argument `--db-cluster-identifier` lorsque vous appelez la commande create-db-instance, comme illustré ci-dessous.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-db-instance \
   --db-instance-identifier instance-name \
   --db-instance-class db.r6g \
   --db-subnet-group-name subnet-group \
   --db-cluster-identifier cluster-name \
   --engine aurora-postgresql
   ```

   Pour Windows :

   ```
   aws rds create-db-instance ^
   --db-instance-identifier instance-name ^
   --db-instance-class db.r6g ^
   --db-subnet-group-name subnet-group ^
   --db-cluster-identifier cluster-name ^
   --engine aurora-postgresql
   ```

Migration d’une base de données SQL Server vers Babelfish pour Aurora PostgreSQL

Vous pouvez utiliser Babelfish pour Aurora PostgreSQL pour migrer une base de données SQL Server vers un cluster de bases de données Amazon Aurora PostgreSQL. Avant toute migration, consultez Utilisation de Babelfish avec une ou plusieurs bases de données.

Rubriques

Présentation du processus de migration

Le résumé suivant énumère les étapes nécessaires pour réussir la migration de votre application SQL Server et la faire fonctionner avec Babelfish. Pour plus d’informations sur les outils que vous pouvez utiliser pour les processus d’exportation et d’importation, et pour plus de détails, consultez Outils d’importation/exportation pour la migration de SQL Server vers Babelfish. Pour charger les données, nous vous recommandons d’utiliser AWS DMS avec un cluster de bases de données Aurora PostgreSQL comme point de terminaison cible.

1. Créez un cluster de bases de données Aurora PostgreSQL dans lequel Babelfish est activé. Pour savoir comment procéder, consultez Création d’un cluster de bases de données Babelfish pour Aurora PostgreSQL.

   Pour importer les différents artefacts SQL exportés de votre base de données SQL Server, connectez-vous au cluster Babelfish en utilisant un outil SQL Server tel que sqlcmd. Pour plus d’informations, consultez Utilisation d’un client SQL Server pour se connecter au cluster de bases de données.
2. Sur la base de données SQL Server que vous souhaitez migrer, exportez le langage de définition de données (DDL). Le DDL est un code SQL qui décrit les objets de base de données contenant des données utilisateur (comme des tables, des index et des vues) et du code de base de données écrit par l’utilisateur (comme des procédures stockées, des fonctions définies par l’utilisateur et des déclencheurs).

   Pour plus d’informations, consultez Utilisation de SQL Server Management Studio (SSMS) pour migrer vers Babelfish.

3. Exécutez un outil d'évaluation pour évaluer la portée des modifications dont vous pourriez avoir besoin pour que Babelfish puisse prendre en charge l'application exécutée sur SQL Server. Pour plus d'informations, consultez Évaluation et gestion des différences entre SQL Server et Babelfish.
4. Vérifiez les limites du point de terminaison cible AWS DMS et mettez à jour le script DDL si nécessaire. Pour plus d'informations, consultez les limitations relatives à l'utilisation d'un point de terminaison cible PostgreSQL avec des tables Babelfish dans Utilisation de Babelfish pour Aurora PostgreSQL en tant que cible.
5. Sur votre nouveau cluster de bases de données Babelfish, exécutez la DDL dans votre base de données T-SQL spécifiée pour créer uniquement les schémas, les types de données définis par l'utilisateur, et les tables avec leurs contraintes de clé primaire.
6. Utilisez AWS DMS pour migrer vos données du serveur SQL vers les tables Babelfish. Pour la réplication continue à l'aide de SQL Server Change Data Capture ou SQL Replication, utilisez Aurora PostgreSQL au lieu de Babelfish comme point de terminaison. Pour ce faire, consultez Utilisation de Babelfish pour Aurora PostgreSQL en tant que cible pour AWS Database Migration Service.
7. Lorsque le chargement des données est terminé, créez tous les objets T-SQL restants qui prennent en charge l'application sur votre cluster Babelfish.
8. Reconfigurez l'application cliente pour qu'elle se connecte au point de terminaison Babelfish au lieu de votre base de données SQL Server. Pour plus d'informations, consultez Connexion à un cluster de bases de données Babelfish.
9. Modifiez l'application si nécessaire et procédez à un nouveau test. Pour plus d'informations, consultez Différences entre Babelfish pour Aurora PostgreSQL et SQL Server.

Vous devez toujours évaluer vos requêtes SQL côté client. Les schémas générés à partir de votre instance SQL Server convertissent uniquement le code SQL côté serveur. Nous vous recommandons d'effectuer les étapes suivantes :

- Capturez les requêtes côté client à l'aide de SQL Server Profiler avec le modèle prédéfini TSQL_Replay. Ce modèle capture les informations des instructions T-SQL que vous pouvez ensuite lire à nouveau pour des réglages et des tests itératifs. Vous pouvez démarrer Profiler dans SQL Studio Management Studio, à partir du menu Tools (Outils). Choisissez SQL Server Profiler pour ouvrir Profiler et choisissez le modèle TSQL_Replay.

  Pour l'utiliser pour votre migration Babelfish, démarrez une trace, puis exécutez votre application à l'aide de vos tests fonctionnels. Profiler capture les instructions T-SQL. Une fois que vous avez terminé le test, arrêtez la trace. Enregistrez le résultat dans un fichier XML avec vos requêtes côté

client (File > Save as > Trace XML File for Replay) (Fichier > Enregistrer sous > Tracer le fichier XML pour le relire).

Pour plus d'informations, consultez SQL Server Profiler dans la documentation Microsoft. Pour plus d'informations sur le modèle TSQL_Replay, consultez Modèles du Générateur de profils SQL Server.

- Pour les applications comportant des requêtes SQL complexes côté client, nous vous recommandons d'utiliser Babelfish Compass pour les analyser afin de vérifier la compatibilité de ces requêtes avec Babelfish. Si l'analyse indique que les instructions SQL côté client contiennent des fonctions SQL non prises en charge, examinez les aspects SQL de l'application cliente et modifiez-les si nécessaire.
- Vous pouvez également capturer les requêtes SQL en tant qu'événements étendus (format .xel). Pour ce faire, utilisez SSMS XEvent Profiler. Après avoir généré le fichier .xel, extrayez les instructions SQL dans des fichiers .xml que Compass peut ensuite traiter. Pour plus d'informations, consultez Use the SSMS XEvent Profiler (Utiliser SSMS XEvent Profiler) dans la documentation Microsoft.

Lorsque vous êtes satisfait de tous les tests, de toutes les analyses et de toutes les modifications nécessaires pour votre application migrée, vous pouvez commencer à utiliser votre base de données Babelfish en production. Pour ce faire, arrêtez la base de données d'origine et redirigez les applications clientes en direct pour utiliser le port Babelfish TDS.

> **Note**
>
> AWS DMS prend désormais en charge la réplication des données de Babelfish. Pour plus d'informations, consultez AWS DMS prend désormais en charge Babelfish pour Aurora PostgreSQL en tant que cible.

Évaluation et gestion des différences entre SQL Server et Babelfish

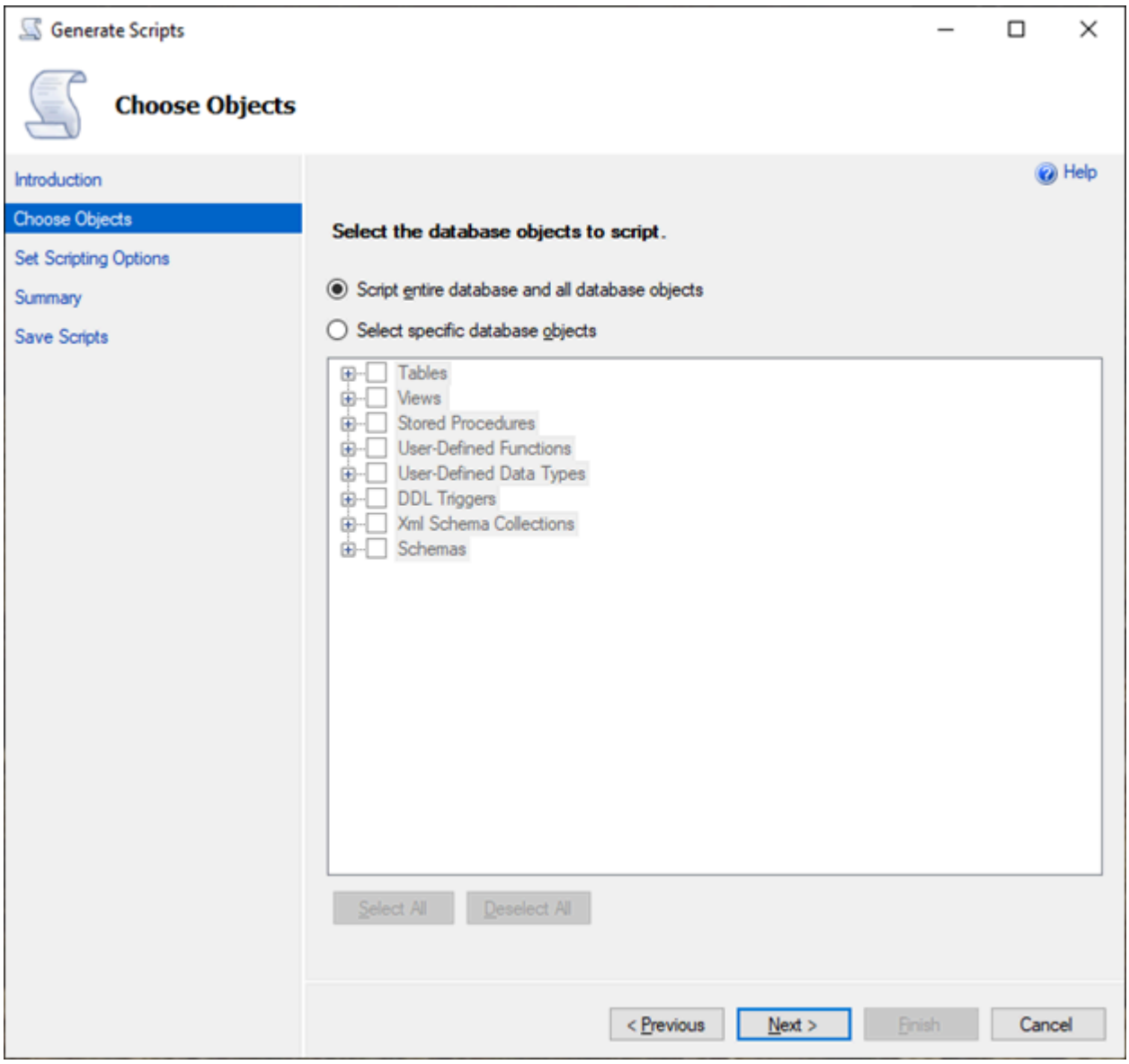
Pour obtenir de meilleurs résultats, nous vous recommandons d'évaluer le DDL/DML généré et le code de la requête client avant de migrer votre application de base de données SQL Server vers Babelfish. Selon la version de Babelfish et les fonctionnalités spécifiques de SQL Server implémentées par votre application, vous devrez peut-être refactoriser votre application ou utiliser des alternatives aux fonctionnalités qui ne sont pas encore entièrement prises en charge dans Babelfish.

- Pour évaluer le code de votre application SQL Server, utilisez Babelfish Compass sur la DDL générée pour déterminer la quantité de code T-SQL prise en charge par Babelfish. Identifiez le code T-SQL qui pourrait nécessiter des modifications avant d'être exécuté sur Babelfish. Pour plus d'informations sur cet outil, consultez la page relative à l'outil Babelfish Compass sur GitHub.

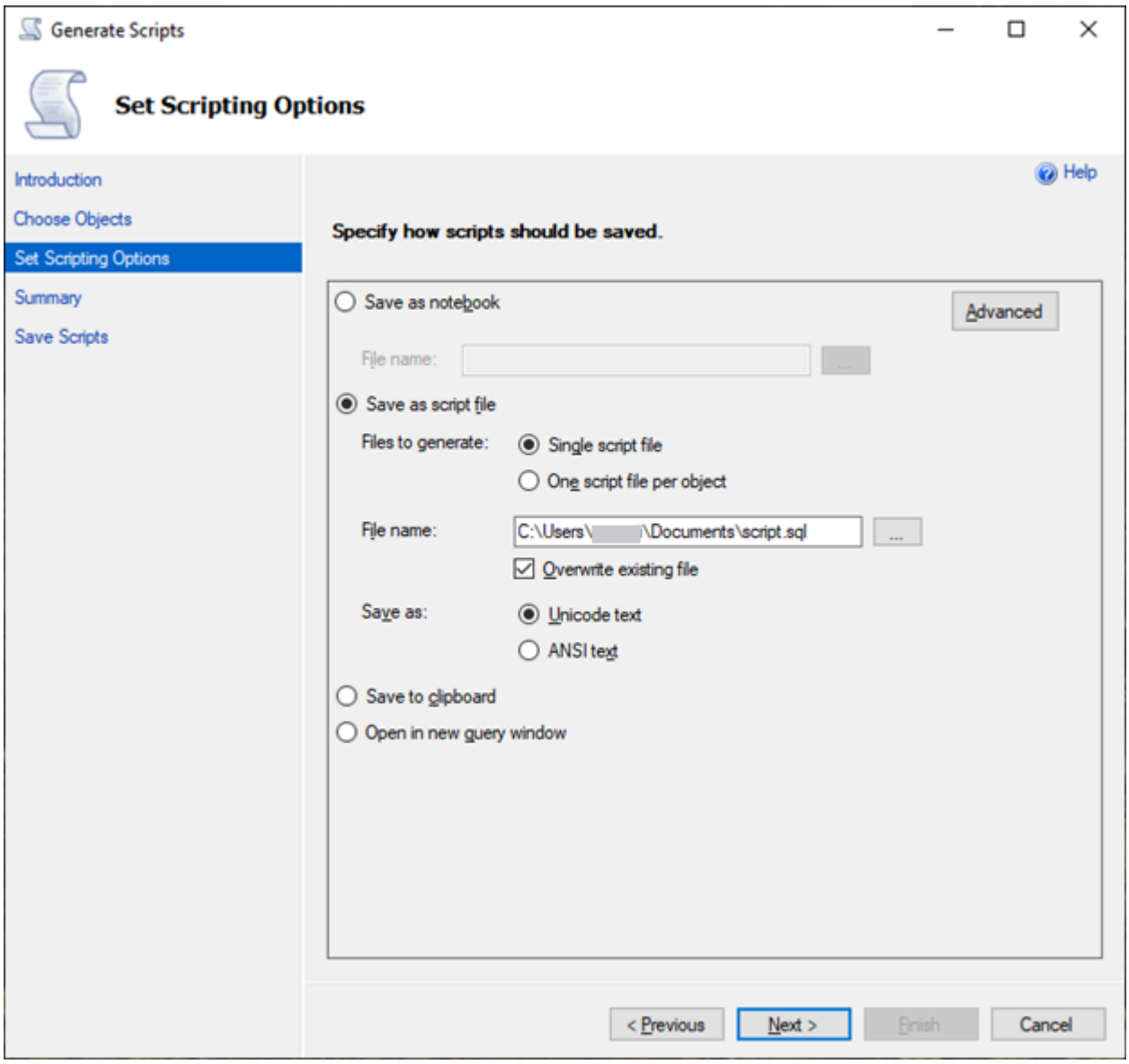
> **Note**
>
> Babelfish Compass est un outil open source. Signalez tout problème avec Babelfish Compass via GitHub plutôt que via AWS Support.

Vous pouvez utiliser l'assistant Generate Script (Génération de scripts) avec SQL Server Management Studio (SSMS) pour générer le fichier SQL qui est évalué par Babelfish Compass ou CLI AWS Schema Conversion Tool. Nous recommandons les étapes suivantes pour rationaliser l'évaluation.

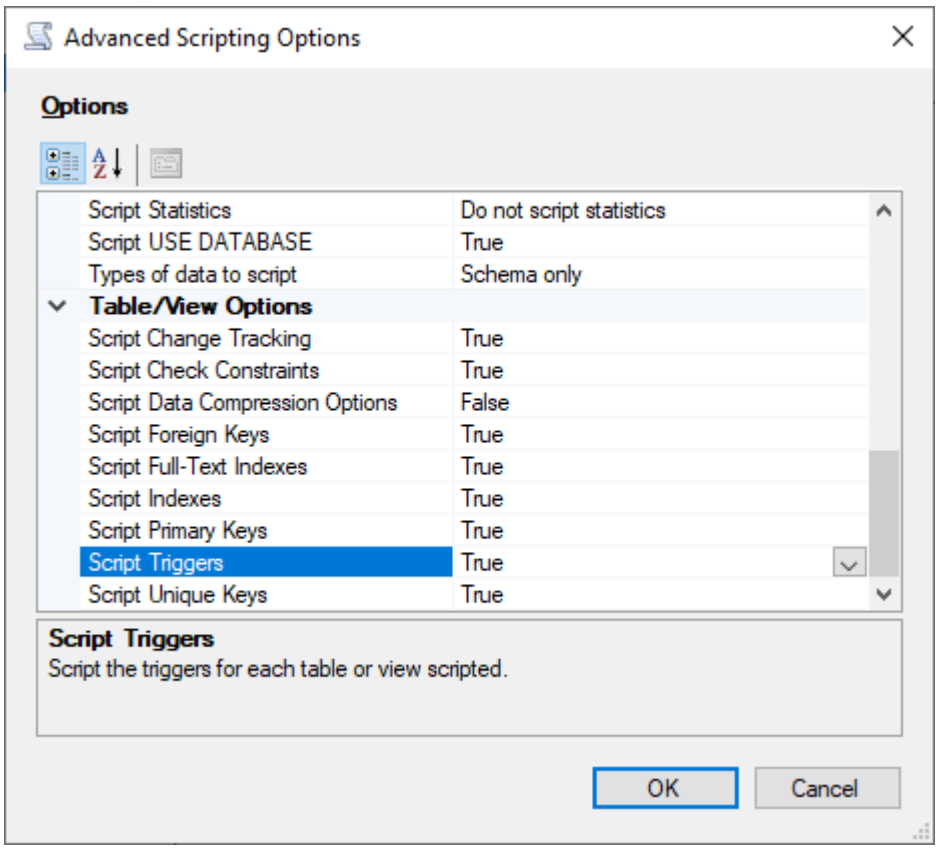
1. Sur la page Choose Objects (Choisir des objets), sélectionnez Script entire database and all database objects (Script de la base de données entière et tous les objets de la base de données).



2. Pour l’option Set Scripting Options (Définir les options de script), choisissez Save as script file (Enregistrer comme fichier de script) en tant que Single script file (Fichier de script unique).



3. Choisissez Advanced (Avancé) pour modifier les options de script par défaut afin d'identifier les fonctionnalités qui sont normalement définies sur faux pour une évaluation complète :
   - Option Script Change Tracking définie sur True
   - Option Script Full-Text Indexes définie sur True
   - Option Script Triggers définie sur True
   - Option Script Logins définie sur True
   - Option Script Owner définie sur True
   - Option Script Object-Level Permissions définie sur True
   - Option Script Collations définie sur True



4. Effectuez les étapes restantes de l'assistant pour générer le fichier.

Outils d'importation/exportation pour la migration de SQL Server vers Babelfish

Nous vous recommandons d'utiliser AWS DMS comme outil principal pour la migration de SQL Server vers Babelfish. Cependant, Babelfish prend en charge plusieurs autres façons de migrer les données à l'aide d'outils SQL Server, dont les suivants.

- SQL Server Integration Services (SSIS) pour toutes les versions de Babelfish. Pour obtenir plus d'informations, consultez Migrate from SQL Server to Aurora PostgreSQL using SSIS and Babelfish (Migration de SQL Server vers Aurora PostgreSQL en utilisant SSIS et Babelfish).
- Utilisez l'assistant d'importation/exportation SSMS pour les versions 2.1.0 et ultérieures de Babelfish. Cet outil est disponible via SSMS, mais également en tant qu'outil autonome. Pour plus d'informations, consultez Assistant Importation et Exportation SQL Server dans la documentation Microsoft.
- L'utilitaire Microsoft bulk data copy (bcp) vous permet de copier les données d'une instance Microsoft SQL Server vers un fichier de données au format que vous spécifiez. Pour plus d'informations, consultez Utilitaire bcp dans la documentation Microsoft. Babelfish prend désormais

en charge la migration des données à l'aide du client BCP et l'utilitaire bcp prend désormais en charge l'indicateur `-E` (pour les colonnes d'identité) et l'indicateur -b (pour les insertions en lot). Certaines options de bcp ne sont pas prises en charge, notamment `-C`, `-T`, `-G`, `-K`, `-R`, `-V` et `-h`.

Utilisation de SQL Server Management Studio (SSMS) pour migrer vers Babelfish

Nous recommandons de générer des fichiers séparés pour chacun des types d'objets spécifiques. Vous pouvez utiliser l'assistant Generate Scripts (Génération de scripts) dans SSMS pour chaque ensemble d'instructions DDL d'abord, puis modifier les objets en tant que groupe pour résoudre tous les problèmes trouvés pendant l'évaluation.

Effectuez ces étapes pour migrer les données en utilisant AWS DMS ou d'autres méthodes de migration de données. Exécutez d'abord ces types de script de création pour une approche plus efficace et plus rapide du chargement des données sur les tables Babelfish dans Aurora PostgreSQL.

1. Exécutez les instructions `CREATE SCHEMA`.
2. Exécutez les instructions `CREATE TYPE` pour créer des types de données définis par l'utilisateur.
3. Exécutez les instructions `CREATE TABLE` de base avec les clés primaires ou les contraintes uniques.

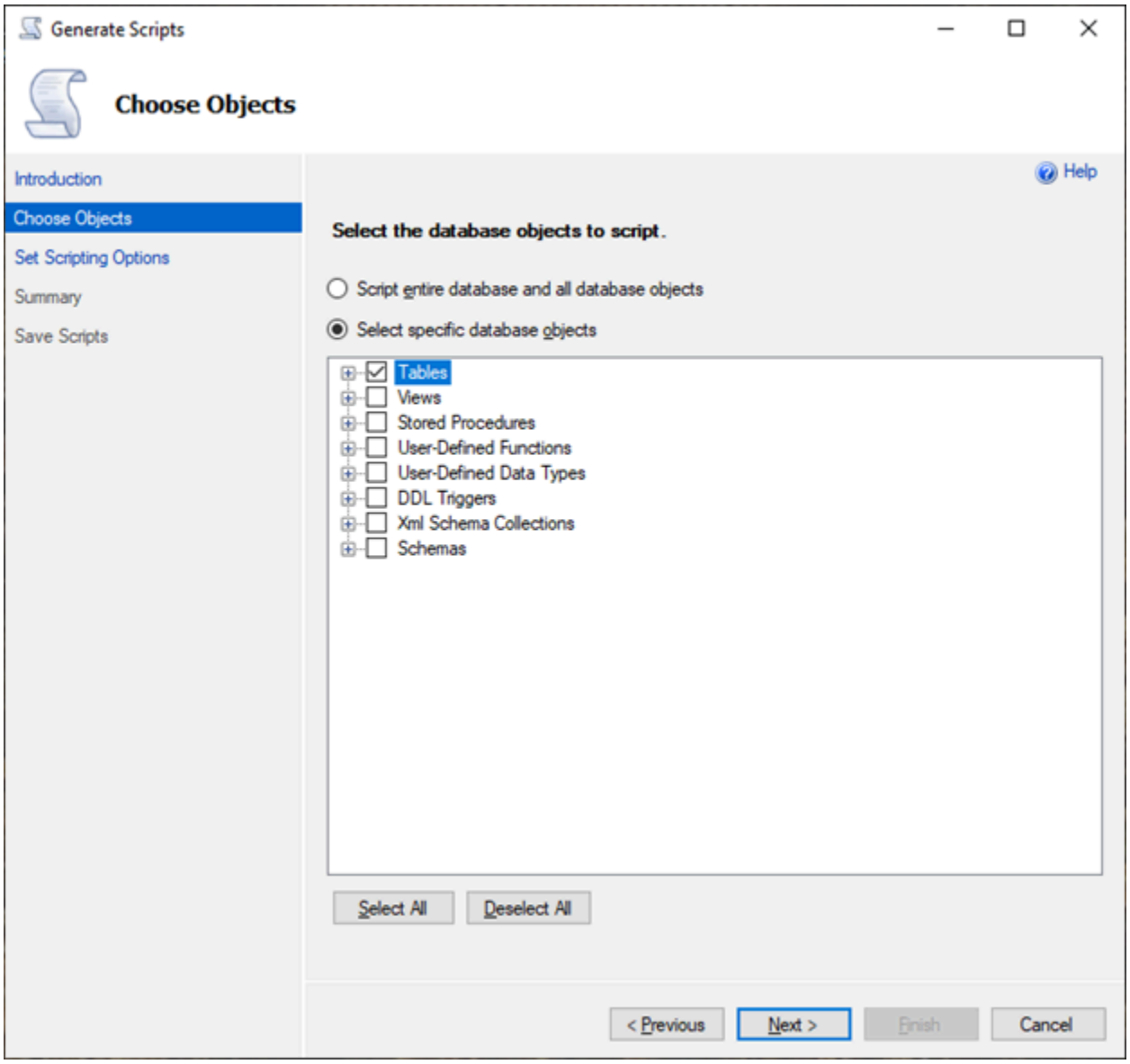
Effectuez le chargement des données en utilisant l'outil d'importation/exportation recommandé. Exécutez les scripts modifiés pour les étapes suivantes afin d'ajouter les objets de base de données restants. Vous avez besoin des instructions de création de table pour exécuter ces scripts pour les contraintes, les déclencheurs et les index. Une fois les scripts générés, supprimez les instructions de création de table.

1. Exécutez les instructions `ALTER TABLE` pour les contraintes de contrôle, les contraintes de clé étrangère, les contraintes par défaut.
2. Exécutez les instructions `CREATE TRIGGER`.
3. Exécutez les instructions `CREATE INDEX`.
4. Exécutez les instructions `CREATE VIEW`.
5. Exécutez les instructions `CREATE STORED PROCEDURE`.

Pour générer des scripts pour chaque type d'objet

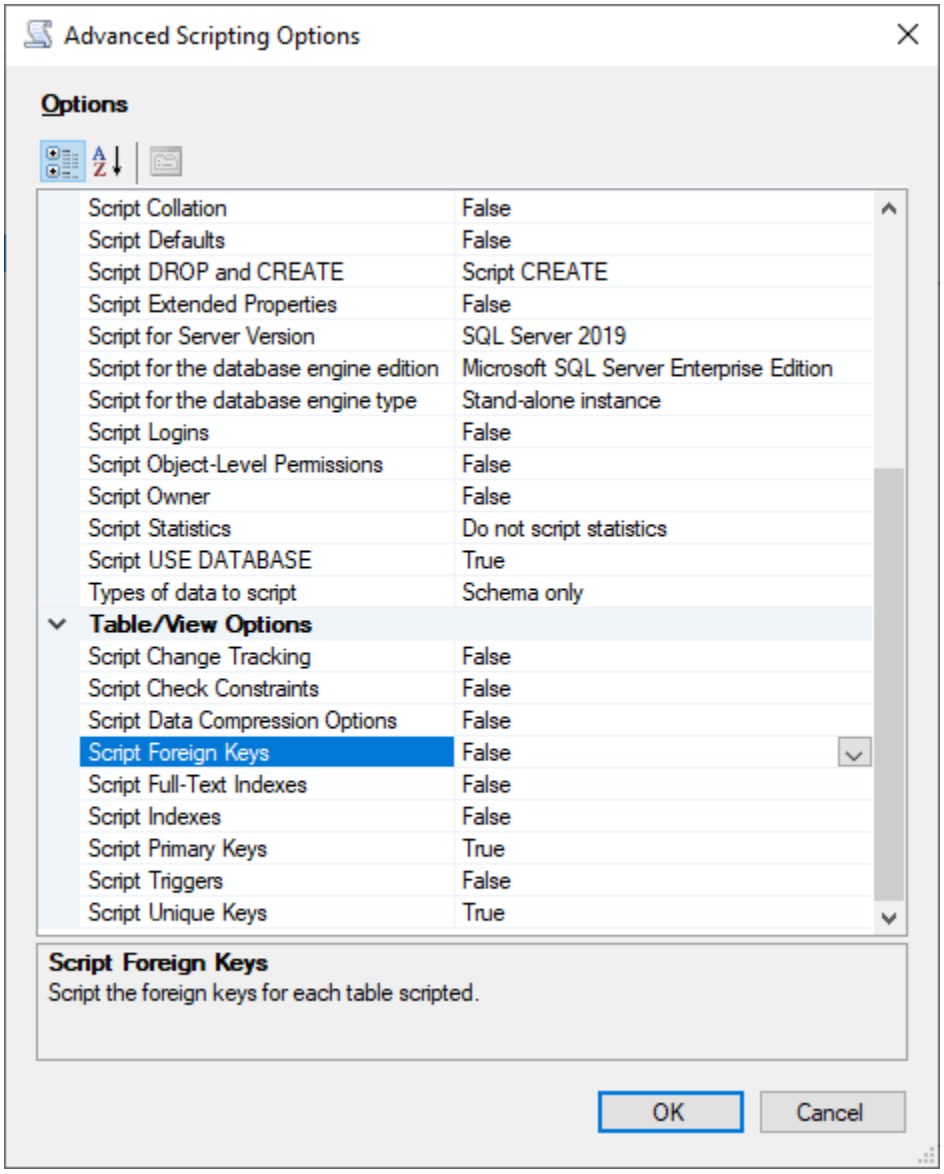
Suivez les étapes suivantes pour créer les instructions de base de création de table à l'aide de l'assistant Generate Scripts (Génération de scripts) dans SSMS. Suivez les mêmes étapes pour générer des scripts pour les différents types d'objets.

1. Connectez-vous à votre instance SQL Server existante.
2. Ouvrez le menu contextuel (clic droit) à partir d'un nom de base de données.
3. Choisissez Tasks (Tâches), puis Generate Scripts... (Générer des scripts...).
4. Dans le panneau Choose Objects (Choisir des objets), choisissez Select specific database objects (Sélectionner des objets de base de données spécifiques). Choisissez Tables, puis sélectionnez toutes les tables. Choisissez Next (Suivant) pour continuer.



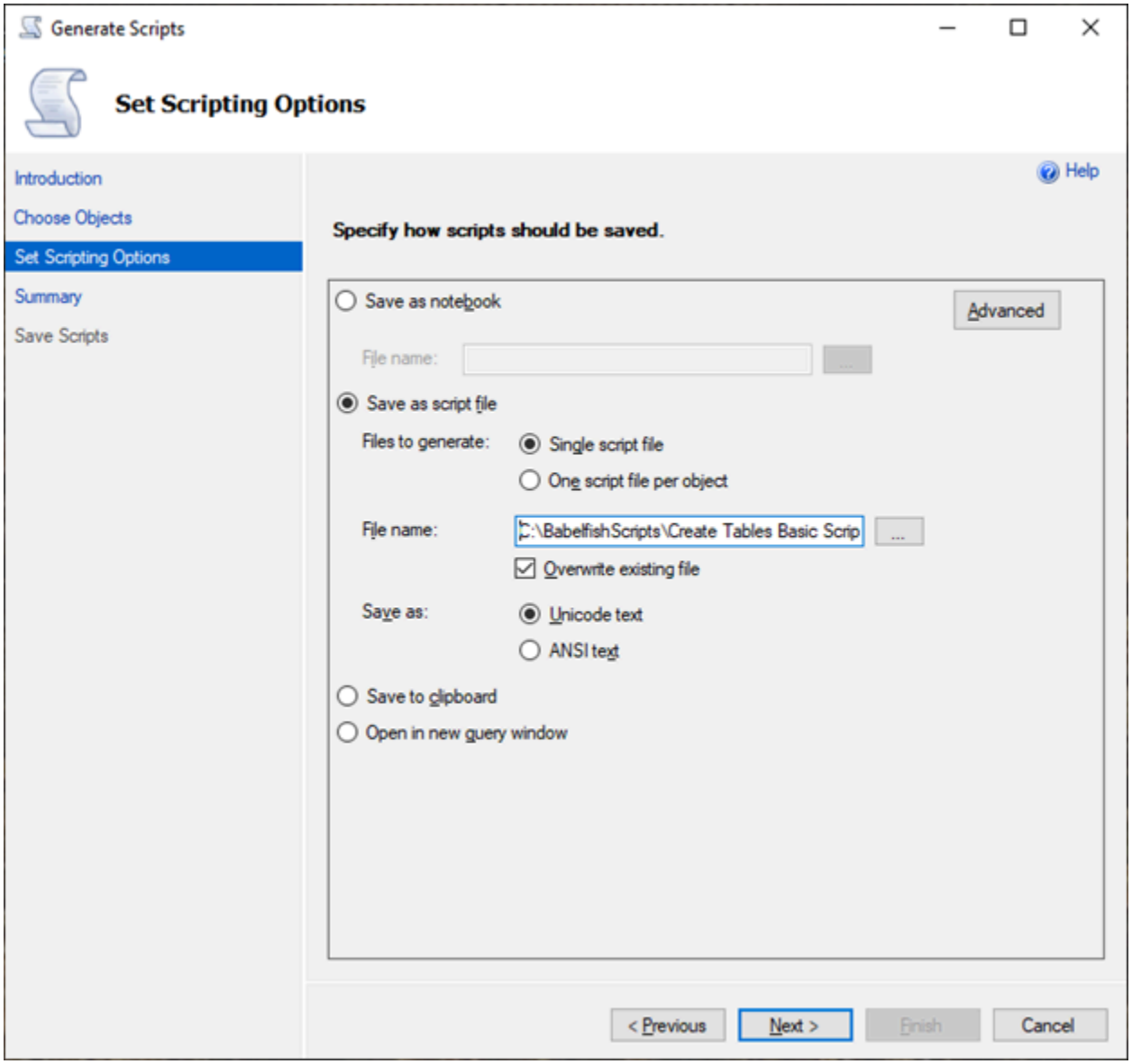
5. Sur la page Set Scripting Options (Définir les options de script), choisissez Advanced (Avancé) pour ouvrir les paramètres Options. Pour générer les instructions de base de création de table, modifiez les valeurs par défaut suivantes :

   - Option Script Defaults définie sur False.
   - Option Script Extended Properties définie sur False. Babelfish ne prend pas en charge les propriétés étendues.
   - Option Script Check Constraints définie sur False. Option Script Foreign Keys définie sur False.



6. Choisissez OK.

7. Sur la page Set Scripting Options (Définir les options de script), choisissez Save as script file (Enregistrer comme fichier de script), puis choisissez l'option Single script file (Fichier de script unique). Saisissez votre File name (Nom de fichier).



8. Choisissez Next (Suivant) pour afficher la page Summary wizard (Assistant Résumé).
9. Choisissez Next (Suivant) pour lancer la génération du script.

   Vous pouvez continuer à générer des scripts pour les autres types d'objets dans l'assistant. Au lieu de choisir Finish (Terminer) après l'enregistrement du fichier, cliquez trois fois sur le bouton Previous (Précédent) pour revenir à la page Choose Objects (Choisir des objets). Répétez ensuite les étapes de l'assistant pour générer des scripts pour les autres types d'objets.

Authentification d'une base de données avec Babelfish for Aurora PostgreSQL

Babelfish for Aurora PostgreSQL prend en charge deux façons d'authentifier les utilisateurs d'une base de données. L'authentification par mot de passe est disponible par défaut pour tous les clusters de bases de données Babelfish. Vous pouvez également ajouter l'authentification Kerberos pour le même cluster de bases de données.

Rubriques

Authentification par mot de passe avec Babelfish

Babelfish for Aurora PostgreSQL prend en charge l'authentification par mot de passe. Les mots de passe sont stockés sous forme chiffrée sur le disque. Pour en savoir plus sur l'authentification sur un cluster Aurora PostgreSQL, consultez Sécurité avec Amazon Aurora PostgreSQL.

Vous pouvez être invité à fournir des informations d'identification chaque fois que vous vous connectez à Babelfish. Tout utilisateur migré vers ou créé sur Aurora PostgreSQL peut utiliser les mêmes informations d'identification sur le port SQL Server et sur le port PostgreSQL. Babelfish n'applique aucune politique de mot de passe, mais nous vous invitons à suivre les recommandations ci-dessous :

- Exiger un mot de passe complexe d'au moins huit (8) caractères.
- Appliquer une politique d'expiration des mots de passe.

Pour consulter la liste complète des utilisateurs de la base de données, utilisez la commande `SELECT * FROM pg_user;`.

Authentification Kerberos avec Babelfish

La version 15.2 de Babelfish for Aurora PostgreSQL prend en charge l'authentification auprès de votre cluster de bases de données à l'aide de Kerberos. Cette méthode vous permet d'utiliser l'authentification Microsoft Windows pour authentifier les utilisateurs lorsqu'ils se connectent à votre base de données Babelfish. Pour ce faire, vous devez d'abord configurer votre cluster de bases de données afin qu'il utilise AWS Directory Service for Microsoft Active Directory pour l'authentification Kerberos. Pour plus d'informations, consultez Qu'est-ce qu'Directory Service ? dans le Guide de l'utilisateur AWS Directory Service.

Configuration de l'authentification Kerberos

Le cluster de bases de données Babelfish for Aurora PostgreSQL peut se connecter via deux ports différents, mais la configuration de l'authentification Kerberos est un processus unique. Par conséquent, vous devez d'abord configurer l'authentification Kerberos pour votre cluster de bases de données. Pour plus d'informations, consultez Configuration de l'authentification Kerberos. Une fois la configuration terminée, assurez-vous de pouvoir vous connecter à un client PostgreSQL à l'aide de Kerberos. Pour plus d'informations, consultez Connexion avec l'authentification Kerberos.

Connexion et mise en service des utilisateurs dans Babelfish

Les connexions Windows créées à partir du port TDS (Tabular Data Stream) peuvent être utilisées avec le port TDS ou le port PostgreSQL. Tout d'abord, la connexion qui peut utiliser Kerberos pour l'authentification doit être provisionnée à partir du port TDS avant d'être utilisée par les utilisateurs et les applications T-SQL pour se connecter à une base de données Babelfish. Lors de la création de connexions Windows, les administrateurs peuvent fournir la connexion en utilisant le nom de domaine DNS ou le nom de domaine NetBIOS. Généralement, le domaine NetBIOS est le sous-domaine du nom de domaine DNS. Par exemple, si le nom de domaine DNS est `CORP.EXAMPLE.COM`, le domaine NetBIOS peut être `CORP`. Si le format du nom de domaine NetBIOS est fourni pour une connexion, un mappage doit exister avec le nom de domaine DNS.

Gestion du mappage entre le nom de domaine NetBIOS et le nom de domaine DNS

Pour gérer les mappages entre le nom de domaine NetBIOS et le nom de domaine DNS, Babelfish fournit des procédures stockées dans le système pour ajouter, supprimer et tronquer des mappages. Seul un utilisateur doté d'un rôle `sysadmin` peut exécuter ces procédures.

Pour créer un mappage entre un nom de domaine NetBIOS et DNS, utilisez la procédure `babelfish_add_domain_mapping_entry` stockée dans le système fournie par Babelfish. Les deux arguments doivent avoir une valeur valide et ne pas être NULL.

Example

```
EXEC babelfish_add_domain_mapping_entry 'netbios_domain_name',
 'fully_qualified_domain_name'
```

L'exemple suivant montre comment créer le mappage entre le nom NetBIOS CORP et le nom de domaine DNS CORP.EXAMPLE.COM.

Example

```
EXEC babelfish_add_domain_mapping_entry 'corp', 'corp.example.com'
```

Pour supprimer une entrée de mappage existante, utilisez la procédure babelfish_remove_domain_mapping_entry stockée dans le système.

Example

```
EXEC babelfish_remove_domain_mapping_entry 'netbios_domain_name'
```

L'exemple suivant montre comment supprimer le mappage pour le nom NetBIOS CORP.

Example

```
EXEC babelfish_remove_domain_mapping_entry 'corp'
```

Pour supprimer une entrée de mappage existante, utilisez la procédure babelfish_remove_domain_mapping_entry stockée dans le système :

Example

```
EXEC babelfish_truncate_domain_mapping_table
```

Pour afficher tous les mappages entre le nom de domaine NetBIOS et DNS, utilisez la requête suivante.

Example

```
SELECT netbios_domain_name, fq_domain_name FROM babelfish_domain_mapping;
```

Gestion des connexions

Créer des connexions

Connectez-vous à la base de données via le point de terminaison TDS à l'aide d'une connexion disposant des autorisations appropriées. Si aucun utilisateur de la base de données n'a été créé pour la connexion, celle-ci est mappée à l'utilisateur invité. Si l'utilisateur invité n'est pas activé, la tentative de connexion échoue.

Créez une connexion Windows à l'aide de la requête suivante. L'option `FROM WINDOWS` permet l'authentification avec Active Directory.

```
CREATE LOGIN login_name FROM WINDOWS [WITH DEFAULT_DATABASE=database]
```

Example

L'exemple suivant montre comment créer une connexion pour l'utilisateur Active Directory [corp\test1] avec la base de données par défaut db1.

```
CREATE LOGIN [corp\test1] FROM WINDOWS WITH DEFAULT_DATABASE=db1
```

Cet exemple suppose qu'il existe un mappage entre le domaine NetBIOS CORP et le nom de domaine DNS CORP.EXAMPLE.COM. S'il n'y a aucun mappage, vous devez fournir le nom de domaine DNS [CORP.EXAMPLE.COM\test1].

> **Note**
>
> Les connexions basées sur les utilisateurs Active Directory sont limitées à des noms de moins de 21 caractères.

Supprimer la connexion

Pour supprimer une connexion, utilisez la même syntaxe que pour n'importe quelle connexion, comme indiqué dans l'exemple suivant :

```
DROP LOGIN [DNS domain name\login]
```

Modifier la connexion

Pour modifier une connexion, utilisez la même syntaxe que pour n'importe quelle connexion, comme dans l'exemple suivant :

```
ALTER LOGIN [DNS domain name\login] { ENABLE|DISABLE|WITH DEFAULT_DATABASE=[master] }
```

La commande ALTER LOGIN prend en charge des options limitées pour les connexions Windows, notamment les suivantes :

- DISABLE : pour désactiver une connexion. Vous ne pouvez pas utiliser une connexion désactivée pour vous authentifier.
- ENABLE : pour activer une connexion désactivée.
- DEFAULT_DATABASE : pour modifier la base de données par défaut d'une connexion.

> **Note**
>
> Toute la gestion des mots de passe est effectuée via Directory Service. La commande ALTER LOGIN n'autorise donc pas les administrateurs de base de données à modifier ou à définir des mots de passe pour les connexions Windows.

Connexion à Babelfish for Aurora PostgreSQL avec une authentification Kerberos

En général, les utilisateurs de base de données qui s'authentifient à l'aide de Kerberos le font à partir de leurs machines clientes. Ces machines sont membres du domaine Active Directory. Ils utilisent l'authentification Windows à partir de leurs applications clientes pour accéder au serveur Babelfish for Aurora PostgreSQL sur le port TDS.

Connexion à Babelfish for Aurora PostgreSQL sur le port PostgreSQL avec une authentification Kerberos

Vous pouvez utiliser des connexions créées à partir du port TDS avec le port TDS ou le port PostgreSQL. Cependant, PostgreSQL utilise par défaut des comparaisons sensibles à la casse pour les noms d'utilisateur. Pour qu'Aurora PostgreSQL interprète les noms d'utilisateur Kerberos sans sensibles à la casse, vous devez définir le paramètre `krb_caseins_users` comme `true` dans le groupe de paramètres du cluster Babelfish personnalisé. Ce paramètre est défini sur `false` par défaut. Pour plus d'informations, consultez Configuration des noms d'utilisateur insensibles

à la casse. En outre, vous devez spécifier le nom d'utilisateur de la connexion sous le format <login@DNS domain name> depuis les applications clientes PostgreSQL. Vous ne pouvez pas utiliser le format <DNS domain name\login>.

Erreurs fréquentes

Vous pouvez configurer une relation d'approbation de forêt entre votre annuaire Microsoft Active Directory sur site et l'AWS Managed Microsoft AD. Pour plus d'informations, consultez Créer une relation de confiance. Vous devez ensuite vous connecter à l'aide d'un point de terminaison spécifique au domaine spécialisé au lieu d'utiliser le domaine Amazon `rds.amazonaws.com` dans le point de terminaison hôte. Si vous n'utilisez pas le point de terminaison spécifique au domaine approprié, il se peut que vous receviez l'erreur suivante :

```
Error: “Authentication method "NTLMSSP" not supported (Microsoft SQL Server, Error:
 514)"
```

Cette erreur se produit lorsque le client TDS ne peut pas mettre en cache le ticket de service pour l'URL de point de terminaison fournie. Pour plus d'informations, consultez Connexion avec Kerberos.

Configuration de l’authentification Kerberos à l’aide de groupes de sécurité Active Directory pour Babelfish

À partir de la version 4.2.0 de Babelfish, vous pouvez configurer l’authentification Kerberos pour Babelfish à l’aide de groupes de sécurité Active Directory. Voici les conditions préalables à remplir pour configurer l’authentification Kerberos à l’aide d’Active Directory :

- Vous devez suivre toutes les étapes mentionnées sous Authentification Kerberos avec Babelfish.
- Assurez-vous que l’instance de base de données est associée à Active Directory. Vous pouvez consulter le statut de l’appartenance au domaine pour l’instance de base de données dans la console ou en exécutant la commande describe-db-instances de l’AWS CLI.

  L’état de l’instance de base de données doit être compatible avec Kerberos. Pour plus d’informations sur l’identification de l’appartenance à un domaine, consultez Présentation de l’appartenance au domaine.
- Vérifiez les mappages entre le nom de domaine NetBIOS et le nom de domaine DNS à l’aide de la requête suivante :

```
SELECT netbios_domain_name, fq_domain_name FROM babelfish_domain_mapping;
```

- Avant de poursuivre, vérifiez que l'authentification Kerberos à l'aide d'un identifiant individuel fonctionne comme prévu. La connexion utilisant l'authentification Kerberos en tant qu'utilisateur Active Directory devrait aboutir. Si vous rencontrez des problèmes, consultez Erreurs fréquentes.

Configuration de l'extension pg_ad_mapping

Vous devez suivre toutes les étapes mentionnées sous Configuration de l'extension pg_ad_mapping . Pour vérifier que l'extension est installée, exécutez la requête suivante à partir du point de terminaison TDS :

```
1> SELECT extname, extversion FROM pg_extension where extname like 'pg_ad_mapping';
2> GO
extname       extversion
------------- ----------
pg_ad_mapping 0.1

(1 rows affected)
```

Gestion des connexions des groupes

Créez des connexions de groupe en suivant les étapes mentionnées sous Gestion des connexions. Pour faciliter la gestion, nous recommandons que l'identifiant soit le même que le nom du groupe de sécurité Active Directory (AD), bien que cela ne soit pas obligatoire. Exemples :

```
CREATE LOGIN [corp\accounts-group] FROM WINDOWS [WITH DEFAULT_DATABASE=database]
```

Mappage des connexions de groupes T-SQL avec le groupe de sécurité AD

Vous devez explicitement configurer un identifiant de groupe Windows T-SQL pour chaque groupe de sécurité AD qui nécessite un accès au serveur de base de données. Un utilisateur AD, qui fait partie d'au moins un groupe de sécurité AD provisionné, aura accès au serveur de base de données.

> **Note**
>
> Cet identifiant T-SQL ne pourra plus s'authentifier à l'aide d'une authentification basée sur un mot de passe.

Par exemple, accounts-group est un groupe de sécurité dans AD et si vous souhaitez configurer ce groupe de sécurité dans Babelfish, vous devez utiliser le format [corp\accounts-group].

- Groupe de sécurité AD : accounts-group
- Identifiant TSQL : [corp\accounts-group]
- Rôle PG équivalent pour un identifiant TSQL donné : accounts-group@CORP.EXAMPLE.COM

L'administrateur peut maintenant procéder à la création du mappage entre le groupe de sécurité AD et l'identifiant T-SQL à partir du point de terminaison PostgreSQL via la commande psql suivante. Pour plus d'informations sur l'utilisation des fonctions, consultez Utilisation des fonctions de l'extension `pg_ad_mapping`.

> **Note**
>
> L'identifiant T-SQL doit être spécifié au format login_name@FQDN lors de l'ajout du mappage. Les poids sont ignorés lors de la connexion via le point de terminaison TDS. Pour plus d'informations l'utilisation de la pondération, consultez Connexion à Babelfish via le point de terminaison PostgreSQL sur le port PostgreSQL.

```
postgres=>select pgadmap_set_mapping('accounts-group', 'accounts-
group@CORP.EXAMPLE.COM', <SID>, <Weight>);
```

Pour plus d'informations sur la récupération du SID du groupe de sécurité AD, consultez Récupération du SID du groupe Active Directory dans PowerShell.

Le tableau suivant présente un exemple de mappage entre les groupes de sécurité AD et les identifiants T-SQL :

Groupes de sécurité AD	Identifiants TSQL	Rôle PG équivalent pour un identifiant TSQL donné	Weight
accounts-group	[corp\accounts-group]	accounts-group@CORP.EXAMPLE.COM	7

Groupes de sécurité AD	Identifiants TSQL	Rôle PG équivalent pour un identifiant TSQL donné	Weight
sales-group	[corp\sales-group]	sales-group@CORP.EXAMPLE.COM	10
dev-group	[corp\dev-group]	dev-group@CORP.EXAMPLE.COM	7

```
postgres=> select admap.ad_sid, admap.ad_grp, lgn.orig_loginname, lgn.rolname,
 admap.weight from pgadmap_read_mapping() as admap, sys.babelfish_authid_login_ext as
 lgn where admap.pg_role = lgn.rolname;
    ad_sid    |     ad_grp     |    orig_loginname   |            rolname
 | weight
--------------+----------------+---------------------+-------------------------------
+--------
 S-1-5-67-890 | accounts-group | corp\accounts-group  | accounts-group@CORP.EXAMPLE.COM
 |     7
 S-1-2-34-560 | sales-group    | corp\sales-group     | sales-group@CORP.EXAMPLE.COM
 |     10
 S-1-8-43-612 | dev-group      | corp\dev-group       | dev-group@CORP.EXAMPLE.COM
 |     7
 (7 rows)
```

Connexion à Babelfish via un point de terminaison TDS

Dans l’exemple suivant, user1 est membre d’accounts-group et de sales-group, et user2 est membre d’accounts-group et de dev-group.

Nom d’utilisateur	Appartenance aux groupes de sécurité AD
user1	accounts-group, sales-group
user2	accounts-group, dev-group

Connectez-vous au serveur de base de données Babelfish à l'aide de l'utilitaire sqlcmd. Vous pouvez vérifier si un utilisateur (user1 dans cet exemple) a été authentifié à l'aide de Kerberos en suivant l'exemple ci-dessous :

```
1> select principal, gss_authenticated from pg_stat_gssapi where pid =
 pg_backend_pid();
2>  GO
principal               gss_authenticated
----------------------  -----------------
user1@CORP.EXAMPLE.COM  1

((1 rows affected))
1> select suser_name();
2>  GO
suser_name
----------
corp\user1

(1 rows affected)
```

Dans cet exemple, user1 hérite des privilèges d'accounts-group et de sales-group. Vous pouvez vérifier l'appartenance au groupe à l'aide de la vue système `sys.login_token`.

```
1> SELECT name, type FROM sys.login_token;
2>  GO
name                type
------------------- ----
corp\accounts-group WINDOWS GROUP
corp\sales-group    WINDOWS GROUP

(2 rows affected)
```

Audit et journalisation

Pour déterminer l'identité du principal de sécurité AD, utilisez les commandes suivantes :

```
1> select suser_name();
2> GO
suser_name
----------
```

```
corp\user1

(1 rows affected)
```

Actuellement, l'identité de l'utilisateur AD n'est pas visible dans les journaux. Vous pouvez activer le paramètre `log_connections` pour journaliser l'établissement des sessions de base de données. Pour plus d'informations, consultez log_connections. La sortie inclut l'identité de l'utilisateur AD en tant que principal, comme illustré dans l'exemple suivant. Le PID du système dorsal associé à cette sortie pourra ensuite aider à attribuer les actions à l'utilisateur AD réel.

```
bbf_group_ad_login@babelfish_db:[615]:LOG: connection authorized:
 user=bbf_group_ad_login database=babelfish_db application_name=sqlcmd GSS
 (authenticated=yes, encrypted=yes, principal=user1@CORP.EXAMPLE.COM)
```

Utilisation des privilèges liés à l'appartenance à un groupe de sécurité AD

Hériter des privilèges au niveau du serveur

Les utilisateurs AD membres d'un groupe de sécurité AD donné héritent des privilèges de niveau serveur accordés à l'identifiant du groupe Windows mappé. Par exemple, considérez le groupe de sécurité AD `accounts-group`, qui est autorisé à appartenir au rôle serveur `sysadmin` sur Babelfish. Vous pouvez hériter des privilèges au niveau serveur à l'aide de la commande suivante :

```
1> ALTER SERVER ROLE sysadmin ADD MEMBER [corp\accounts-group];
```

Par conséquent, tout utilisateur Active Directory membre du groupe de sécurité AD `accounts-group` hérite des privilèges de niveau serveur associés au rôle `sysadmin`. Cela signifie qu'un utilisateur comme `corp\user1`, en tant que membre de `accounts-group`, a désormais la possibilité d'effectuer des opérations au niveau du serveur dans Babelfish.

> **Note**
>
> Pour exécuter des instructions DDL au niveau du serveur, il doit exister un identifiant Windows pour l'utilisateur AD individuel. Pour plus d'informations, consultez Limitations.

Hériter des privilèges au niveau de la base de données

Pour accorder des privilèges au niveau de la base de données, un utilisateur de base de données doit être créé et mappé avec un identifiant de groupe Windows. Les utilisateurs AD membres d'un groupe de sécurité AD donné héritent des privilèges accordés à cet utilisateur de base de données au niveau de la base de données. Dans l'exemple suivant, vous pouvez voir comment les privilèges au niveau de la base de données sont attribués au groupe Windows [corp\accounts-group].

```
1> CREATE DATABASE db1;
2> GO
1> USE db1;
2> GO
Changed database context to 'db1'.
1> CREATE TABLE dbo.t1(a int);
2> GO
```

Créez un utilisateur de base de données [corp sales-group] pour la connexion du groupe Windows [corp\accounts-group]. Pour effectuer cette étape, connectez-vous via le point de terminaison TDS en utilisant un identifiant qui est membre de sysadmin.

```
1> CREATE USER [corp\accounts-group] FOR LOGIN [corp\accounts-group];
2> GO
```

Maintenant, connectez-vous en tant qu'utilisateur AD user1 pour vérifier l'accès à la table t1. Comme nous n'avons pas encore accordé les privilèges au niveau de la base de données, cela entraîne une erreur de refus d'autorisation.

```
1> SELECT * FROM dbo.t1;
2> GO
Msg 33557097, Level 16, State 1, Server db-inst, Line 1
permission denied for table t1
```

Accordez SELECT sur la table t1 à l’utilisateur de base de données [corp\accounts-group]. Pour effectuer cette étape, connectez-vous via le point de terminaison TDS en utilisant un identifiant qui est membre de sysadmin.

```
1> GRANT SELECT ON dbo.t1 TO [corp\accounts-group];
2> GO
```

Connectez-vous en tant qu’utilisateur AD user1 pour valider l’accès.

```
1> SELECT * FROM dbo.t1;
2> GO
a
-----------

(0 rows affected)
```

Traitement du comportement des instructions DDL sur la base d’un schéma par défaut ou explicite

Lorsque vous utilisez une session authentifiée AD, le schéma par défaut de la session en cours est déterminé par les conditions suivantes :

- S’il existe un utilisateur de base de données individuel, le schéma par défaut de cet utilisateur est considéré comme le schéma par défaut de la session en cours.
- S’il existe le schéma par défaut pour un utilisateur de base de données de groupe, le schéma par défaut de cet utilisateur est considéré comme le schéma par défaut de la session en cours avec l’identifiant de principal le plus faible.

Comprendre le comportement de l’instruction CREATE DDL

Si aucun schéma explicite n’est spécifié dans l’instruction CREATE DDL, la création de l’objet aura lieu dans le schéma par défaut de la session en cours. S’il est impossible de déterminer si le schéma est le schéma par défaut ou explicite, l’instruction DDL génère l’erreur suivante :

```
"Babelfish Unsupported Command : Schema required for CREATE DDLs when connecting with
 Active Directory Group authentication. Assign default schema to group user or specify
 schema in command."
```

Example : il n'existe pas de schéma par défaut pour l'utilisateur du groupe Windows

L'utilisateur du groupe Windows [corp\accounts-group] a un schéma par défaut NULL, et l'utilisateur AD user1 essaie d'exécuter le DDL sans spécifier le schéma explicitement. Étant donné qu'il n'existe pas d'identifiant ni d'utilisateur Windows individuel pour user1, il obtiendra uniquement les privilèges de l'utilisateur du groupe Windows [corp\accounts-group] au niveau de la base de données.

```
1> create TABLE t2(a int);
2> GO

Msg 33557097, Level 16, State 1, Server db-inst, Line 1
Babelfish Unsupported Command : Schema required for CREATE DDLs when connecting with
 Active Directory Group authentication. Assign default schema to group user or specify
 schema in command.
```

> **Note**
>
> Il n'existe pas d'identifiant ni d'utilisateur Windows individuel pour l'utilisateur AD user1

Example : il existe un schéma par défaut pour les utilisateurs du groupe Windows

Créez un utilisateur de groupe Windows pour l'identifiant [corp\accounts-group] avec le schéma par défaut à l'aide de sysadmin.

```
1> CREATE USER [corp\accounts-group] FOR LOGIN [corp\accounts-group] WITH
 DEFAULT_SCHEMA = sch_acc;
2> GO
1> CREATE SCHEMA sch_acc AUTHORIZATION [gad\accounts-group];
2> GO
1> SELECT name, principal_id, default_schema_name FROM sys.database_principals WHERE
 name = 'corp\accounts-group';
```

```
2> GO

name                principal_id default_schema_name
------------------- ------------ ------------------
corp\accounts-group 24162        sch_acc

(1 rows affected)
```

Essayez de créer un objet sans spécifier explicitement le schéma à l'aide de l'utilisateur AD user1. La table t2 sera créée dans le schéma par défaut de l'utilisateur du groupe Windows [corp\accounts-group]. Le propriétaire de cet objet sera le même que le propriétaire du schéma sch_acc.

```
1> CREATE TABLE t_group(a int);
2> GO
1> SELECT name, schema_name(schema_id) FROM sys.objects WHERE name like 't_group';
2> GO

name    schema_name
------- -----------
t_group sch_acc

(1 rows affected)
```

> **Note**
>
> Il n'existe pas d'identifiant ni d'utilisateur Windows individuel pour l'utilisateur AD user1

Example : il existe également un utilisateur de base de données individuel pour un utilisateur AD

S'il existe également un utilisateur de base de données individuel pour un utilisateur AD, les objets seront toujours créés dans le schéma associé à cet utilisateur. S'il n'existe pas de schéma pour l'utilisateur de base de données, le schéma dbo sera utilisé. Créez un identifiant et un utilisateur de base de données Windows individuels pour l'utilisateur AD user1. Connectez-vous via le point de terminaison TDS à l'aide d'un identifiant sysadmin.

```
1> CREATE LOGIN [corp\user1] FROM WINDOWS;
2> GO
1> CREATE USER [corp\user1] FOR LOGIN [corp\user1] WITH DEFAULT_SCHEMA = sch1;
2> GO
1> CREATE SCHEMA sch1 AUTHORIZATION [corp\user1];
2> GO
1> SELECT name, default_schema_name FROM sys.database_principals WHERE name = 'corp
\user1';
2> GO

name      default_schema_name
--------- -------------------
corp\user1 sch1

(1 rows affected)
```

Connectez-vous à l'aide de l'utilisateur AD user1 et essayez de créer un objet sans spécifier explicitement le schéma. La table t2 sera créée dans le schéma sch1. Notez également que le propriétaire de cet objet sera le même que le propriétaire du schéma sch1.

```
1> CREATE TABLE t2(a int);
2> GO
1> SELECT name, schema_name(schema_id) FROM sys.objects WHERE name like 't2';
2> GO

name schema_name
---- -----------
t2   sch1

(1 rows affected)
```

Limitations

- L'utilitaire de vidage et de restauration ne prend pas en charge le vidage des mappages d'extension pg_ad_mapping. Vous devrez recréer ces mappages après la restauration.

- Le déploiement bleu/vert n’est pas pris en charge pour les instances Babelfish pour Aurora PostgreSQL avec `pg_ad_mapping`.
- La création de schéma implicite n’est pas prise en charge. Les instructions DDL qui nécessitent la création d’un schéma implicite ne sont pas prises en charge.
- Les instructions DDL ALTER AUTHORIZATION ON DATABASE , CREATE DATABASE, CREATE LOGIN, ALTER LOGIN, ALTER SERVER ROLE, ALTER DATABASE au niveau du serveur ne sont pas prises en charge dans une session authentifiée en tant que groupe AD lorsqu’il n’existe pas d’identifiant Windows individuel, mais seulement l’identifiant Windows du groupe. Pour contourner cette limitation, il est recommandé d’effectuer ces opérations dans le cadre d’une session authentifiée par mot de passe ou de créer un identifiant Windows individuel.
- La création d’utilisateurs implicite n’est pas prise en charge. Comportement T-SQL idéal [pas encore pris en charge dans Babelfish] : dans certains cas, comme les instructions DDL et les instructions de contrôle d’accès telles que GRANT/REVOKE où le nom de l’utilisateur AD est spécifié dans la commande mais n’existe pas dans la base de données, l’utilisateur de base de données nommé en tant qu’utilisateur AD est implicitement créé.
- Pour les instructions DDL dans les procédures ou fonctions PL/pgSQL créées à partir du point de terminaison PSQL et exécutées à partir du point de terminaison TDS dans une session authentifiée en tant que groupe AD :
  - Les instructions ALTER/DROP sont prises en charge.
  - CREATE TABLE, CREATE VIEW, CREATE INDEX, CREATE FUNCTION/PROC, CREATE TYPE, CREATE SEQUENCE, CREATE TRIGGER, SELECT INTO, CREATE FULLTEXT INDEX , CREATE UNIQUE INDEX génèrent une erreur si le schéma n’est pas fourni explicitement et si le schéma par défaut est nul pour la session en cours.
  - CREATE DATABASE, CREATE EXTENSION et toutes les autres instructions CREATE pour les objets spécifiques à PG (sauf dans T-SQL) CREATE subscription, CREATE tablespace, CREATE policy, CREATE conversion ne sont pas prises en charge.
- Les instructions DDL provenant du point de terminaison PostgreSQL ne sont pas prises en charge dans les sessions authentifiées en tant que groupe AD. Pour contourner le problème, vous pouvez toujours vous connecter à l’aide d’un identifiant principal ou de tout autre utilisateur utilisant un mécanisme d’authentification basé sur un mot de passe.
- Les objets système tels que SUSER_SID(), IS_SRVROLEMEMBER(), IS_MEMBER(), sys.dm_exec_sessions ont les limites suivantes.
  - SUSER_SID() ne renvoie pas le SID lorsque l’utilisateur AD ou le groupe de sécurité AD est fourni.

- IS_SRVROLEMEMBER() ne tient pas compte de l'appartenance au rôle si l'utilisateur AD actuel hérite de l'appartenance au rôle serveur d'un identifiant de groupe Windows quelconque.
- IS_MEMBER() renvoie false pour toute requête liée au groupe Windows.
- sys.dm_exec_sessions n'affiche pas les valeurs attendues dans les colonnes login_name et nt_user_name.

Connexion à Babelfish via le point de terminaison PostgreSQL sur le port PostgreSQL

Vous pouvez utiliser les identifiants de groupe créées à partir du port TDS pour vous connecter via le port PostgreSQL. Pour vous connecter via le port PostgreSQL, vous devez spécifier le nom de l'utilisateur AD au format `<ad_username@FQDN>` à partir des applications clientes PostgreSQL. Vous ne pouvez pas utiliser le format `<DNS domain name\ad_username>`.

PostgreSQL utilise par défaut des comparaisons sensibles à la casse pour les noms d'utilisateur. Pour qu'Aurora PostgreSQL interprète les noms d'utilisateur Kerberos comme non sensibles à la casse, vous devez définir le paramètre krb_caseins_users comme true dans le groupe de paramètres du cluster Babelfish personnalisé. Ce paramètre est défini sur false par défaut. Pour plus d'informations, consultez Configuration de votre cluster de bases de données Aurora PostgreSQL pour les noms d'utilisateur insensibles à la casse.

Différences de comportement entre les points de terminaison T-SQL et PostgreSQL lorsqu'un utilisateur AD fait partie de plusieurs groupes

Supposons que l'utilisateur AD user1 fasse partie de deux groupes de sécurité AD [corp\accounts-group] et [corp\sales-group], et que l'administrateur de la base de données ait défini le mappage des utilisateurs de la manière suivante.

```
postgres=> select * from pgadmap_read_mapping();

ad_sid       | pg_role                          | weight | ad_grp
-------------+----------------------------------+--------+--------------
S-1-5-67-980 | accounts-group@CORP.EXAMPLE.COM | 7      | accounts-group
S-1-2-34-560 | sales-group@CORP.EXAMPLE.COM    | 10     | sales-group
(2 rows)
```

Si l'utilisateur se connecte depuis le point de terminaison T-SQL, lors de l'autorisation, il héritera des privilèges de tous les identifiants T-SQL associés. Dans cet exemple, user1 hérite de l'union des privilèges de l'identifiant du groupe T-SQL, et les poids sont ignorés. Ceci est conforme au comportement standard de T-SQL.

Cependant, si le même utilisateur se connecte depuis le point de terminaison PostgreSQL, il ne peut hériter des privilèges que d'un seul identifiant T-SQL associé ayant le poids le plus élevé. Si le même poids a été attribué aux deux identifiants de groupe T-SQL, l'utilisateur AD héritera des privilèges de l'identifiant T-SQL correspondant au mappage ajouté le plus récemment. Pour PostgreSQL, il est recommandé de spécifier des poids qui reflètent les autorisations/privilèges relatifs des rôles de base de données individuels afin d'éviter toute ambiguïté. Dans l'exemple ci-dessous, user1 s'est connecté via le point de terminaison PSQL et n'a hérité que des privilèges de sales-group.

```
babelfish_db=> select session_user, current_user;

   session_user               |   current_user
------------------------------+--------------------------
 sales-group@CORP.EXAMPLE.COM | sales-group@CORP.EXAMPLE.COM
(1 row)


babelfish_db=> select principal, gss_authenticated from pg_stat_gssapi where pid =
 pg_backend_pid();

     principal          | gss_authenticated
------------------------+-------------------
 user1@CORP.EXAMPLE.COM | t
(1 row)
```

Connexion à un cluster de bases de données Babelfish

Pour vous connecter à Babelfish, connectez-vous au point de terminaison du cluster Aurora PostgreSQL exécutant Babelfish. Votre client peut utiliser l'un des pilotes clients suivants, compatibles avec TDS version 7.1 à 7.4 :

- Open Database Connectivity (ODBC)
- OLE DB Driver/MSOLEDBSQL
- Java Database Connectivity (JDBC) version 8.2.2 (mssql-jdbc-8.2.2) et ultérieure
- Fournisseur de données Microsoft SqlClient pour SQL Server
- Fournisseur de données .NET pour SQL Server
- SQL Server Native Client 11.0 (obsolète)
- OLE DB Provider/SQLOLEDB (obsolète)

Babelfish vous permet d'exécuter ce qui suit :

- Outils, applications et syntaxe SQL Server sur le port TDS, qui correspond par défaut au port 1433.
- Outils, applications et syntaxe PostgreSQL sur le port PostgreSQL, qui correspond par défaut au port 5432.

Pour en savoir plus sur la connexion à Aurora PostgreSQL en général, consultez Connexion à un cluster de bases de données Amazon Aurora PostgreSQL.

> **Note**
>
> Les outils de développement tiers utilisant le fournisseur SQL Server OLEDB pour accéder aux métadonnées ne sont pas pris en charge. Nous vous recommandons d'utiliser les connexions client JDBC, ODBC ou SQL Native de SQL Server pour ces outils.

À partir de la version 5.1.0 de Babelfish, le chiffrement des connexions de bout en bout est appliqué par défaut. Pour garantir une connectivité continue :

- Configurez le chiffrement SSL/TLS pour vos connexions. Pour plus d'informations, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.

- Importez les certificats requis sur vos ordinateurs clients. Pour plus d'informations, consultez Utilisation de SSL avec une instance de base de données Microsoft SQL Server.

Si vous souhaitez continuer à utiliser les paramètres de chiffrement d'une version précédente de Babelfish (antérieure à la version 5.1.0), vous pouvez définir le paramètre `rds.force_ssl` sur `0` dans le groupe de paramètres de votre cluster de bases de données.

Rubriques

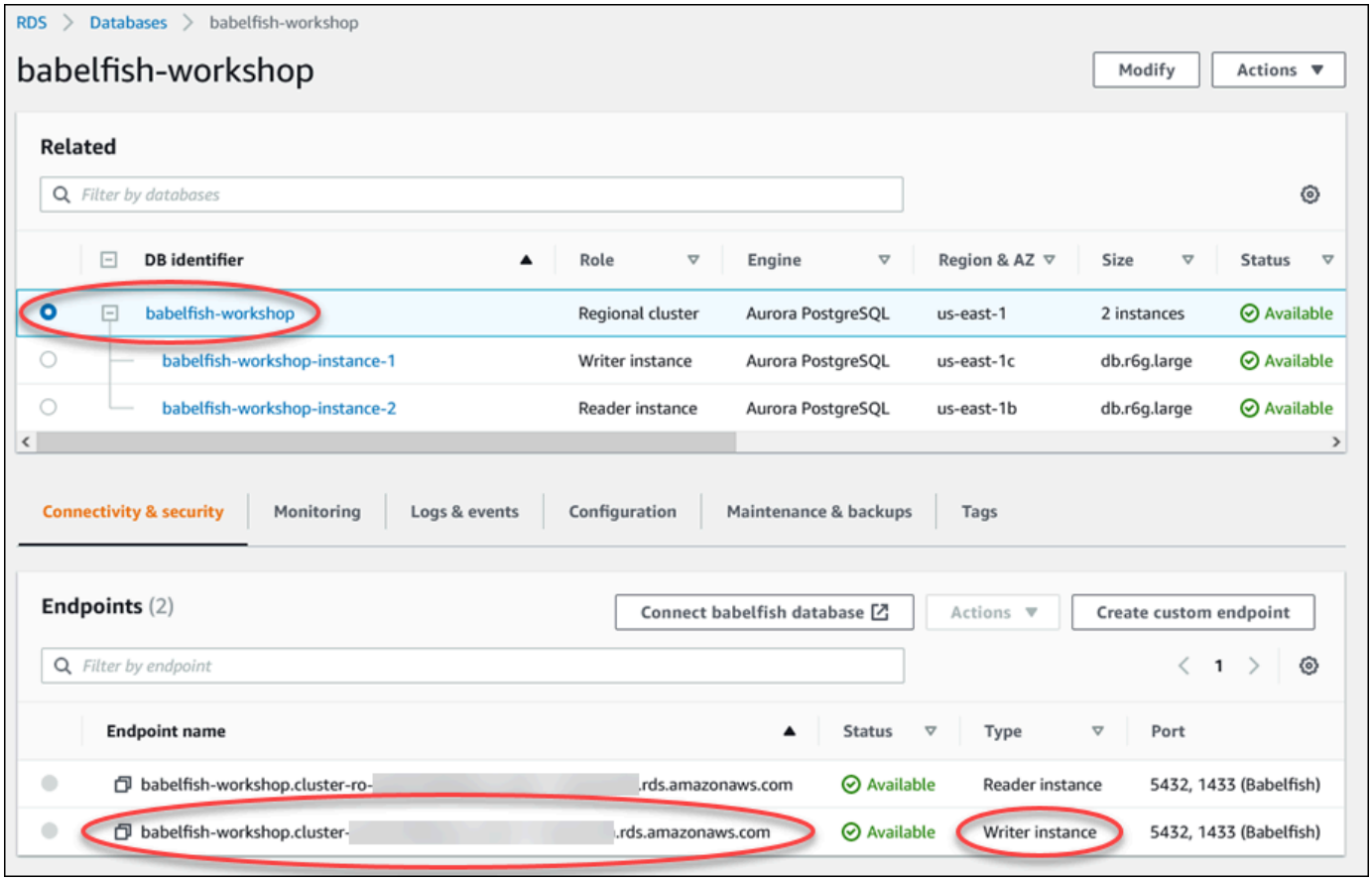
- Recherche du point de terminaison et du numéro de port de l'enregistreur
- Création de connexions client C# ou JDBC à Babelfish
- Utilisation d'un client SQL Server pour se connecter au cluster de bases de données
- Utilisation d'un client PostgreSQL pour se connecter au cluster de bases de données

Recherche du point de terminaison et du numéro de port de l'enregistreur

Pour vous connecter à votre cluster de bases de données Babelfish, vous utilisez le point de terminaison associé à l'instance (principale) d'enregistreur du cluster de bases de données. L'instance doit être à l'état Available (Disponible). La disponibilité des instances peut prendre jusqu'à 20 minutes après la création du cluster de bases de données Babelfish pour Aurora PostgreSQL.

Pour rechercher le point de terminaison de votre base de données

1. Ouvrez la console de Babelfish.
2. Choisissez Bases de données dans le panneau de navigation.
3. Choisissez votre cluster de bases de données Babelfish pour Aurora PostgreSQL parmi ceux répertoriés pour en savoir plus.
4. Dans l'onglet Connectivity & security (Connectivité et sécurité), notez les valeurs des points de terminaison (Endpoints) de cluster disponibles. Utilisez le point de terminaison de cluster de l'instance de l'enregistreur dans vos chaînes de connexion pour toute application qui exécute des opérations d'écriture ou de lecture à partir de la base de données.



Pour en savoir plus sur les détails d'un cluster de bases de données Aurora, consultez Création d'un cluster de bases de données Amazon Aurora.

Création de connexions client C# ou JDBC à Babelfish

Vous trouverez ci-dessous des exemples d'utilisation des classes C # et JDBC pour vous connecter à un cluster de bases de données Babelfish for Aurora PostgreSQL.

Example : utilisation de code C# pour se connecter à un cluster de bases de données

```
string dataSource = 'babelfishServer_11_24';

//Create connection
connectionString = @"Data Source=" + dataSource
    +";Initial Catalog=your-DB-name"
    +";User ID=user-id;Password=password";

// [optional] To validate server certificate during TLS/SSL connection
connectionString = connectionString + ";ServerCertificate=/path/to/certificate.pem";


SqlConnection cnn = new SqlConnection(connectionString);
cnn.Open();
```

Example : utilisation des classes et interfaces génériques de l'API JDBC pour se connecter à un cluster de bases de données

```
String dbServer =
   "database-babelfish.cluster-123abc456def.us-east-1-rds.amazonaws.com";
String connectionUrl = "jdbc:sqlserver://" + dbServer + ":1433;" +
    "databaseName=your-DB-name;user=user-id;password=password";

// [optional] To validate server certificate during TLS/SSL connection
connectionUrl = connectionUrl + ";serverCertificate=/path/to/certificate.pem";

// Load the SQL Server JDBC driver and establish the connection.
System.out.print("Connecting Babelfish Server ... ");
Connection cnn = DriverManager.getConnection(connectionUrl);
```

Example : utilisation des classes et interfaces JDBC spécifiques à SQL Server pour se connecter à un cluster de bases de données

```
// Create datasource.
SQLServerDataSource ds = new SQLServerDataSource();
ds.setUser("user-id");
```

```
ds.setPassword("password");
String babelfishServer =
   "database-babelfish.cluster-123abc456def.us-east-1-rds.amazonaws.com";

ds.setServerName(babelfishServer);
ds.setPortNumber(1433);
ds.setDatabaseName("your-DB-name");

// [optional] To validate server certificate during TLS/SSL connection
ds.setServerCertificate("/path/to/certificate.pem");

Connection con = ds.getConnection();
```

Important

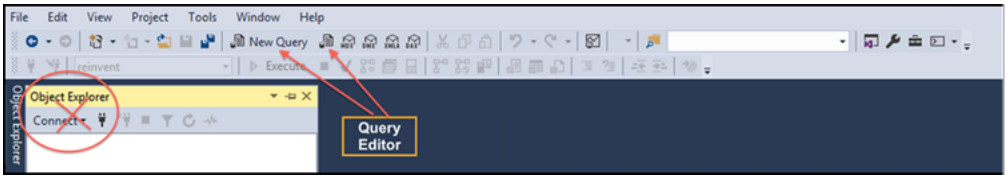
Assurez-vous que le certificat correspond à l'autorité de certification indiquée dans la configuration de votre cluster de bases de données dans la AWS Management Console.

Utilisation d'un client SQL Server pour se connecter au cluster de bases de données

Vous pouvez utiliser un client SQL Server pour vous connecter à Babelfish sur le port TDS. À partir de Babelfish 2.1.0 et versions ultérieures, vous pouvez utiliser l'explorateur d'objets SSMS ou l'éditeur de requêtes SSMS pour vous connecter à votre cluster Babelfish.

Limitations

- Dans Babelfish 2.1.0 et les versions antérieures, l'utilisation de `PARSE` pour vérifier la syntaxe SQL ne fonctionne pas comme prévu. Plutôt que de vérifier la syntaxe sans exécuter la requête, la commande `PARSE` exécute la requête mais n'affiche aucun résultat. L'utilisation de la combinaison de touches <Ctrl><F5> dans SMSS pour vérifier la syntaxe a le même comportement anormal : Babelfish exécute la requête de façon inattendue sans fournir de sortie.
- Babelfish ne prend pas en charge MARS (Multiple Active Result Sets). Assurez-vous que toutes les applications clientes que vous utilisez pour vous connecter à Babelfish ne sont pas configurées pour utiliser MARS.
- Pour Babelfish 1.3.0 et versions antérieures, seul l'éditeur de requêtes est pris en charge pour SSMS. Pour utiliser SSMS avec Babelfish, veillez à ouvrir la boîte de dialogue de connexion à l'éditeur de requêtes dans SSMS, et non l'explorateur d'objets. Si la boîte de dialogue de l'explorateur d'objets s'ouvre, annulez la boîte de dialogue et rouvrez l'éditeur de requêtes. L'image suivante illustre les options de menu à choisir lors de la connexion à Babelfish 1.3.0 ou versions antérieures.



Pour plus d'informations sur l'interopérabilité et les différences de comportement entre SQL Server et Babelfish, consultez Différences entre Babelfish pour Aurora PostgreSQL et SQL Server.

Utilisation de sqlcmd pour se connecter au cluster de bases de données

Vous pouvez vous connecter à un cluster de bases de données Aurora PostgreSQL qui prend en charge Babelfish et interagir avec en n'utilisant que la version 19.1 et les versions antérieures du client de ligne de commande SQL Server `sqlcmd`. La version 19.2 de SSMS n'est pas prise en charge pour se connecter à un cluster Babelfish. Utilisez la commande suivante pour vous connecter.

```
sqlcmd -S endpoint,port -U login-id -P password -d your-DB-name
```

Les options sont les suivantes :

- `-S` est le point de terminaison et (facultatif) le port TDS du cluster de bases de données.
- `-U` est le nom de connexion de l'utilisateur.
- `-P` est le mot de passe associé à l'utilisateur.
- `-d` est le nom de votre base de données Babelfish.

Après la connexion, vous pouvez utiliser la plupart des commandes que vous utilisez avec SQL Server. Pour obtenir quelques exemples, consultez Obtention d'informations dans le catalogue système Babelfish.

Utilisation de SSMS pour se connecter au cluster de bases de données

Vous pouvez vous connecter à un cluster de bases de données Aurora PostgreSQL exécutant Babelfish à l'aide de Microsoft SQL Server Management Studio (SSMS). SSMS comprend divers outils, y compris l'assistant d'importation et d'exportation SQL Server mentionné dans Migration d'une base de données SQL Server vers Babelfish pour Aurora PostgreSQL. Pour plus d'informations sur SSMS, consultez Télécharger SQL Server Management Studio (SSMS) dans la documentation de Microsoft. Pour configurer SSL/TLS, consultez Utilisation de SSL avec une instance de base de données Microsoft SQL Server.

Pour vous connecter à votre base de données Babelfish avec SSMS

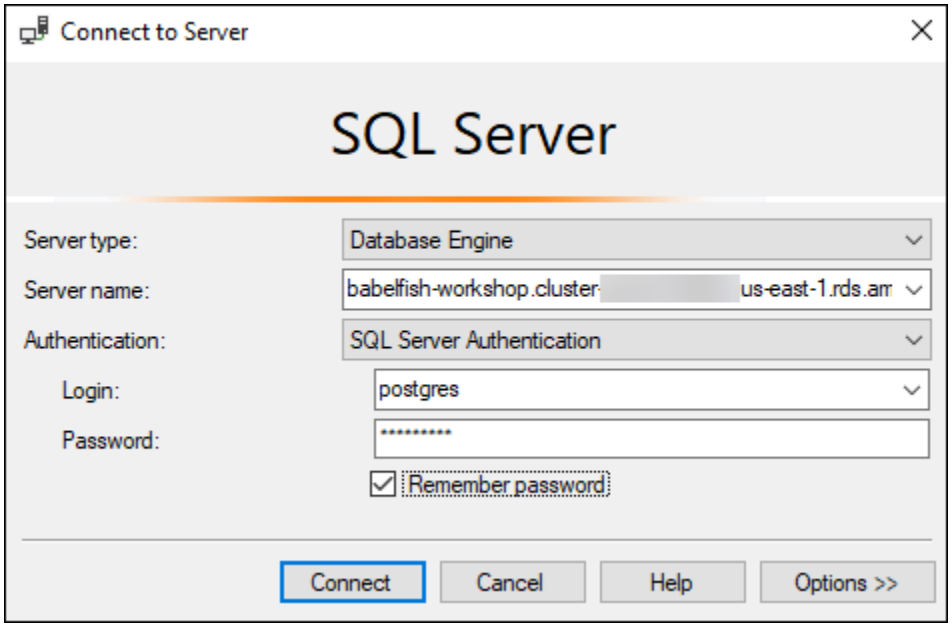
1. Démarrez SSMS.
2. Ouvrez la boîte de dialogue Connect to Server (Se connecter à un serveur). Pour poursuivre la connexion, effectuez l'une des actions suivantes :
   - Choisissez New Query (Nouvelle requête).
   - Si l'éditeur de requêtes est ouvert, choisissez Query (Requête), Connection (Connexion), Connect (Se connecter).
3. Fournissez les informations suivantes pour votre base de données :
   a. Pour Server type (Type de serveur), choisissez Database Engine (Moteur de base de données).
   b. Dans le champ Server name (Nom du serveur), saisissez le nom DNS. Par exemple, le nom du serveur doit être semblable au suivant.

```
cluster-name.cluster-555555555555.aws-region.rds.amazonaws.com,1433
```

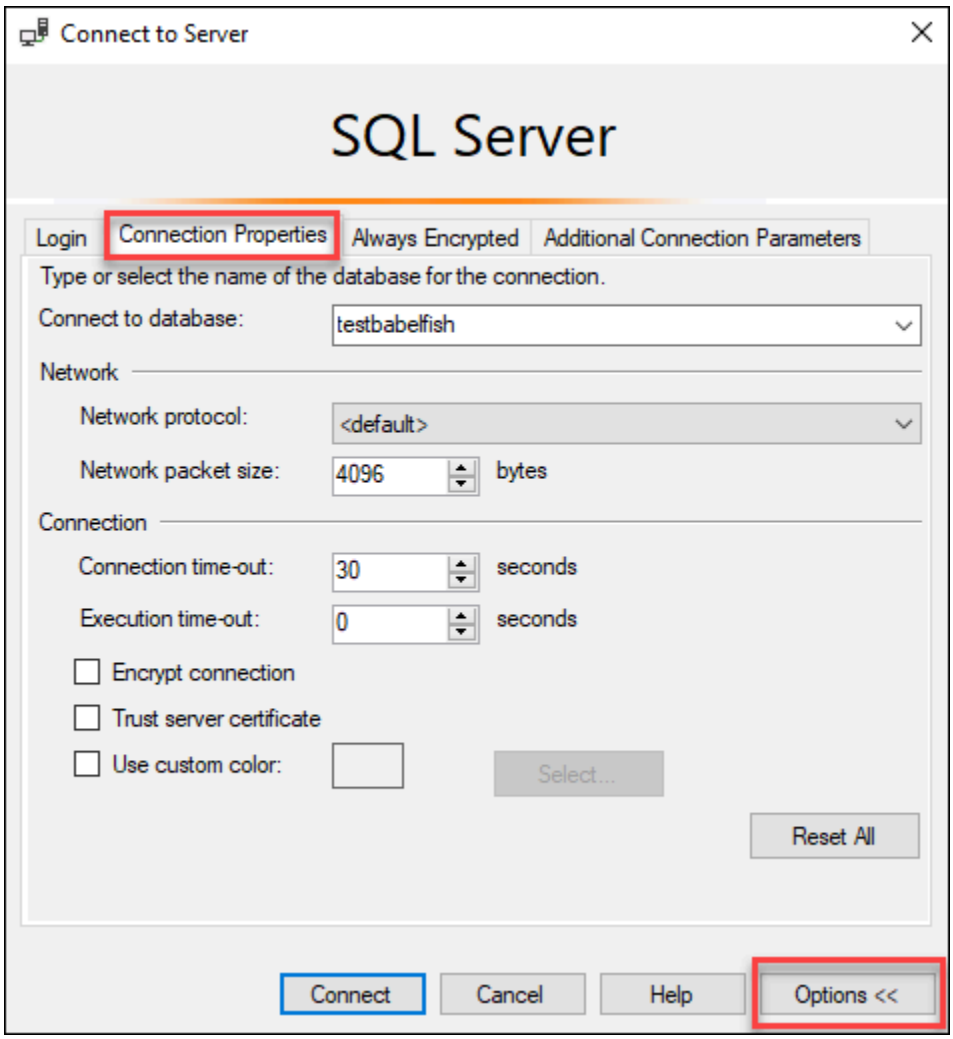
c. Pour Authentication, choisissez Authentication SQL Server.

d. Dans le champ Login (Identifiant), saisissez le nom d'utilisateur que vous avez choisi lors de la création de votre base de données.

e. Dans le champ Password (Mot de passe), saisissez le mot de passe que vous avez choisi lors de la création de votre base de données.



4. (Facultatif) Choisissez Options, puis l'onglet Connection Properties (Propriétés de connexion).



5. (Facultatif) Dans le champ Connect to database (Connexion à la base de données), spécifiez le nom de la base de données SQL Server migrée à laquelle vous souhaitez vous connecter, puis choisissez Connect (Se connecter).

   Si un message apparaît pour indiquer que SSMS ne peut pas appliquer de chaînes de connexion, choisissez OK.

   Si vous avez des difficultés à vous connecter à Babelfish, consultez Échec de connexion.

   Pour plus d'informations sur les problèmes de connexion au serveur SQL, consultez Troubleshooting connections to your SQL Server DB instance (Dépannage des connexions à votre instance de base de données SQL Server) dans le Guide de l'utilisateur Amazon RDS.

Utilisation d’un client PostgreSQL pour se connecter au cluster de bases de données

Vous pouvez utiliser un client PostgreSQL pour vous connecter à Babelfish sur le port PostgreSQL. À partir de la version 5.1.0, le serveur Babelfish applique le chiffrement des connexions de bout en bout par défaut. Mettez à jour votre application pour qu’elle utilise les certificats SSL/TLS. Pour plus d’informations sur la configuration des certificats SSL/TLS, consultez the section called “Sécurisation des données Aurora PostgreSQL avec SSL/TLS”.

Utilisation de psql pour se connecter au cluster de bases de données

Vous pouvez télécharger le client PostgreSQL depuis le site Web de PostgreSQL. Pour installer psql, suivez les instructions spécifiques à votre version du système d’exploitation.

Vous pouvez interroger un cluster de bases de données Aurora PostgreSQL prenant en charge Babelfish à l’aide du client de ligne de commande `psql`. Lors de la connexion, utilisez le port PostgreSQL (le port 5432 par défaut). En règle générale, vous n’avez pas besoin de spécifier le numéro de port, sauf si vous l’avez modifié par rapport à la valeur par défaut. Utilisez la commande suivante pour vous connecter à Babelfish à partir du client `psql` :

```
psql -h bfish-db.cluster-123456789012.aws-region.rds.amazonaws.com
-p 5432 -U postgres -d babelfish_db
```

Les paramètres sont les suivants :

- `-h` : nom d’hôte du cluster de bases de données (point de terminaison du cluster) auquel vous souhaitez accéder.
- `-p` : numéro de port PostgreSQL utilisé pour la connexion à votre instance de base de données.
- `-d` : base de données à laquelle vous souhaitez vous connecter. La valeur par défaut est `babelfish_db`.
- `-U` : compte d’utilisateur de la base de données auquel vous souhaitez accéder. (L’exemple montre le nom d’utilisateur principal par défaut.)

Lorsque vous exécutez une commande SQL sur le client psql, vous concluez la commande par un point-virgule. Par exemple, la commande SQL suivante interroge la vue système pg_tables pour renvoyer des informations sur chaque table de la base de données.

```
SELECT * FROM pg_tables;
```

Le client psql dispose également d'un ensemble de métacommandes intégrées. Une métacommande est un raccourci qui ajuste la mise en forme ou fournit un raccourci qui renvoie des métadonnées dans un format facile à utiliser. Par exemple, la métacommande suivante renvoie des informations semblables à la commande SQL précédente :

```
\d
```

Il n'est pas nécessaire de conclure les métacommandes par un point-virgule (;).

Pour quitter le client psql, saisissez \q.

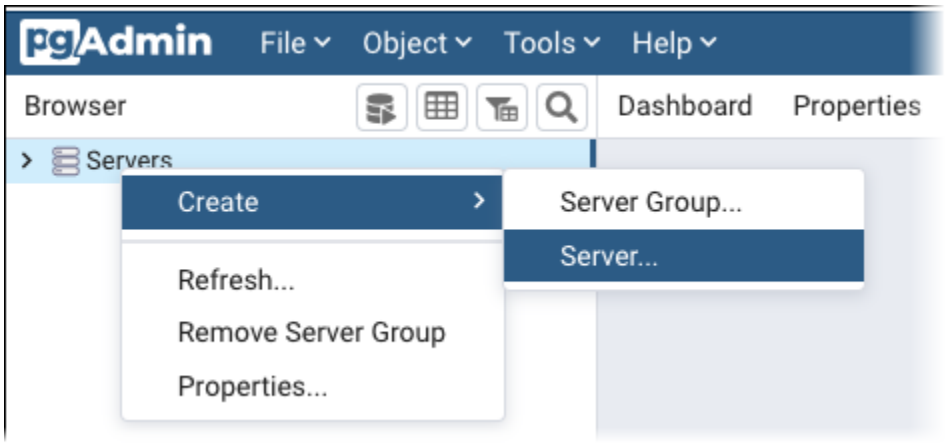
Pour en savoir plus sur l'utilisation du client psql pour interroger un cluster Aurora PostgreSQL, consultez la documentation PostgreSQL.

Utilisation de pgAdmin pour se connecter au cluster de bases de données

Vous pouvez utiliser le client pgAdmin pour accéder à vos données en dialecte PostgreSQL natif.

Pour vous connecter au cluster avec le client pgAdmin

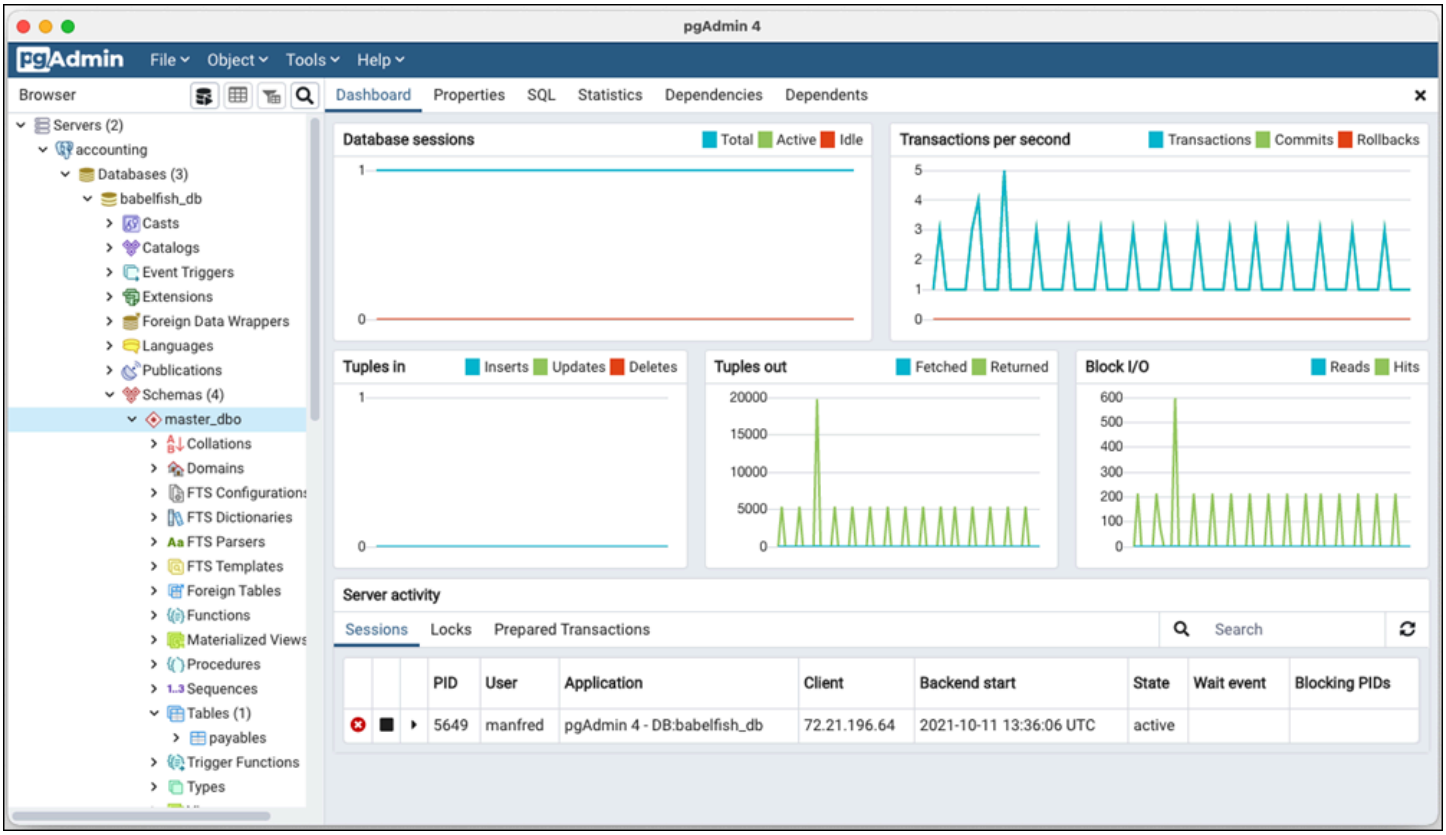
1. Téléchargez et installez le client pgAdmin à partir du site Internet de pgAdmin.
2. Ouvrez le client et authentifiez-vous auprès de pgAdmin.
3. Ouvrez le menu contextuel (clic droit) pour accéder à l'option Servers (Serveurs), et choisissez Create (Créer), Server (Serveur).



4. Saisissez les informations dans la boîte de dialogue Create - Server (Créer - Serveur).

   Sur la page Connection (Connexion), ajoutez l'adresse du cluster Aurora PostgreSQL dans le champ Host (Hôte) et le numéro de port PostgreSQL (par défaut, 5432) dans le champ Port. Fournissez les informations d'authentification requises et choisissez Save (Enregistrer).

Une fois connecté, vous pouvez utiliser la fonctionnalité PGADmin pour surveiller et gérer votre cluster Aurora PostgreSQL sur le port PostgreSQL.



Pour en savoir plus, consultez la page Web pgAdmin.

Utilisation de Babelfish

La présente section fournit des informations d'utilisation de Babelfish, y compris certaines des différences entre l'utilisation de Babelfish et de SQL Server, et entre les bases de données Babelfish et PostgreSQL.

Rubriques

Obtention d’informations dans le catalogue système Babelfish

Vous pouvez obtenir des informations sur les objets de base de données stockés dans votre cluster Babelfish en interrogeant la plupart des mêmes vues système que celles utilisées dans SQL Server. Chaque nouvelle version de Babelfish prend en charge davantage de vues système. Pour obtenir la liste des vues disponibles actuellement, consultez le tableau SQL Server system catalog views.

Ces vues système fournissent des informations issues du catalogue système (`sys.schemas`). Dans le cas de Babelfish, ces vues contiennent à la fois des schémas système SQL Server et PostgreSQL. Pour interroger Babelfish au sujet d’informations sur le catalogue système, vous pouvez utiliser le port TDS ou le port PostgreSQL, comme illustré dans les exemples suivants.

- Interrogez le port T-SQL à l’aide de **`sqlcmd`** ou d’un autre client SQL Server.

```
1> SELECT * FROM sys.schemas
2> GO
```

Cette requête renvoie les schémas système SQL Server et Aurora PostgreSQL, comme illustré dans l'exemple suivant.

```
name
------------------------------------------------------------
demographic_dbo
public
sys
master_dbo
tempdb_dbo
...
```

- Interrogez le port PostgreSQL à l'aide de **`psql`** ou **`pgAdmin`**. Cet exemple utilise la métacommande de schémas de liste `psql` (`\dn`) :

```
babelfish_db=> \dn
```

La requête renvoie le même jeu de résultats que celui renvoyé par `sqlcmd` sur le port T-SQL.

```
        List of schemas
             Name
------------------------------

 demographic_dbo

 public
 sys
 master_dbo
 tempdb_dbo
...
```

Catalogues système SQL Server disponibles dans Babelfish

Le tableau suivant fournit les vues SQL Server actuellement implémentées dans Babelfish. Pour plus d'informations sur les catalogues système dans SQL Server, consultez Vues de catalogue système (Transact-SQL) dans la documentation Microsoft.

Nom de la vue	Description ou limite Babelfish (le cas échéant)
`sys.all_columns`	Toutes les colonnes de toutes les tables et vues
`sys.all_objects`	Tous les objets de tous les schémas
`sys.all_sql_modules`	L'union de `sys.sql_modules` et `sys.system_sql_modules`
`sys.all_views`	Toutes les vues de tous les schémas
`sys.columns`	Toutes les colonnes des tables et vues définies par l'utilisateur
`sys.configurations`	Prise en charge de Babelfish limitée à une seule configuration en lecture seule.
`sys.data_spaces`	Contient une ligne pour chaque espace de données. Il peut s'agir d'un groupe de fichiers, d'un schéma de partition ou d'un groupe de fichiers de données FILESTREAM.
`sys.database_files`	Vue par base de données contenant une ligne pour chaque fichier d'une base de données telle qu'elle est stockée dans la base de données elle-même.
`sys.database_mirroring`	Pour plus d'informations, consultez sys.database_mirroring dans la documentation Microsoft Transact-SQL.
`sys.database_principals`	Pour plus d'informations, consultez sys.database_principals dans la documentation Microsoft Transact-SQL.
`sys.database_role_members`	Pour plus d'informations, consultez sys.database_role_members dans la documentation Microsoft Transact-SQL.

Nom de la vue	Description ou limite Babelfish (le cas échéant)
`sys.databases`	Toutes les bases de données de tous les schémas
`sys.dm_exec_connections`	Pour plus d'informations, consultez sys.dm_exec_connections dans la documentation Microsoft Transact-SQL.
`sys.dm_exec_sessions`	Pour plus d'informations, consultez sys.dm_exec_sessions dans la documentation Microsoft Transact-SQL.
`sys.dm_hadr_database_replica_states`	Pour plus d'informations, consultez sys.dm_hadr_database_replica_states dans la documentation Microsoft Transact-SQL.
`sys.dm_os_host_info`	Pour plus d'informations, consultez sys.dm_os_host_info dans la documentation Microsoft Transact-SQL.
`sys.endpoints`	Pour plus d'informations, consultez sys.endpoints dans la documentation Microsoft Transact-SQL.
`sys.indexes`	Pour plus d'informations, consultez sys.indexes dans la documentation Microsoft Transact-SQL.
`sys.languages`	Pour plus d'informations, consultez sys.languages dans la documentation Microsoft Transact-SQL.
`sys.schemas`	Tous les schémas
`sys.server_principals`	Tous les identifiants et rôles
`sys.sql_modules`	Pour plus d'informations, consultez sys.sql_modules dans la documentation Microsoft Transact-SQL.

Nom de la vue	Description ou limite Babelfish (le cas échéant)
`sys.sysconfigures`	Prise en charge de Babelfish limitée à une seule configuration en lecture seule.
`sys.syscurconfigs`	Prise en charge de Babelfish limitée à une seule configuration en lecture seule.
`sys.sysprocesses`	Pour plus d'informations, consultez sys.sysprocesses dans la documentation Microsoft Transact-SQL.
`sys.system_sql_modules`	Pour plus d'informations, consultez sys.system_sql_modules dans la documentation Microsoft Transact-SQL.
`sys.table_types`	Pour plus d'informations, consultez sys.table_types dans la documentation Microsoft Transact-SQL.
`sys.tables`	Toutes les tables d'un schéma
`sys.xml_schema_collections`	Pour plus d'informations, consultez sys.xml_schema_collections dans la documentation Microsoft Transact-SQL.

PostgreSQL implémente des catalogues système semblables aux vues du catalogue d'objets SQL Server. Pour obtenir la liste complète des catalogues système, consultez System Catalogs dans la documentation PostgreSQL.

Exportations DDL prises en charge par Babelfish

Dans les versions 2.4.0 et 3.1.0 de Babelfish, Babelfish prend en charge les exportations DDL à l'aide de divers outils. Par exemple, vous pouvez utiliser cette fonctionnalité de SQL Server Management Studio (SSMS) pour générer les scripts de définition de données pour différents objets dans une base de données Babelfish pour Aurora PostgreSQL. Vous pouvez ensuite utiliser les commandes DDL générées dans ce script pour créer les mêmes objets dans une autre base de données Babelfish pour Aurora PostgreSQL ou SQL Server.

Babelfish prend en charge les exportations DDL pour les objets suivants dans les versions spécifiées.

Liste d’objets	2.4.0	3.1.0
Tables d’utilisateurs	Oui	Oui
Clés primaires	Oui	Oui
Clés étrangères	Oui	Oui
Contraintes uniques	Oui	Oui
Index	Oui	Oui
Contraintes de validation	Oui	Oui
Vues	Oui	Oui
Procédures stockées	Oui	Oui
Fonctions définies par l’utilisateur	Oui	Oui
Fonctions à valeur tabulaire	Oui	Oui
Déclencheurs	Oui	Oui
Types de données définis par l’utilisateur	Non	Non
Types de tables définis par l’utilisateur	Non	Non
Users	Non	Non
Connexions	Non	Non
Séquences	Non	Non
Rôles	Non	Non

Limites liées aux DDL exportées

- Utiliser des trappes de secours avant de recréer les objets avec les DDL exportées : Babelfish ne prend pas en charge toutes les commandes du script DDL exporté. Utilisez des trappes de secours pour éviter les erreurs causées lors de la recréation des objets à partir des commandes DDL dans Babelfish. Pour plus d'informations sur les trappes de secours, consultez Gestion du traitement des erreurs Babelfish avec des trappes de secours.
- Objets contenant des contraintes CHECK avec des clauses COLLATE explicites : les scripts contenant ces objets générés à partir d'une base de données SQL Server ont des classements différents mais équivalents à ceux de la base de données Babelfish. Par exemple, quelques classements, tels que sql_latin1_general_cp1_cs_as, sql_latin1_general_cp1251_cs_as et latin1_general_cs_as, sont générés sous la forme latin1_general_cs_as, qui est le classement Windows le plus proche.

Gestion des autorisations et des contrôles d'accès dans Babelfish pour Aurora PostgreSQL

Dans Babelfish pour Aurora PostgreSQL, vous pouvez gérer les autorisations et le contrôle d'accès pour les bases de données, les schémas et les objets. Le tableau suivant décrit les commandes SQL spécifiques permettant d'accorder des autorisations dans Babelfish afin de réaliser différents scénarios de contrôle d'accès. Il couvre les cas d'utilisation pris en charge qui peuvent être mis en œuvre ainsi que les solutions de contournement pour les scénarios qui ne sont pas encore pris en charge. Cela vous permettra de configurer les autorisations appropriées pour répondre à vos exigences de sécurité et de conformité lors de l'utilisation de bases de données Babelfish.

Cas d'utilisation pris en charge

Le tableau suivant explique les cas d'utilisation pris en charge par Babelfish. Pour chaque cas d'utilisation, le tableau indique l'action nécessaire pour y parvenir ainsi que des exemples de commandes SQL.

Cas d'utilisation	Action	Commandes SQL	Commentaires	Compatibilité des versions Babelfish
Autoriser l'identif iant à effectuer	Ajouter l'identif iant au rôle	ALTER SERVER ROLE sysadmin	Aucun	Toutes les versions

Cas d’utilisation	Action	Commandes SQL	Commentaires	Compatibilité des versions Babelfish
des instructions SELECT/DML/DDL dans n’importe quelle base de données	serveur sysadmin	ADD MEMBER `login`		
Autoriser l’identifiant à effectuer des instructions SELECT/DML/DDL dans une base de données	Faire de l’identifiant le propriétaire de la base de données	ALTER AUTHORIZATION ON DATABASE::`database` TO `login`	Une base de données ne peut avoir qu’un seul propriétaire.	Versions 3.4 et ultérieures
Autoriser l’utilisateur de la base de données à effectuer des instructions SELECT/DML sur un schéma	Accorder l’autorisation à l’utilisateur de base de données sur le schéma	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON SCHEMA::`schema` TO `user`	Aucun	Versions 3.6 et ultérieures, 4.2 et ultérieures
Autoriser l’utilisateur de la base de données à effectuer des instructions SELECT/DML sur un schéma	Rendre l’utilisateur de base de données propriétaire du schéma au moment de sa création	CREATE SCHEMA `schema` AUTHORIZATION `user`	La modification de la propriété d’un schéma après sa création n’est actuellement pas prise en charge.	Versions 1.2 et ultérieures

Cas d’utilisation	Action	Commandes SQL	Commentaires	Compatibilité des versions Babelfish
Autoriser l’utilisateur de la base de données à effectuer des instructions SELECT/DML pour un objet	Accorder l’autorisation à l’utilisateur de base de données sur l’objet	GRANT SELECT/EXECUTE/INSERT/UPDATE/DELETE ON OBJECT::`object` TO `user`	Aucun	Toutes les versions
Autoriser l’utilisateur de la base de données à effectuer des instructions SELECT/DML/DDL dans une base de données, y compris supprimer la base de données	Ajouter un utilisateur au rôle de base de données fixe db_owner	ALTER ROLE db_owner ADD MEMBER `user`	Seuls les utilisateurs peuvent recevoir le rôle de base de données fixe db_owner. L’ajout de rôles au rôle db_owner n’est pas encore pris en charge.	Versions 4.5 et ultérieures, 5.1 et ultérieures
Autoriser l’utilisateur ou les membres d’un rôle de base de données personnalisé à exécuter uniquement des instructions SELECT dans une base de données	Ajouter un utilisateur ou un rôle au rôle de base de données fixe db_datareader	ALTER ROLE db_datareader ADD MEMBER `user` / `role`	Aucun	Versions 4.5 et ultérieures, 5.1 et ultérieures

Cas d’utilisation	Action	Commandes SQL	Commentaires	Compatibilité des versions Babelfish
Autoriser l’utilisateur ou les membres d’un rôle de base de données personnalisé à exécuter uniquement des instructions DML dans une base de données	Ajouter un utilisateur ou un rôle au rôle de base de données fixe db_datawriter	ALTER ROLE db_datawriter ADD MEMBER `user` / `role`	Aucun	Versions 4.5 et ultérieures, 5.1 et ultérieures
Autoriser l’utilisateur ou les membres d’un rôle de base de données personnalisé à exécuter uniquement des instructions DDL dans une base de données	Ajouter un utilisateur ou un rôle au rôle de base de données fixe db_accessadmin	ALTER ROLE db_accessadmin ADD MEMBER `user` / `role`	Aucun	Versions 4.5 et ultérieures, 5.1 et ultérieures

Cas d’utilisation	Action	Commandes SQL	Commentaires	Compatibilité des versions Babelfish
Autoriser l’utilisateur ou les membres d’un rôle de base de données personnalisé à exécuter uniquement des instructions CREATE/ALTER/DROP pour des rôles personnalisés, des instructions GRANT/REVOKE pour les autorisations liées aux objets d’une base de données et/ou des instructions CREATE SCHEMA dans une base de données	Ajouter un utilisateur ou un rôle au rôle de base de données fixe db_securityadmin	ALTER ROLE db_securityadmin ADD MEMBER `user` / `role`	Aucun	Versions 4.5 et ultérieures, 5.1 et ultérieures

Cas d’utilisation	Action	Commandes SQL	Commentaires	Compatibilité des versions Babelfish
Autoriser un utilisateur ou les membres d’un rôle de base de données personnalisé à exécuter uniquement des instructions CREATE/ALTER/DROP pour les utilisateurs, à accorder et à révoquer l’accès à la base de données et à mapper les comptes utilisateur aux identifiants et/ou à exécuter une instruction CREATE SCHEMA dans une base de données	Ajouter un utilisateur ou un rôle au rôle de base de données fixe db_access admin	ALTER ROLE db_accessadmin ADD MEMBER `user` / `role`	Aucun	Versions 4.5 et ultérieures, 5.1 et ultérieures

Cas d’utilisation	Action	Commandes SQL	Commentaires	Compatibilité des versions Babelfish
Autoriser l’identifiant à exécuter uniquement des instructions CREATE/DROP/ALTER pour n’importe quelle base de données	Ajouter un identifiant au rôle serveur fixe dbcreator	ALTER SERVER ROLE dbcreator ADD MEMBER `login`	Seules les bases de données auxquelles l’identifiant dbcreator a accès pourront être modifiées.	Versions 4.5 et ultérieures, 5.1 et ultérieures
Autoriser l’identifiant à exécuter uniquement des instructions CREATE/DROP/ALTER pour n’importe quel identifiant	Ajouter un identifiant au rôle serveur fixe securityadmin	ALTER SERVER ROLE securityadmin ADD MEMBER `login`	Aucun	Versions 4.5 et ultérieures, 5.1 et ultérieures

Cas d’utilisation non pris en charge avec solutions de contournement

Le tableau suivant explique les cas d’utilisation qui ne sont pas pris en charge dans Babelfish, mais qui peuvent être mis en place à l’aide d’une solution de contournement.

Cas d’utilisation	Action	Commandes SQL	Commentaires	Versions Babelfish compatibles pour les solutions de contournement
Autoriser l’utilisateur à exécuter des instructions SELECT/DM	Exécuter l’instruction GRANT pour accorder les autorisat	GRANT SELECT/EXECUTE/INSERT/UPDATE/	GRANT ... WITH GRANT OPTION n’est actuellem	Versions 3.6 et ultérieures, 4.2 et ultérieures

Cas d’utilisation	Action	Commandes SQL	Commentaires	Versions Babelfish compatibles pour les solutions de contournement
L sur un objet/ schéma avec la possibilité d’accorder ces autorisations à d’autres utilisate urs avec GRANT	ions à tous les autres utilisate urs directement	DELETE ON OBJECT/SC HEMA::`object/sc` TO `user`	ent pas pris en charge.	
Autoriser l’utilisa teur de la base de données à effectuer des instructi ons SELECT/ DML/DDL dans une base de données, y compris supprimer la base de données	Ajouter les membres du rôle au rôle de base de données fixe db_owner	ALTER ROLE db_owner ADD MEMBER `user`	L’ajout de rôles au rôle db_owner n’est actuellem ent pas pris en charge.	Versions 4.5 et ultérieures, 5.1 et ultérieures

Cas d’utilisation non pris en charge

Le tableau suivant explique les cas d’utilisation qui ne sont pas pris en charge dans Babelfish.

Cas d’utilisation	Commentaires
Interdire à un utilisateur ou à des membres d’un rôle de base de données personnalisé d’exécuter des instructions SELECT dans une base de données	Le rôle de base de données fixe db_denyda tareader n’est pas encore pris en charge

Cas d’utilisation	Commentaires
Interdire à un utilisateur ou à des membres d’un rôle de base de données personnalisé d’exécuter des instructions DML dans une base de données	Le rôle de base de données fixe db_denyda tawriter n’est actuellement pas pris en charge.
Autoriser l’identifiant à exécuter uniquemen t une instruction KILL pour n’importe quelle connexion à la base de données	Le rôle serveur fixe processadmin n’est actuellement pas pris en charge.
Autoriser l’identifiant à ajouter ou supprimer uniquement des serveurs liés	Le rôle serveur fixe setupadmin n’est actuellem ent pas pris en charge.

Gestion des différences de propriété des objets après la mise à niveau

Les versions 4.6 et ultérieures ainsi que les versions 5.2 et ultérieures de Babelfish incluent une modification de la gestion de la propriété des objets via le point de terminaison TDS. Lorsque vous créez des objets via le point de terminaison TDS, ils appartiennent désormais au propriétaire du schéma plutôt qu’à l’utilisateur actuel. Ce changement de propriétaire peut affecter le comportement des autorisations pour les nouveaux objets par rapport aux objets existants lorsque vous effectuez une mise à niveau à partir de versions antérieures à 4.6 ou 5.2.

Pour résoudre ces différences de propriété, Babelfish fournit la fonction `sys.generate_alter_ownership_statements()`. Cette fonction génère des instructions SQL qui alignent la propriété des objets sur la propriété du schéma.

Veillez à tenir compte des limites suivantes concernant la propriété des objets :

- Les utilisateurs disposant d’autorisations CREATE accordées via le point de terminaison PostgreSQL ne peuvent pas créer d’objets via le point de terminaison TDS dans ces schémas.
- Il n’est pas recommandé de modifier les autorisations liées aux objets T-SQL via le point de terminaison PostgreSQL. Cela peut entraîner un comportement T-SQL incorrect.
- Les autorisations d’accès peuvent différer entre les anciens et les nouveaux objets en raison de l’incompatibilité de leur propriété. Prenons l’exemple d’un schéma appartenant à `sch_own` et incluant des objets appartenant à `dbo`. Dans ce cas, les objets appartenant à `dbo` qui ont été créés avant la mise à niveau peuvent avoir des autorisations d’accès différentes par rapport aux objets

appartenant à `sch_own` créés après la mise à niveau. Cela peut affecter les opérations telles que SELECT et INSERT.

Si votre cluster de bases de données inclut des objets créés dans les versions de Babelfish antérieures à 4.6 ou 5.2, pensez à en aligner la propriété.

Pour remédier aux différences de propriété des objets

1. Connectez-vous à la base de données `babelfish_db` de votre cluster de bases de données à l'aide du point de terminaison PostgreSQL.
2. Exécutez la commande suivante :

   ```
   SELECT * from sys.generate_alter_ownership_statements();
   ```

   Cela génère une liste d'instructions SQL destinées à normaliser la propriété des objets.
3. Exécutez d'abord les instructions générées dans un environnement de test pour valider leur effet avant de les appliquer à votre environnement de production.

Nous vous recommandons d'exécuter ces instructions pour obtenir un modèle de propriété des objets cohérent dans l'ensemble de votre base de données.

Différences entre Babelfish pour Aurora PostgreSQL et SQL Server

Babelfish est une fonctionnalité évolutive d'Aurora PostgreSQL, avec de nouvelles fonctionnalités ajoutées à chaque version depuis l'offre initiale dans Aurora PostgreSQL 13.4. Il est conçu pour fournir une sémantique T-SQL sur PostgreSQL via le langage T-SQL en utilisant le port TDS. Chaque nouvelle version de Babelfish ajoute des fonctionnalités et des fonctions qui s'alignent mieux sur les fonctionnalités et le comportement de T-SQL, comme le montre le tableau Fonctionnalités prises en charge dans Babelfish, classées par version. Pour obtenir les meilleurs résultats lorsque vous travaillez avec Babelfish, nous vous recommandons de comprendre les différences qui existent actuellement entre le T-SQL pris en charge par SQL Server et Babelfish pour la dernière version. Pour en savoir plus, consultez Différences T-SQL dans Babelfish.

En plus des différences entre le T-SQL pris en charge par Babelfish et SQL Server, vous devrez peut-être aussi prendre en compte les problèmes d'interopérabilité entre Babelfish et PostgreSQL dans le contexte du cluster de bases de données Aurora PostgreSQL. Comme mentionné précédemment, Babelfish prend en charge la sémantique T-SQL sur PostgreSQL via le langage T-

SQL en utilisant le port TDS. En même temps, vous pouvez également accéder à la base de données Babelfish via le port standard PostgreSQL avec des instructions SQL PostgreSQL. Si vous envisagez d'utiliser les fonctionnalités de PostgreSQL et de Babelfish dans un déploiement de production, vous devez être conscient des problèmes d'interopérabilité potentiels entre les noms de schémas, les identifiants, les autorisations, la sémantique transactionnelle, les ensembles de résultats multiples, les classements par défaut, etc. Pour faire simple, lorsque des instructions PostgreSQL ou des accès PostgreSQL se produisent dans le contexte de Babelfish, des interférences entre PostgreSQL et Babelfish peuvent survenir et peuvent potentiellement affecter la syntaxe, la sémantique et la compatibilité lors du lancement de nouvelles versions de Babelfish. Pour obtenir des informations complètes et des conseils sur toutes ces considérations, consultez Guidance on Babelfish Interoperability (Conseils sur l'interopérabilité de Babelfish) dans la documentation de Babelfish pour PostgreSQL.

> **Note**
>
> Avant d'utiliser à la fois la fonctionnalité native de PostgreSQL et la fonctionnalité Babelfish dans le même contexte d'application, nous vous recommandons fortement de tenir compte des questions discutées dans la section Guidance on Babelfish Interoperability (Conseils sur l'interopérabilité de Babelfish) de la documentation de Babelfish pour PostgreSQL. Ces problèmes d'interopérabilité (Aurora PostgreSQL et Babelfish) ne sont pertinents que si vous prévoyez d'utiliser l'instance de la base de données PostgreSQL dans le même contexte applicatif que Babelfish.

Rubriques

- Vidage et restauration Babelfish
- Différences T-SQL dans Babelfish
- Niveaux d'isolement des transactions dans Babelfish

Vidage et restauration Babelfish

À partir des versions 4.0.0 et 3.4.0, les utilisateurs de Babelfish peuvent désormais utiliser les utilitaires de vidage et de restauration pour sauvegarder et restaurer leurs bases de données. Pour plus d'informations, consultez Vidage et restauration Babelfish. Cette fonctionnalité s'appuie sur les utilitaires de vidage et de restauration de PostgreSQL. Pour plus d'informations, consultez pg_dump et pg_restore. Afin d'utiliser efficacement cette fonctionnalité dans Babelfish, vous devez utiliser des outils basés sur PostgreSQL spécifiquement adaptés à Babelfish. La fonctionnalité de

sauvegarde et de restauration de Babelfish est très différente de celle de SQL Server. Pour plus d'informations sur ces différences, consultez Différences entre les fonctionnalités de vidage et de restauration : Babelfish et SQL Server. Babelfish pour Aurora PostgreSQL fournit des fonctionnalités supplémentaires pour la sauvegarde et la restauration des clusters de bases de données Amazon Aurora PostgreSQL. Pour plus d'informations, consultez Sauvegarde et restauration d'un cluster de bases de données Amazon Aurora.

Différences T-SQL dans Babelfish

Vous trouverez ci-dessous un tableau des fonctionnalités T-SQL prises en charge par la version actuelle de Babelfish, ainsi que quelques notes sur les différences de comportement par rapport à SQL Server.

Pour plus d'informations sur le support dans les différentes versions, consultez Fonctionnalités prises en charge dans Babelfish, classées par version. Pour plus d'informations sur les fonctionnalités qui ne sont actuellement pas prises en charge, consultez Fonctionnalités non prises en charge dans Babelfish.

Babelfish est disponible avec Aurora PostgreSQL-Compatible Edition. Pour obtenir plus d'informations sur les mises à jour de Babelfish, consultez Notes de mise à jour d'Aurora PostgreSQL.

Fonctionnalité ou syntaxe	Description du comportement ou de la différence
\ (caractère de continuation de ligne)	Le caractère de continuation de ligne (une barre oblique inverse avant une nouvelle ligne) pour les chaînes de caractères et les chaînes hexadécimales n'est pour le moment pas pris en charge. Pour les chaînes de caractères, le signe backslash-newline est interprété comme des caractères dans la chaîne. Pour les chaînes hexadécimales, le signe backslash-newline entraîne une erreur de syntaxe.
@@version	Le format de la valeur renvoyée par `@@version` est légèrement différent de celui de la valeur renvoyée par SQL Server. Votre code peut ne pas fonctionner correctement s'il repose sur le format de `@@version` .
Fonctions d'agrégation	Les fonctions d'agrégation sont partiellement prises en charge (les fonctions AVG, COUNT, COUNT_BIG, GROUPING, MAX,

Fonctionnalité ou syntaxe	Description du comportement ou de la différence
	MIN, STRING_AGG et SUM sont prises en charge). Pour obtenir une liste des fonctions d'agrégation non prises en charge, consultez Fonctions non prises en charge
ALTER TABLE	Prend en charge l'ajout ou la suppression d'une seule colonne ou d'une seule contrainte.
ALTER TABLE..ALTER COLUMN	Les valeurs NULL et NOT NULL ne peuvent pas être spécifiée s actuellement. Pour modifier la valeur NULL d'une colonne, utilisez l'instruction postgreSQL ALTER TABLE..{SET|DROP} NOT NULL.
AT TIME ZONE	Pendant le passage de l'heure d'été à l'heure standard, la période de chevauchement est affichée en utilisant le décalage horaire standard. Pour clarifier, prenez l'exemple suivant : `SELECT CONVERT(DATETIME2(0), '2022-10-30T02:00:00', 126) AT TIME ZONE 'Central European Standard Time';` `GO;` `Result: 2022-10-30 02:00:00 +01:00`
Noms de colonnes vides sans alias de colonne	Les utilitaires `sqlcmd` et `psql` traitent différemment les colonnes dont le nom est vide : • SQL Server `sqlcmd` renvoie un nom de colonne vide. • PostgreSQL `psql` renvoie un nom de colonne généré.
Fonction CHECKSUM	Babelfish et SQL Server utilisent des algorithmes de hachage différents pour la fonction CHECKSUM. Par conséquent, les valeurs de hachage générées par la fonction CHECKSUM dans Babelfish peuvent être différentes de celles générées par la fonction CHECKSUM dans SQL Server.

Fonctionnalité ou syntaxe	Description du comportement ou de la différence
Colonne par défaut	Lors de la création d'une colonne par défaut, le nom de la contrainte est ignoré. Pour supprimer une colonne par défaut, utilisez la syntaxe suivante : `ALTER TABLE...ALTER COLUMN..DROP DEFAULT...`
Constraint_name	Dans SQL Server, les noms de contrainte doivent être uniques dans le schéma auquel appartient la table. Cependant, dans Babelfish, cela ne s'applique qu'aux contraintes PRIMARY KEY et UNIQUE. Les autres types de contraintes ne sont pas soumis à cette restriction.
Constaintes	PostgreSQL ne prend pas en charge l'activation et la désactiva tion des contraintes individuelles. L'instruction est ignorée et un avertissement est émis.
Contraintes avec IGNORE_DU P_KEY	Les contraintes sont créées sans cette propriété.
CREATE, ALTER, DROP SERVER ROLE	ALTER SERVER ROLE est uniquement pris en charge pour `sysadmin`. Aucune autre syntaxe n'est prise en charge. L'utilisateur T-SQL de Babelfish a une expérience similaire à celle de SQL Server pour les concepts d'identifiant (principa l du serveur), de base de données et d'utilisateur de base de données (principal de la base de données).
Les clauses CREATE, ALTER LOGIN sont prises en charge avec une syntaxe limitée.	La clause CREATE LOGIN... PASSWORD, la clause ...DEFAUL T_DATABASE et la clause ...DEFAULT_LANGUAGE sont prises en charge. La clause ALTER LOGIN... PASSWORD est prise en charge, mais pas la clause ALTER LOGIN... OLD_PASSWORD. Seul un identifiant correspondant à un membre sysadmin peut modifier un mot de passe.
CREATE DATABASE – Classement sensible à la casse	Les classements sensibles à la casse ne sont pas pris en charge par l'instruction CREATE DATABASE.

Fonctionnalité ou syntaxe	Description du comportement ou de la différence
Mots-clés et clauses CREATE DATABASE	Les options autres que COLLATE et CONTAINMENT=NONE ne sont pas prises en charge. La clause COLLATE est acceptée et est toujours définie sur la valeur `babelfishpg_tsql.server_collation_name` .
Clauses CREATE SCHEMA...	Vous pouvez utiliser la commande CREATE SCHEMA pour créer un schéma vide. Utilisez des commandes supplémentaires pour créer des objets de schéma.
Les valeurs d'ID de base de données sont différentes sur Babelfish	Les bases de données master et tempdb ne correspondront pas aux ID de base de données 1 et 2.
La fonction de type FORMAT est prise en charge avec les limitations suivantes :	Le méridien à caractère unique n'est pas pris en charge. Le format « yyy » dans SQL Server renvoie 4 chiffres pour les années supérieures à 1 000, mais seulement 3 chiffres pour les autres. Les formats « g » et « R » ne sont pas pris en charge La traduction locale « vi-VN » est légèrement différente.
Identifiants dépassant 63 caractères	PostgreSQL prend en charge un maximum de 63 caractères pour les identifiants. Babelfish convertit les identifiants de plus de 63 caractères en un nom qui inclut un hachage du nom d'origine. Par exemple, une table créée sous la forme « AB(ABC12345678901234567890123456789012345678901234567890123456789012345678901234567890 » peut être convertie en « ABC12345678901234567890123456789012345678901234567890123456789012345678901234567890 ».

Fonctionnalité ou syntaxe	Description du comportement ou de la différence
Prise en charge des colonnes IDENTITY	Les colonnes `IDENTITY` sont prises en charge pour les types de données `tinyint`, `smallint`, `int`, `bigint`. `numeric` et `decimal`. SQL Server prend en charge une précision allant jusqu'à 38 positions pour les types de données `numeric` et `decimal` dans les colonnes IDENTITY. PostgreSQL prend en charge une précision allant jusqu'à 19 positions pour les types de données `numeric` et `decimal` dans les colonnes IDENTITY.
Index avec IGNORE_DUP_KEY	La syntaxe qui crée un index incluant IGNORE_DUP_KEY crée un index comme si cette propriété était omise.
Index comportant plus de 32 colonnes	Un index ne peut pas comporter plus de 32 colonnes. Les colonnes d'index incluses comptent pour le maximum dans PostgreSQL, mais pas dans SQL Server.
Index (en cluster)	Les index en cluster sont créés comme si NONCLUSTERED était spécifié.
Clauses d'index	Les clauses suivantes sont ignorées : FILLFACTOR, ALLOW_PAGE_LOCKS, ALLOW_ROW_LOCKS, PAD_INDEX, STATISTICS_NORECOMPUTE, OPTIMIZE_FOR_SEQUENTIAL_KEY, SORT_IN_TEMPDB, DROP_EXISTING, ONLINE, COMPRESSION_DELAY, MAXDOP et DATA_COMPRESSION
Prise en charge JSON	L'ordre des paires nom-valeur n'est pas garanti. Mais le type de table n'est pas affecté.
Objets LOGIN	Toutes les options des objets LOGIN ne sont pas prises en charge, à l'exception de PASSWORD, DEFAULT_DATABASE, DEFAULT_LANGUAGE, ENABLE, DISABLE.

Fonctionnalité ou syntaxe	Description du comportement ou de la différence
Fonction NEWSSEQUENTIALID	Implémenté en tant que NEWID ; le comportement séquentiel n'est pas garanti. Lors de l'appel de `NEWSEQUENTIALID` , PostgreSQL génère une nouvelle valeur GUID.
La clause OUTPUT est prise en charge avec les limitations suivantes	OUTPUT et OUTPUT INTO ne sont pas pris en charge dans la même requête DML. Les références à des tables non ciblées par des opérations UPDATE ou DELETE dans une clause OUTPUT ne sont pas prises en charge. OUTPUT... DELETED *, INSERTED * ne sont pas pris en charge dans la même requête.
Limite de paramètres de procédure ou de fonction	Babelfish prend en charge un maximum de 100 paramètres pour une procédure ou une fonction.
ROWGUIDCOL	Cette clause est actuellement ignorée. Les requêtes faisant référence à `$GUIDGOL` provoquent une erreur de syntaxe.
Prise en charge des objets SEQUENCE	Les objets SEQUENCE sont pris en charge pour les types de données tinyint, smallint, int, bigint, numeric et decimal. Aurora PostgreSQL prend en charge une précision allant jusqu'à 19 positions pour les types de données numeric et decimal contenus dans un objet SEQUENCE.
Rôles au niveau du serveur	Le rôle `sysadmin` au niveau du serveur est pris en charge. Les autres rôles au niveau du serveur (autres que `sysadmin`) ne sont pas pris en charge.
Rôles de niveau base de données autres que `db_owner`	Les rôles au niveau de la base de données `db_owner` et les rôles au niveau de la base de données définis par l'utilisateur sont pris en charge. Les autres rôles au niveau de la base de données (autres que db_owner) ne sont pas pris en charge.
Mot-clé SQL `SPARSE`	Le mot-clé SPARSE est accepté et ignoré.
Clause de mot-clé SQL `ON filegroup`	Cette clause est actuellement ignorée.

Fonctionnalité ou syntaxe	Description du comportement ou de la différence
Mots-clés SQL `CLUSTERED` et `NONCLUSTERED` pour les index et les contraintes	Babelfish accepte et ignore les mots-clés `CLUSTERED` et `NONCLUSTERED`
`sysdatabases.cmptlevel`	`sysdatabases.cmptlevel` est toujours défini sur 120.
La base de données tempdb n'est pas réinitialisée au redémarrage	Les objets permanents (comme les tables et les procédures) créés dans tempdb ne sont pas supprimés au redémarrage de la base de données.
Groupe de fichiers TEXTIMAGE_ON	Babelfish ignore la clause *`filegroup`* `TEXTIMAGE_ON` .
Précision temporelle	Babelfish prend en charge une précision à 6 chiffres pour les fractions de seconde. Aucun effet indésirable n'est anticipé avec ce comportement.
Niveaux d'isolement des transactions	READUNCOMMITTED est traité de la même manière que READCOMMITTED.
Colonnes virtuelles calculées (non persistantes)	Les colonnes virtuelles calculées sont créées en tant que colonnes persistantes.
Sans la clause SCHEMABINDING	Cette clause n'est pas prise en charge dans les fonctions , procédures, déclencheurs ou vues. L'objet est créé, mais comme si WITH SCHEMABINDING avait été spécifié.

Niveaux d’isolement des transactions dans Babelfish

Babelfish prend en charge les niveaux d’isolement des transactions `READ UNCOMMITTED`, `READ COMMITTED` et `SNAPSHOT`. À partir de la version 3.4 de Babelfish, les niveaux d’isolement supplémentaires `REPEATABLE READ` et `SERIALIZABLE` sont pris en charge. Tous les niveaux d’isolement de Babelfish sont pris en charge avec le comportement des niveaux d’isolement correspondants dans PostgreSQL. SQL Server et Babelfish utilisent différents mécanismes sous-jacents pour implémenter les niveaux d’isolement des transactions (blocage des accès simultanés, verrous détenus par les transactions, gestion des erreurs, etc.). Il existe aussi des différences subtiles dans la manière dont l’accès simultané peut fonctionner pour différentes charges de travail. Pour plus d’informations sur ce comportement de PostgreSQL, consultez Isolement des transactions.

Rubriques

Présentation des niveaux d’isolement des transactions

Les niveaux d’isolement des transactions SQL Server d’origine sont basés sur un verrouillage pessimiste où il n’existe qu’une seule copie des données et où les requêtes doivent verrouiller des ressources telles que les lignes avant de pouvoir y accéder. Par la suite, une variation du niveau d’isolement `READ COMMITTED` a été introduite. Cela permet d’utiliser des versions de ligne pour améliorer la simultanéité entre les lecteurs et les rédacteurs en utilisant un accès non bloquant. De plus, un nouveau niveau d’isolement appelé `SNAPSHOT` est disponible. Il utilise également des versions de ligne pour offrir une meilleure simultanéité que le niveau d’isolement `REPEATABLE READ` en évitant le verrouillage partagé des données de lecture qui sont bloquées jusqu’à la fin de la transaction.

Contrairement à SQL Server, tous les niveaux d’isolement des transactions de Babelfish sont basés sur le verrouillage optimiste (MVCC). Chaque transaction affiche un instantané des données au début de l’instruction (`READ COMMITTED`) ou au début de la transaction (`REPEATABLE READ`, `SERIALIZABLE`), quel que soit l’état actuel des données sous-jacentes. Par conséquent, le comportement d’exécution des transactions simultanées dans Babelfish peut être différent de celui de SQL Server.

Prenons l'exemple d'une transaction avec un niveau d'isolement SERIALIZABLE initialement bloqué dans SQL Server, mais qui aboutit ultérieurement. Elle peut échouer dans Babelfish en raison d'un conflit de sérialisation avec une transaction simultanée qui lit ou met à jour les mêmes lignes. Il peut également arriver que l'exécution de plusieurs transactions simultanées produise un résultat final différent dans Babelfish par rapport à SQL Server. Les scénarios de simultanéité des applications qui utilisent des niveaux d'isolement doivent être testés minutieusement.

Niveaux d'isolement dans SQL Server	Niveau d'isolement Babelfish	Niveaux d'isolement PostgreSQL	Commentaires
READ UNCOMMITTED	READ UNCOMMITTED	READ UNCOMMITTED	READ UNCOMMITTED est le même que READ COMMITTED dans Babelfish ou PostgreSQL
READ COMMITTED	READ COMMITTED	READ COMMITTED	READ COMMITTED dans SQL Server est basé sur le verrouillage pessimiste, READ COMMITTED dans Babelfish est basé sur les instantanés (MVCC).
READ COMMITTED SNAPSHOT	READ COMMITTED	READ COMMITTED	Les deux sont basés sur des instantanés (MVCC), mais ne sont pas exactement les mêmes.
SNAPSHOT	SNAPSHOT	REPEATABLE READ	Exactement pareil.
REPEATABLE READ	REPEATABLE READ	REPEATABLE READ	REPEATABLE READ dans SQL Server est basé sur le verrouillage pessimiste, REPEATABLE READ

Niveaux d’isolement dans SQL Server	Niveau d’isolement Babelfish	Niveaux d’isolement PostgreSQL	Commentaires
			dans Babelfish est basé sur les instantanés (MVCC).
`SERIALIZABLE`	`SERIALIZABLE`	`SERIALIZABLE`	`SERIALIZABLE` dans SQL Server est un système d’isolement pessimiste, `SERIALIZABLE` dans Babelfish est basé sur les instantanés (MVCC).

> **Note**
>
> Les indicateurs de table ne sont actuellement pas pris en charge, et leur comportement est contrôlé à l’aide de la trappe de secours prédéfinie `escape_hatch_table_hints` de Babelfish.

Configuration des niveaux d’isolement des transactions

Utilisez la commande suivante pour définir le niveau d’isolement des transactions :

Example

```
SET TRANSACTION ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ |
 SNAPSHOT | SERIALIZABLE }
```

Activation ou désactivation des niveaux d’isolement des transactions

Les niveaux d’isolement des transactions `REPEATABLE READ` et `SERIALIZABLE` sont désactivés par défaut dans Babelfish. Vous devez donc les activer explicitement en réglant l’utilisation de la trappe de secours `babelfishpg_tsql.isolation_level_serializable` ou

`babelfishpg_tsql.isolation_level_repeatable_read` sur `pg_isolation` à l'aide de `sp_babelfish_configure`. Pour plus d'informations, consultez Gestion du traitement des erreurs Babelfish avec des trappes de secours.

Vous trouverez ci-dessous des exemples d'activation ou de désactivation de l'utilisation de `REPEATABLE READ` et `SERIALIZABLE` dans la session en cours en définissant leur trappe de secours respective. Incluez éventuellement un paramètre `server` pour définir la trappe de secours pour la session en cours ainsi que pour toutes les nouvelles sessions suivantes.

Pour activer l'utilisation de `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` uniquement dans la session en cours.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation'
```

Pour permettre l'utilisation de `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` dans la session en cours et dans toutes les nouvelles sessions qui en découlent.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation',
 'server'
```

Pour désactiver l'utilisation de `SET TRANSACTION ISOLATION LEVEL REPEATABLE READ` dans la session en cours et dans les nouvelles sessions qui en découlent.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'off', 'server'
```

Pour activer l'utilisation de `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` uniquement dans la session en cours.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation'
```

Pour permettre l'utilisation de `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` dans la session en cours et dans toutes les nouvelles sessions qui en découlent.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation', 'server'
```

Pour désactiver l'utilisation de `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE` dans la session en cours et dans les nouvelles sessions qui en découlent.

Example

```
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'off', 'server'
```

Comparaison des niveaux d'isolement Babelfish et SQL Server

Vous trouverez ci-dessous quelques exemples illustrant les nuances dans la manière dont SQL Server et Babelfish implémentent les niveaux d'isolement ANSI.

> **Note**
>
> - Les niveaux d'isolement `REPEATABLE READ` et `SNAPSHOT` sont les mêmes dans Babelfish.
> - Les niveaux d'isolement `READ UNCOMMITTED` et `READ COMMITTED` sont les mêmes dans Babelfish.

L'exemple suivant montre comment créer la table de base pour tous les exemples mentionnés ci-dessous :

```
CREATE TABLE employee (
    id sys.INT NOT NULL PRIMARY KEY,
    name sys.VARCHAR(255)NOT NULL,
    age sys.INT NOT NULL
);
INSERT INTO employee (id, name, age) VALUES (1, 'A', 10);
INSERT INTO employee (id, name, age) VALUES (2, 'B', 20);
INSERT INTO employee (id, name, age) VALUES (3, 'C', 30);
```

Rubriques

Comparaison de **READ UNCOMMITTED** dans Babelfish avec le niveau d'isolement **READ UNCOMMITTED** de SQL Server

Le tableau suivant fournit des détails sur les lectures incorrectes lors de l'exécution de transactions simultanées. Il montre les résultats observés lors de l'utilisation du niveau d'isolement `READ UNCOMMITTED` dans SQL Server par rapport à l'implémentation Babelfish.

Transaction 1	Transaction 2	**READ UNCOMMITTED** dans SQL Server	**READ UNCOMMITTED** dans Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;`	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`UPDATE employee SET age=0;`	Mise à jour réussie.	Mise à jour réussie.
État Idle in transaction (Transaction inactive)	`INSERT INTO employee VALUES (4, 'D', 40);`	Insertion réussie.	Insertion réussie.

Transaction 1	Transaction 2	**READ UNCOMMITTED** dans SQL Server	**READ UNCOMMITTED** dans Babelfish
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	La transaction 1 peut voir les modifications non validées de la transaction 2.	Comme `READ COMMITTED` dans Babelfish. Les modifications non validées de la transaction 2 ne sont pas visibles pour la transaction 1.
État Idle in transaction (Transaction inactive)	`COMMIT`	Aucun	Aucun
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	Voit les modifications validées par la transaction 2.	Voit les modifications validées par la transaction 2.

Comparaison de **`READ COMMITTED`** dans Babelfish avec le niveau d'isolement **`READ COMMITTED`** de SQL Server

Le tableau suivant fournit des détails sur le comportement de blocage en lecture-écriture lorsque des transactions simultanées sont exécutées. Il montre les résultats observés lors de l'utilisation du niveau d'isolement `READ COMMITTED` dans SQL Server par rapport à l'implémentation Babelfish.

Transaction 1	Transaction 2	**READ COMMITTED** dans SQL Server	**READ COMMITTED** dans Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Aucun	Aucun

Transaction 1	Transaction 2	**READ COMMITTED** dans SQL Server	**READ COMMITTED** dans Babelfish
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`UPDATE employee SET age=100 WHERE id = 1;`	Mise à jour réussie.	Mise à jour réussie.
`UPDATE employee SET age = 0 WHERE age IN (SELECT MAX(age) FROM employee);`	État Idle in transaction (Transaction inactive)	Étape bloquée jusqu'à ce que la transacti on 2 soit validée.	Les modifications de la transaction 2 ne sont pas encore visibles. Met à jour la ligne avec id=3.
État Idle in transaction (Transaction inactive)	`COMMIT`	La transaction 2 est validée avec succès. La transaction 1 est maintenant débloquée et voit la mise à jour de la transaction 2.	La transaction 2 est validée avec succès.
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	La transaction 1 met à jour la ligne avec id = 1.	La transaction 1 met à jour la ligne avec id = 3.

Comparaison de **`READ COMMITTED`** dans Babelfish avec le niveau d'isolement **`READ COMMITTED SNAPSHOT`** de SQL Server

Le tableau suivant fournit des détails sur le comportement de blocage des lignes nouvellement insérées lorsque des transactions simultanées sont exécutées. Il montre les résultats observés lors de l'utilisation du niveau d'isolement `READ COMMITTED SNAPSHOT` dans SQL Server par rapport à l'implémentation de `READ COMMITTED` dans Babelfish.

Transaction 1	Transaction 2	**READ COMMITTED SNAPSHOT** dans SQL Server	**READ COMMITTED** dans Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	`SET TRANSACTION ISOLATION LEVEL READ COMMITTED;`	Aucun	Aucun
`INSERT INTO employee VALUES (4, 'D', 40);`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`UPDATE employee SET age = 99;`	Étape bloquée jusqu'à ce que la transaction 1 soit validée. La ligne insérée est verrouillée par la transaction 1.	Trois lignes mises à jour. La ligne nouvellement insérée n'est pas encore visible.
`COMMIT`	État Idle in transaction (Transaction inactive)	Validation réussie. La transaction 2 est désormais débloquée.	Validation réussie.
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee;`	Les 4 lignes ont le valeur age=99.	La ligne avec id = 4 a la valeur d'âge 40, car elle n'était pas visible pour la transaction 2 lors de la requête de mise à jour. Les autres lignes sont mises à jour avec age=99.

Comparaison de **`REPEATABLE READ`** dans Babelfish avec le niveau d'isolement **`REPEATABLE READ`** de SQL Server

Le tableau suivant fournit des détails sur le comportement de blocage en lecture-écriture lorsque des transactions simultanées sont exécutées. Il montre les résultats observés lors de l'utilisation du niveau d'isolement `REPEATABLE READ` dans SQL Server par rapport à l'implémentation de `REPEATABLE READ` dans Babelfish.

Transaction 1	Transaction 2	**`REPEATABLE READ`** dans SQL Server	**`REPEATABLE READ`** dans Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Aucun	Aucun
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
`UPDATE employee SET name='A_TXN1' WHERE id=1;`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee WHERE id != 1;`	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee;`	La transaction 2 est bloquée jusqu'à ce que la transaction 1 soit validée.	La transaction 2 se déroule normalement.
`COMMIT`	État Idle in transaction (Transaction inactive)	Aucun	Aucun

Transaction 1	Transaction 2	**REPEATABLE READ** dans SQL Server	**REPEATABLE READ** dans Babelfish
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee;`	La mise à jour de la transaction 1 est visible.	La mise à jour de la transaction 1 n'est pas visible.
`COMMIT`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee;`	voit la mise à jour de la transaction 1.	voit la mise à jour de la transaction 1.

Le tableau suivant fournit des détails sur le comportement de blocage de l'écriture-écriture lorsque des transactions simultanées sont exécutées. Il montre les résultats observés lors de l'utilisation du niveau d'isolement `REPEATABLE READ` dans SQL Server par rapport à l'implémentation de `REPEATABLE READ` dans Babelfish.

Transaction 1	Transaction 2	**REPEATABLE READ** dans SQL Server	**REPEATABLE READ** dans Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Aucun	Aucun
`UPDATE employee SET name='A_TXN1' WHERE id=1;`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`UPDATE employee SET name='A_T`	Transaction 2 bloquée.	Transaction 2 bloquée.

Transaction 1	Transaction 2	**REPEATABLE READ** dans SQL Server	**REPEATABLE READ** dans Babelfish
	`XN2' WHERE id=1;`		
`COMMIT`	État Idle in transaction (Transaction inactive)	La validation a été effectuée avec succès et la transaction 2 a été débloquée.	La validation a été effectuée avec succès et la transaction 2 échoue avec une erreur signalant que l'accès n'a pas pu être sérialisé en raison d'une mise à jour simultanée.
État Idle in transaction (Transaction inactive)	`COMMIT`	Validation réussie.	La transaction 2 a déjà été abandonnée.
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee;`	La ligne avec id=1 contient name='A_TX2'.	La ligne avec id=1 contient name='A_TX1'.

Le tableau suivant fournit des détails sur le comportement des lectures fantômes lorsque des transactions simultanées sont exécutées. Il montre les résultats observés lors de l'utilisation du niveau d'isolement `REPEATABLE READ` dans SQL Server par rapport à l'implémentation de `REPEATABLE READ` dans Babelfish.

Transaction 1	Transaction 2	**REPEATABLE READ** dans SQL Server	**REPEATABLE READ** dans Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Aucun	Aucun
`SET TRANSACTION ISOLATION`	`SET TRANSACTION ISOLATION`	Aucun	Aucun

Transaction 1	Transaction 2	**REPEATABLE READ** dans SQL Server	**REPEATABLE READ** dans Babelfish
`LEVEL REPEATABL E READ;`	`LEVEL REPEATABL E READ;`		
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`INSERT INTO employee VALUES (4, 'NewRowNa me', 20);`	La transaction 2 se déroule sans aucun blocage.	La transaction 2 se déroule sans aucun blocage.
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee;`	La ligne nouvellement insérée est visible.	La ligne nouvellement insérée est visible.
État Idle in transaction (Transaction inactive)	`COMMIT`	Aucun	Aucun
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	La nouvelle ligne insérée par la transaction 2 est visible.	La nouvelle ligne insérée par la transaction 2 n’est pas visible.
`COMMIT`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	La ligne nouvellement insérée est visible.	La ligne nouvellement insérée est visible.

Le tableau suivant fournit des détails lorsque des transactions simultanées sont exécutées, et indique les différents résultats finaux lors de l’utilisation du niveau d’isolement `REPEATABLE READ` dans SQL Server par rapport à l’implémentation Babelfish `REPEATABLE READ`.

Transaction 1	Transaction 2	**REPEATABLE READ** dans SQL Server	**REPEATABLE READ** dans Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Aucun	Aucun
`UPDATE employee SET age = 100 WHERE age IN (SELECT MIN(age) FROM employee);`	État Idle in transaction (Transaction inactive)	La transaction 1 met à jour la ligne avec l'identifiant 1.	La transaction 1 met à jour la ligne avec l'identifiant 1.
État Idle in transaction (Transaction inactive)	`UPDATE employee SET age = 0 WHERE age IN (SELECT MAX(age) FROM employee);`	La transaction 2 est bloquée, car l'instruction SELECT tente de lire les lignes verrouillées par la requête UPDATE dans la transaction 1.	La transaction 2 se déroule sans aucun blocage, car la lecture n'est jamais bloquée, l'instruction SELECT s'exécute et la ligne avec id = 3 finira par être mise à jour, car les modifications de la transaction 1 ne sont pas encore visibles.
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee;`	Cette étape est exécutée une fois la transaction 1 validée. La ligne avec id = 1 est mise à jour par la transaction 2 à l'étape précédente et est visible ici.	La ligne avec id = 3 est mise à jour par la transaction 2.

Transaction 1	Transaction 2	**REPEATABLE READ** dans SQL Server	**REPEATABLE READ** dans Babelfish
`COMMIT`	État Idle in transaction (Transaction inactive)	La transaction 2 est désormais débloquée.	Validation réussie.
État Idle in transaction (Transaction inactive)	`COMMIT`	Aucun	Aucun
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	Les deux transacti ons exécutent la mise à jour sur la ligne contenant id = 1.	Les différentes lignes sont mises à jour par les transactions 1 et 2.

Comparaison de **`SERIALIZABLE`** dans Babelfish avec le niveau d'isolement **`SERIALIZABLE`** de SQL Server

Le tableau suivant fournit des détails sur les blocages de plage lorsque des transactions simultanées sont exécutées. Il montre les résultats observés lors de l'utilisation du niveau d'isolement `SERIALIZABLE` dans SQL Server par rapport à l'implémentation de `SERIALIZABLE` dans Babelfish.

Transaction 1	Transaction 2	**SERIALIZABLE** dans SQL Server	**SERIALIZABLE** dans Babelfish
`BEGIN TRANSACTI ON`	`BEGIN TRANSACTI ON`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Aucun	Aucun
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`INSERT INTO employee VALUES (4, 'D', 35);`	La transaction 2 est bloquée jusqu'à ce	La transaction 2 se déroule sans aucun blocage.

Transaction 1	Transaction 2	**SERIALIZABLE** dans SQL Server	**SERIALIZABLE** dans Babelfish
		que la transaction 1 soit validée.	
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee;`	Aucun	Aucun
`COMMIT`	État Idle in transaction (Transaction inactive)	La transaction 1 est validée avec succès. La transaction 2 est désormais débloquée.	La transaction 1 est validée avec succès.
État Idle in transaction (Transaction inactive)	`COMMIT`	Aucun	Aucun
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	La ligne nouvellement insérée est visible.	La ligne nouvellement insérée est visible.

Le tableau suivant fournit des détails lorsque des transactions simultanées sont exécutées, et indique les différents résultats finaux lors de l'utilisation du niveau d'isolement `SERIALIZABLE` dans SQL Server par rapport à l'implémentation Babelfish `SERIALIZABLE`.

Transaction 1	Transaction 2	**SERIALIZABLE** dans SQL Server	**SERIALIZABLE** dans Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`INSERT INTO employee VALUES (4, 'D', 40);`	Aucun	Aucun

Transaction 1	Transaction 2	**SERIALIZABLE** dans SQL Server	**SERIALIZABLE** dans Babelfish
`UPDATE employee SET age =99 WHERE id = 4;`	État Idle in transaction (Transaction inactive)	La transaction 1 est bloquée jusqu'à ce que la transaction 2 soit validée.	La transaction 1 se déroule sans aucun blocage.
État Idle in transaction (Transaction inactive)	`COMMIT`	La transaction 2 est validée avec succès. La transaction 1 est désormais débloquée.	La transaction 2 est validée avec succès.
`COMMIT`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	La ligne nouvellem ent insérée est visible avec une valeur d'âge = 99.	La ligne nouvellem ent insérée est visible avec une valeur d'âge = 40.

Le tableau suivant fournit des informations détaillées lorsque vous exécutez une instruction `INSERT` dans une table avec une contrainte unique. Il montre les résultats observés lors de l'utilisation du niveau d'isolement `SERIALIZABLE` dans SQL Server par rapport à l'implémentation de `SERIALIZABLE` dans Babelfish.

Transaction 1	Transaction 2	**SERIALIZABLE** dans SQL Server	**SERIALIZABLE** dans Babelfish
`BEGIN TRANSACTI ON`	`BEGIN TRANSACTI ON`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Aucun	Aucun

Transaction 1	Transaction 2	**SERIALIZABLE** dans SQL Server	**SERIALIZABLE** dans Babelfish
État Idle in transaction (Transaction inactive)	`INSERT INTO employee VALUES (4, 'D', 40);`	Aucun	Aucun
`INSERT INTO employee VALUES ((SELECT MAX(id)+1 FROM employee), 'E', 50);`	État Idle in transaction (Transaction inactive)	La transaction 1 est bloquée jusqu'à ce que la transaction 2 soit validée.	La transaction 1 est bloquée jusqu'à ce que la transaction 2 soit validée.
État Idle in transaction (Transaction inactive)	`COMMIT`	La transaction 2 est validée avec succès. La transaction 1 est désormais débloquée.	La transaction 2 est validée avec succès. La transaction 1 a été abandonnée avec une erreur de valeur de clé en double qui va à l'encontre de la contrainte unique.
`COMMIT`	État Idle in transaction (Transaction inactive)	La transaction 1 est validée avec succès.	Les validations de la transaction 1 échouent, car l'accès n'a pas pu être sérialisé en raison de dépendances en lecture ou en écriture entre les transactions.
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	La ligne (5, 'E', 50) est insérée.	Il n'existe que 4 lignes.

Dans Babelfish, les transactions simultanées exécutées avec le niveau d'isolement serializable échouent avec une erreur d'anomalie de sérialisation si l'exécution de ces transactions est incompatible avec toutes les exécutions en série (une par une) possibles de ces transactions.

Les tableaux suivants fournissent des détails sur les anomalies de sérialisation lors de l'exécution de transactions simultanées. Il montre les résultats observés lors de l'utilisation du niveau d'isolement `SERIALIZABLE` dans SQL Server par rapport à l'implémentation de `SERIALIZABLE` dans Babelfish.

Transaction 1	Transaction 2	**`SERIALIZABLE`** dans SQL Server	**`SERIALIZABLE`** dans Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	Aucun	Aucun
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
`UPDATE employee SET age=5 WHERE age=10;`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee;`	La transaction 2 est bloquée jusqu'à ce que la transaction 1 soit validée.	La transaction 2 se déroule sans aucun blocage.
État Idle in transaction (Transaction inactive)	`UPDATE employee SET age=35 WHERE age=30;`	Aucun	Aucun
`COMMIT`	État Idle in transaction (Transaction inactive)	La transaction 1 est validée avec succès.	La transaction 1 est validée en premier et peut être validée avec succès.

Transaction 1	Transaction 2	**SERIALIZABLE** dans SQL Server	**SERIALIZABLE** dans Babelfish
État Idle in transaction (Transaction inactive)	`COMMIT`	La transaction 2 est validée avec succès.	La validation de la transaction 2 échoue avec une erreur de sérialisation, l'ensemble de la transaction a été annulée. Réessayez la transaction 2.
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	Les modifications apportées par les deux transactions sont visibles.	La transaction 2 a été annulée. Seules les modifications de la transaction 1 sont visibles.

Dans Babelfish, l'anomalie de sérialisation n'est possible que si toutes les transactions simultanées s'exécutent au niveau d'isolement `SERIALIZABLE`. Dans le tableau suivant, prenons l'exemple ci-dessus, mais définissons plutôt la transaction 2 sur le niveau d'isolement `REPEATABLE READ`.

Transaction 1	Transaction 2	Niveaux d'isolement SQL Server	Niveaux d'isolement Babelfish
`BEGIN TRANSACTION`	`BEGIN TRANSACTION`	Aucun	Aucun
`SET TRANSACTION ISOLATION LEVEL SERILAIZABLE;`	`SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;`	Aucun	Aucun
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	Aucun	Aucun

Transaction 1	Transaction 2	Niveaux d’isolement SQL Server	Niveaux d’isolement Babelfish
`UPDATE employee SET age=5 WHERE age=10;`	État Idle in transaction (Transaction inactive)	Aucun	Aucun
État Idle in transaction (Transaction inactive)	`SELECT * FROM employee;`	La transaction 2 est bloquée jusqu’à ce que la transaction 1 soit validée.	La transaction 2 se déroule sans aucun blocage.
État Idle in transaction (Transaction inactive)	`UPDATE employee SET age=35 WHERE age=30;`	Aucun	Aucun
`COMMIT`	État Idle in transaction (Transaction inactive)	La transaction 1 est validée avec succès.	La transaction 1 est validée avec succès.
État Idle in transaction (Transaction inactive)	`COMMIT`	La transaction 2 est validée avec succès.	La transaction 2 est validée avec succès.
`SELECT * FROM employee;`	État Idle in transaction (Transaction inactive)	Les modifications apportées par les deux transactions sont visibles.	Les modifications apportées par les deux transactions sont visibles.

Utilisation des fonctionnalités Babelfish dont la mise en œuvre est limitée

Chaque nouvelle version de Babelfish ajoute la prise en charge d’autres fonctionnalités qui s’alignent mieux avec la fonctionnalité et le comportement de T-SQL. Cependant, il existe certaines fonctionnalités non prises en charge et des différences dans la mise en œuvre actuelle. Dans ce qui suit, vous pouvez trouver des informations sur les différences fonctionnelles entre Babelfish et T-SQL, avec quelques solutions de contournement ou des notes d’utilisation.

À partir de la version 1.2.0 de Babelfish, les fonctionnalités suivantes ont actuellement des mises en œuvres limitées :

- Catalogues SQL Server (vues système) — les catalogues `sys.sysconfigures`, `sys.syscurconfigs` et `sys.configurations` ne prennent en charge qu'une seule configuration en lecture seule. Le catalogue `sp_configure` n'est actuellement pas pris en charge. Pour plus d'informations sur les autres vues SQL Server mises en œuvre par Babelfish, consultez Obtention d'informations dans le catalogue système Babelfish.
- GRANT permissions (ACCORDER des autorisations) — la fonction GRANT...TO PUBLIC est prise en charge, mais la fonction GRANT...TO PUBLIC WITH GRANT OPTION n'est actuellement pas prise en charge.
- SQL Server ownership chain and permission mechanism limitation (Limitation de la chaîne de propriété et du mécanisme d'autorisation de SQL Server) — dans Babelfish, la chaîne de propriété de SQL Server fonctionne pour les vues mais pas pour les procédures stockées. Cela signifie que les procédures doivent disposer d'un accès explicite aux autres objets appartenant au même propriétaire que les procédures appelantes. Dans SQL Server, accorder à l'appelant les autorisations EXECUTE sur la procédure est suffisant pour appeler d'autres objets appartenant au même propriétaire. Dans Babelfish, l'appelant doit également disposer d'autorisations sur les objets auxquels la procédure accède.
- Resolution of unqualified (without schema name) object references (Résolution des références d'objets non qualifiées (sans nom de schéma)) — lorsqu'un objet SQL (procédure, vue, fonction ou déclencheur) fait référence à un objet sans le qualifier avec un nom de schéma, SQL Server résout le nom de schéma de l'objet en utilisant le nom de schéma de l'objet SQL dans lequel la référence se produit. Actuellement, Babelfish résout ce problème différemment, en utilisant le schéma par défaut de l'utilisateur de la base de données qui exécute la procédure.
- Default schema changes, sessions, and connections (Modifications du schéma par défaut, sessions et connexions) — si les utilisateurs modifient leur schéma par défaut avec `ALTER USER...WITH DEFAULT SCHEMA`, la modification prend effet immédiatement dans cette session. Cependant, pour les autres sessions actuellement connectées appartenant au même utilisateur, le timing diffère, comme suit :
  - Pour SQL Server : — la modification prend effet immédiatement sur toutes les autres connexions de cet utilisateur.
  - Pour Babelfish : — Le changement prend effet pour cet utilisateur pour les nouvelles connexions seulement.
- ROWVERSION and TIMESTAMP datatypes implementation and escape hatch setting (Mise en œuvre des types de données ROWVERSION et TIMESTAMP et réglage des trappes de secours) — les types de données ROWVERSION et TIMESTAMP sont maintenant pris en charge dans Babelfish. Pour utiliser ROWVERSION ou TIMESTAMP dans Babelfish, vous devez modifier

le paramètre de la trappe de secours `babelfishpg_tsql.escape_hatch_rowversion` de sa valeur par défaut (strict) à `ignore`. La mise en œuvre Babelfish des types de données ROWVERSION et TIMESTAMP est sémantiquement identique à celle de SQL Server, avec les exceptions suivantes :

- La fonction @@DBTS intégrée se comporte de la même manière que SQL Server, mais avec de petites différences. Plutôt que de renvoyer la dernière valeur utilisée pour `SELECT @@DBTS`, Babelfish génère un nouvel horodatage, en raison du moteur de base de données PostgreSQL sous-jacent et de son implémentation MVCC (Multiversion Concurrency Control).
- Dans SQL Server, chaque ligne insérée ou mise à jour reçoit une ROWVERSION/TIMESTAMP value. In Babelfish, every inserted row updated by the same statement is assigned the same ROWVERSION/TIMESTAMP valeur unique.

  Par exemple, lorsqu'une instruction UPDATE ou une instruction INSERT-SELECT affecte plusieurs lignes, dans SQL Server, les lignes concernées ont toutes des valeurs différentes dans leur ROWVERSION/TIMESTAMP colonne. Dans Babelfish (PostgreSQL), les lignes ont la même valeur.
- Dans SQL Server, lorsque vous créez une nouvelle table avec SELECT-INTO, vous pouvez convertir une valeur explicite (telle que NULL) en une to-be-created ROWVERSION/TIMESTAMP column. When you do the same thing in Babelfish, an actual ROWVERSION/TIMESTAMP valeur attribuée à chaque ligne de la nouvelle table pour vous, par Babelfish.

Ces différences mineures dans les ROWVERSION/TIMESTAMP types de données ne devraient pas avoir d'impact négatif sur les applications exécutées sur Babelfish.

- Clause TOP N PERCENT — Babelfish fournit un support pour la clause TOP N PERCENT avec certaines limitations. Les opérations SELECT sont prises en charge tandis que les opérations UPDATE, DELETE et INSERT avec TOP N PERCENT ne sont pas prises en charge. L'option WITH TIES et les sous-requêtes de la clause TOP ne sont pas non plus prises en charge. Lorsque la valeur de l'expression est supérieure à 100, le comportement est différent :
  - Pour SQL Server : génère une erreur.
  - Pour Babelfish : traite la valeur comme valide et renvoie les résultats.
- Création de schéma, propriété et autorisations : les autorisations pour créer des objets et y accéder dans un schéma détenu par un utilisateur non-DBO (à l'aide de `CREATE SCHEMA` *`schema name`* `AUTHORIZATION` *`user name`*) diffèrent pour les utilisateurs de SQL Server et les utilisateurs de Babelfish non-DBO, comme le montre le tableau suivant :

L'utilisateur de base de données (non-DBO) qui possède le schéma peut effectuer les opérations suivantes :	SQL Server	Babelfish
Créer des objets dans le schéma sans subventions supplémentaires par le DBO ?	Non	Oui
Accéder aux objets créés par DBO dans le schéma sans octrois supplémentaires ?	Oui	Non

- Syntaxe CREATE OR ALTER VIEW/ALTER VIEW — La prise en charge de ces syntaxes dans Babelfish présente les limites suivantes :
  - Ces instructions ne peuvent pas être utilisées sur les vues associées à un déclencheur INSTEAD-OF.
  - Ces instructions ne peuvent pas être utilisées sur les vues ayant une autre vue basée sur cette vue.

Amélioration des performances des requêtes Babelfish

Vous pouvez accélérer le traitement des requêtes dans Babelfish à l'aide d'indicateurs de requête et de l'optimiseur PostgreSQL.

Rubriques

- Utilisation du plan d'explication pour améliorer les performances des requêtes Babelfish
- Utilisation des indicateurs de requête T-SQL pour améliorer les performances des requêtes Babelfish

Vous pouvez également améliorer les performances des requêtes à l'aide de la procédure `sp_babelfish_volatility`. Pour plus d'informations, consultez sp_babelfish_volatility.

Vous pouvez également améliorer les performances des requêtes à l'aide de la transformation des sous-requêtes et du cache des sous-requêtes. Pour plus d'informations, consultez Optimisation des sous-requêtes corrélées dans Aurora PostgreSQL.

Utilisation du plan d'explication pour améliorer les performances des requêtes Babelfish

À partir de la version 2.1.0, Babelfish inclut deux fonctions qui utilisent de manière transparente l'optimiseur PostgreSQL pour générer des plans de requêtes estimés et réels pour les requêtes T-SQL sur le port TDS. Ces fonctions sont similaires à l'utilisation de SET STATISTICS PROFILE ou de SET SHOWPLAN_ALL avec des bases de données SQL Server pour identifier et améliorer les requêtes s'exécutant lentement.

> **Note**
>
> L'obtention de plans de requête à partir de fonctions, de flux de contrôle et de curseurs n'est actuellement pas prise en charge.

Le tableau fournit une comparaison des fonctions d'explication de plan de requête sur SQL Server, Babelfish et PostgreSQL.

SQL Server	Babelfish	PostgreSQL
SHOWPLAN_ALL	BABELFISH_SHOWPLAN_ALL	EXPLAIN
STATISTICS PROFILE	BABELFISH_STATISTICS PROFILE	EXPLAIN ANALYZE
Utilise l'optimiseur SQL Server	Utilise l'optimiseur PostgreSQL	Utilise l'optimiseur PostgreSQL
Format d'entrée et de sortie SQL Server	Format d'entrée SQL Server et de sortie PostgreSQL	Format d'entrée et de sortie PostgreSQL
Défini pour la session	Défini pour la session	Appliquer à une instruction spécifique
Prend en charge : • SELECT • INSERT	Prend en charge : • SELECT • INSERT	Prend en charge : • SELECT • INSERT

SQL Server	Babelfish	PostgreSQL
• UPDATE • DELETE • CURSOR • CREATE • EXECUTE • EXEC et fonctions, y compris le flux de contrôle (CASE, WHILE-BREAK-CONTINUE, WAITFOR, BEGIN-END, IF-ELSE, etc.)	• UPDATE • DELETE • CREATE • EXECUTE • EXEC • RAISEERROR • THROW • PRINT • USE	• UPDATE • DELETE • CURSOR • CREATE • EXECUTE

Utilisez les fonctions Babelfish comme suit :

- SET BABELFISH_SHOWPLAN_ALL [ON|OFF] : définissez la valeur ON pour générer un plan d'exécution de requête estimé. Cette fonction implémente le comportement de la commande PostgreSQL `EXPLAIN`. Utilisez cette commande pour obtenir le plan d'explication pour une requête donnée.
- SET BABELFISH_STATISTICS PROFILE [ON|OFF] : définissez la valeur ON pour les plans d'exécution de requête réels. Cette fonction implémente le comportement de la commande PostgreSQL `EXPLAIN ANALYZE`.

Pour plus d'informations sur `EXPLAIN` et `EXPLAIN ANALYZE` de PostgreSQL, consultez EXPLAIN dans la documentation PostgreSQL.

> **Note**
>
> À partir de la version 2.2.0, vous pouvez définir le paramètre `escape_hatch_showplan_all` sur ignore afin d'éviter l'utilisation du préfixe BABELFISH_ dans la syntaxe SQL Server pour les commandes SET `SHOWPLAN_ALL` et `STATISTICS PROFILE`.

Par exemple, la séquence de commandes suivante active la planification des requêtes, puis renvoie un plan d'exécution de requête estimé pour l'instruction SELECT sans exécuter la requête. Cet exemple utilise l'exemple de base de données SQL Server `northwind` qui utilise l'outil de ligne de commande `sqlcmd` pour interroger le port TDS :

```
1> SET BABELFISH_SHOWPLAN_ALL ON
2> GO
1> SELECT t.territoryid, e.employeeid FROM
2> dbo.employeeterritories e, dbo.territories t
3> WHERE e.territoryid=e.territoryid ORDER BY t.territoryid;
4> GO

QUERY PLAN


------------------------------------------------------------------------------


Query Text: SELECT t.territoryid, e.employeeid FROM
dbo.employeeterritories e, dbo.territories t
WHERE e.territoryid=e.territoryid ORDER BY t.territoryid
Sort  (cost=6231.74..6399.22 rows=66992 width=10)
  Sort Key: t.territoryid NULLS FIRST
  ->  Nested Loop  (cost=0.00..861.76 rows=66992 width=10)
        ->  Seq Scan on employeeterritories e  (cost=0.00..22.70 rows=1264 width=4)
              Filter: ((territoryid)::"varchar" IS NOT NULL)
        ->  Materialize  (cost=0.00..1.79 rows=53 width=6)
              ->  Seq Scan on territories t  (cost=0.00..1.53 rows=53 width=6)
```

Lorsque vous avez terminé d'examiner et d'ajuster votre requête, désactivez la fonction comme indiqué ci-dessous :

```
1> SET BABELFISH_SHOWPLAN_ALL OFF
```

Lorsque BABELFISH_STATISTICS PROFILE est défini sur ON, chaque requête exécutée renvoie son jeu de résultats normal suivi d'un jeu de résultats supplémentaire qui affiche les plans d'exécution de requêtes réels. Babelfish génère le plan de requête qui fournit le jeu de résultats le plus rapide lorsqu'il appelle l'instruction SELECT.

```
1> SET BABELFISH_STATISTICS PROFILE ON
1>
```

```
2> GO
1> SELECT e.employeeid, t.territoryid FROM
2> dbo.employeeterritories e, dbo.territories t
3> WHERE t.territoryid=e.territoryid ORDER BY t.territoryid;
4> GO
```

Le jeu de résultats et le plan de requête sont renvoyés (cet exemple montre uniquement le plan de requête).

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT e.employeeid, t.territoryid FROM
dbo.employeeterritories e, dbo.territories t
WHERE t.territoryid=e.territoryid ORDER BY t.territoryid
Sort  (cost=42.44..43.28 rows=337 width=10)
  Sort Key: t.territoryid NULLS FIRST


  ->  Hash Join  (cost=2.19..28.29 rows=337 width=10)
       Hash Cond: ((e.territoryid)::"varchar" = (t.territoryid)::"varchar")
        ->  Seq Scan on employeeterritories e  (cost=0.00..22.70 rows=1270 width=36)
        ->  Hash  (cost=1.53..1.53 rows=53 width=6)
             ->  Seq Scan on territories t  (cost=0.00..1.53 rows=53 width=6)
```

Pour en savoir plus sur l'analyse de vos requêtes et des résultats renvoyés par l'optimiseur PostgreSQL, consultez explain.depesz.com. Pour plus d'informations sur EXPLAIN et EXPLAIN ANALYZE de PostgreSQL, consultez EXPLAIN dans la documentation PostgreSQL.

Paramètres qui contrôlent les options d'explication Babelfish

Vous pouvez utiliser les paramètres du tableau suivant pour contrôler le type d'informations affichées par votre plan de requête.

Paramètre	Description
babelfishpg_tsql.explain_buffers	Booléen qui active (et désactive) les informations d'utilisation du tampon pour l'optimiseur. (Par défaut : off) (Autorisé : off, on)

Paramètre	Description
babelfishpg_tsql.explain_costs	Booléen qui active (et désactive) les informations de coût total et de démarrage estimé pour l'optimiseur. (Par défaut : on) (Autorisé : off, on)
babelfishpg_tsql.explain_format	Spécifie le format de sortie pour le plan `EXPLAIN`. (Par défaut : text) (Autorisé : text, xml, json, yaml)
babelfishpg_tsql.explain_settings	Booléen qui active (ou désactive) l'inclusion d'informations sur les paramètres de configuration dans la sortie du plan EXPLAIN. (Par défaut : off) (Autorisé : off, on)
babelfishpg_tsql.explain_summary	Booléen qui active (ou désactive) des informations récapitulatives telles que le temps total après le plan de requête. (Par défaut : on) (Autorisé : off, on)
babelfishpg_tsql.explain_timing	Booléen qui active (ou désactive) l'heure de démarrage réelle et le temps passé dans chaque nœud de la sortie. (Par défaut : on) (Autorisé : off, on)
babelfishpg_tsql.explain_verbose	Booléen qui active (ou désactive) la version la plus détaillée d'un plan d'explication. (Par défaut : off) (Autorisé : off, on)
babelfishpg_tsql.explain_wal	Booléen qui active (ou désactive) la génération d'informations de registre WAL dans le cadre d'un plan d'explication. (Par défaut : off) (Autorisé : off, on)

Vous pouvez vérifier les valeurs de n'importe quel paramètre lié à Babelfish sur votre système à l'aide du client PostgreSQL ou du client SQL Server. Exécutez la commande suivante pour obtenir la valeur de vos paramètres actuels :

```
1> execute sp_babelfish_configure '%explain%';
2> GO
```

Dans la sortie suivante, vous observez que tous les paramètres de ce cluster de bases de données Babelfish particulier sont définis sur leurs valeurs par défaut. Les résultats ne sont pas tous affichés dans cet exemple.

```
             name                    setting                     short_desc
--------------------------------- --------
 ---------------------------------------------------------
babelfishpg_tsql.explain_buffers   off      Include information on buffer usage
babelfishpg_tsql.explain_costs     on       Include information on estimated startup
 and total cost
babelfishpg_tsql.explain_format    text     Specify the output format, which can be
 TEXT, XML, JSON, or YAML
babelfishpg_tsql.explain_settings  off      Include information on configuration
 parameters
babelfishpg_tsql.explain_summary   on       Include summary information (e.g.,totaled
 timing information) after the query plan
babelfishpg_tsql.explain_timing    on       Include actual startup time and time spent
 in each node in the output
babelfishpg_tsql.explain_verbose   off      Display additional information regarding
 the plan
babelfishpg_tsql.explain_wal       off      Include information on WAL record
 generation

(8 rows affected)
```

Vous pouvez modifier la valeur de ces paramètres avec `sp_babelfish_configure`, comme illustré dans l'exemple suivant.

```
1> execute sp_babelfish_configure 'explain_verbose', 'on';
2> GO
```

Si vous voulez rendre les valeurs permanentes à l'échelle d'un cluster, ajoutez le mot-clé server, comme indiqué dans l'exemple suivant.

```
1> execute sp_babelfish_configure 'explain_verbose', 'on', 'server';
2> GO
```

Utilisation des indicateurs de requête T-SQL pour améliorer les performances des requêtes Babelfish

À partir de la version 2.3.0, Babelfish prend en charge l'utilisation des indicateurs de requête avec `pg_hint_plan`. Dans Aurora PostgreSQL, `pg_hint_plan` est installé par défaut. Pour plus d'informations sur l'extension PostgreSQL `pg_hint_plan`, consultez https://github.com/ossc-db/pg_hint_plan. Pour plus de détails sur la version de cette extension prise en charge par Aurora PostgreSQL, consultez Extension versions for Amazon Aurora PostgreSQL (Versions des extensions pour Amazon Aurora PostgreSQL) dans les Notes de mise à jour pour Aurora PostgreSQL.

L'optimiseur de requêtes est conçu pour rechercher le plan d'exécution optimal pour une instruction SQL. Lors de la sélection d'un plan, l'optimiseur de requêtes prend en compte à la fois le modèle de coût du moteur et les statistiques des colonnes et des tables. Toutefois, le plan suggéré peut ne pas répondre aux besoins de vos jeux de données. Ainsi, les indicateurs de requête résolvent les problèmes de performances afin d'améliorer les plans d'exécution. `query hint` est une syntaxe ajoutée à la norme SQL qui indique au moteur de base de données comment exécuter la requête. Par exemple, un indicateur peut indiquer au moteur de suivre une analyse séquentielle et de remplacer tout plan sélectionné par l'optimiseur de requêtes.

Activer les indicateurs de requête T-SQL dans Babelfish

Actuellement, Babelfish ignore tous les indicateurs T-SQL par défaut. Pour appliquer des indicateurs T-SQL, exécutez la commande `sp_babelfish_configure` avec la valeur enable_pg_hint sur ON.

```
EXECUTE sp_babelfish_configure 'enable_pg_hint', 'on' [, 'server']
```

Vous pouvez rendre les paramètres permanents à l'échelle du cluster en incluant le mot-clé server. Pour configurer les paramètres de la session en cours uniquement, n'utilisez pas server.

Une fois `enable_pg_hint` sur ON, Babelfish applique les indicateurs T-SQL suivants.

- Indicateurs INDEX
- Indicateurs JOIN
- Indicateur FORCE ORDER
- Indicateur MAXDOP

Par exemple, la séquence de commandes suivante active `pg_hint_plan`.

```
1> CREATE TABLE t1 (a1 INT PRIMARY KEY, b1 INT);
```

```
2> CREATE TABLE t2 (a2 INT PRIMARY KEY, b2 INT);
3> GO
1> EXECUTE sp_babelfish_configure 'enable_pg_hint', 'on';
2> GO
1> SET BABELFISH_SHOWPLAN_ALL ON;
2> GO
1> SELECT * FROM t1 JOIN t2 ON t1.a1 = t2.a2; --NO HINTS (HASH JOIN)
2> GO
```

Aucun indicateur n'est appliqué à l'instruction SELECT. Le plan de requête sans indicateur est renvoyé.

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT * FROM t1 JOIN t2 ON t1.a1 = t2.a2
Hash Join (cost=60.85..99.39 rows=2260 width=16)
 Hash Cond: (t1.a1 = t2.a2)
 -> Seq Scan on t1 (cost=0.00..32.60 rows=2260 width=8)
 -> Hash (cost=32.60..32.60 rows=2260 width=8)
 -> Seq Scan on t2 (cost=0.00..32.60 rows=2260 width=8)
```

```
1> SELECT * FROM t1 INNER MERGE JOIN t2 ON t1.a1 = t2.a2;
2> GO
```

L'indicateur de requête est appliqué à l'instruction SELECT. La sortie suivante indique que le plan de requête avec un fusion JOIN est renvoyé.

```
QUERY PLAN


------------------------------------------------------------------------------
Query Text: SELECT/*+ MergeJoin(t1 t2) Leading(t1 t2)*/ * FROM t1 INNER JOIN t2 ON
 t1.a1 = t2.a2
Merge Join (cost=0.31..190.01 rows=2260 width=16)
 Merge Cond: (t1.a1 = t2.a2)
 -> Index Scan using t1_pkey on t1 (cost=0.15..78.06 rows=2260 width=8)
```

```
 -> Index Scan using t2_pkey on t2 (cost=0.15..78.06 rows=2260 width=8)
```

```
1> SET BABELFISH_SHOWPLAN_ALL OFF;
2> GO
```

Limites

Lorsque vous utilisez les indicateurs de requête, prenez en compte les limites suivantes :

- Si un plan de requête est mis en cache avant que `enable_pg_hint` ne soit activé, les indicateurs ne seront pas appliqués au cours de la même session. Ils seront appliqués lors de la nouvelle session.
- Si les noms de schéma sont explicitement donnés, les indicateurs ne peuvent pas être appliqués. Vous pouvez utiliser des alias de table pour contourner le problème.
- Un indicateur de requête ne peut pas être appliqué aux vues et aux sous-requêtes.
- Les indicateurs ne fonctionnent pas pour les instructions UPDATE/DELETE avec JOIN.
- Un indicateur INDEX pour un index ou une table inexistant est ignoré.
- L'indicateur FORCE ORDER ne fonctionne pas pour HASH JOIN et non HASH JOIN.

Utilisation des extensions Aurora PostgreSQL avec Babelfish

Aurora PostgreSQL fournit des extensions permettant de travailler avec d'autres services. AWS Il s’agit d’extensions facultatives qui prennent en charge divers cas d’utilisation, tels que l’utilisation d’Amazon S3 avec votre cluster de bases de données pour importer ou exporter des données.

- Pour importer des données d’un compartiment Amazon S3 vers votre cluster de bases de données Babelfish, vous devez configurer l’extension `aws_s3` Aurora PostgreSQL. Cette extension vous permet également d’exporter des données de votre cluster de bases de données Aurora PostgreSQL vers un compartiment Amazon S3.
- AWS Lambda est un service de calcul qui vous permet d'exécuter du code sans provisionner ni gérer de serveurs. Vous pouvez par exemple utiliser des fonctions Lambda pour traiter les notifications d’événements à partir de votre instance de base de données. Pour en savoir plus sur Lambda, consultez Qu’est-ce qu’ AWS Lambda ? dans le Guide du développeur AWS Lambda . Pour appeler des fonctions Lambda à partir de votre cluster de bases de données Babelfish, vous devez configurer l’extension `aws_lambda` Aurora PostgreSQL.

Pour configurer ces extensions pour votre cluster Babelfish, vous devez d'abord accorder à l'utilisateur Babelfish interne l'autorisation de charger les extensions. Après avoir accordé l'autorisation, vous pouvez charger les extensions Aurora PostgreSQL.

Activation des extensions Aurora PostgreSQL dans votre cluster de bases de données Babelfish

Avant de pouvoir charger les extensions `aws_s3` ou `aws_lambda`, vous accordez les privilèges nécessaires à votre cluster de bases de données Babelfish.

La procédure suivante utilise l'outil de ligne de commande `psql` PostgreSQL pour se connecter au cluster de bases de données. Pour plus d'informations, consultez Utilisation de psql pour se connecter au cluster de bases de données. Vous pouvez également utiliser pgAdmin. Pour en savoir plus, consultez Utilisation de pgAdmin pour se connecter au cluster de bases de données.

Cette procédure charge les extensions `aws_s3` et `aws_lambda` l'une après l'autre. Si vous ne comptez utiliser que l'une de ces extensions, vous n'avez pas besoin de les charger toutes les deux. L'extension `aws_commons` est requise par chacune, et elle est chargée par défaut, comme indiqué dans la sortie.

Pour configurer votre cluster de bases de données Babelfish avec des privilèges pour les extensions Aurora PostgreSQL

1. Connectez-vous à votre cluster de bases de données Babelfish. Utilisez le nom de l'utilisateur « principal » (-U) que vous avez spécifié lors de la création du cluster de bases de données Babelfish. La valeur par défaut (`postgres`) est illustrée dans les exemples.

   Pour Linux, macOS ou Unix :

   ```
   psql -h your-Babelfish.cluster.444455556666-us-east-1.rds.amazonaws.com \
   -U postgres \
   -d babelfish_db \
   -p 5432
   ```

   Pour Windows :

   ```
   psql -h your-Babelfish.cluster.444455556666-us-east-1.rds.amazonaws.com ^
   -U postgres ^
   -d babelfish_db ^
   -p 5432
   ```

La commande répond par une invite à saisir le mot de passe du nom d'utilisateur (-U).

```
Password:
```

Saisissez le mot de passe du nom d'utilisateur (-U) du cluster de bases de données. Lorsque vous serez connecté, vous obtiendrez une sortie similaire à ce qui suit.

```
psql (13.4)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
 compression: off)
Type "help" for help.

postgres=>
```

2. Accordez des privilèges à l'utilisateur Babelfish interne pour créer et charger des extensions.

```
babelfish_db=> GRANT rds_superuser TO master_dbo;
GRANT ROLE
```

3. Créez et chargez l'extension `aws_s3`. L'extension `aws_commons` est nécessaire et elle est installée automatiquement lorsque l'extension `aws_s3` est installée.

```
babelfish_db=> create extension aws_s3 cascade;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
```

4. Créez et chargez l'extension `aws_lambda`.

```
babelfish_db=> create extension aws_lambda cascade;
CREATE EXTENSION
babelfish_db=>
```

Utilisation de Babelfish avec Amazon S3

Si vous ne disposez pas d'un compartiment Amazon S3 à utiliser avec votre cluster de bases de données Babelfish, vous pouvez en créer un. Vous devez accorder l'accès pour tout compartiment Amazon S3 que vous souhaitez utiliser.

Avant d'essayer d'importer ou d'exporter des données à l'aide d'un compartiment Amazon S3, suivez les étapes uniques suivantes.

Pour configurer l'accès de votre instance de base de données Babelfish à votre compartiment Amazon S3

1. Créez un compartiment Amazon S3 pour votre instance Babelfish, si nécessaire. Pour ce faire, suivez les instructions de la section Création d'un compartiment dans le Guide de l'utilisateur Amazon Simple Storage Service.
2. Chargez les fichiers dans votre compartiment Amazon S3. Pour ce faire, suivez les étapes de la section Ajout d'un objet dans un compartiment dans le Guide de l'utilisateur Amazon Simple Storage Service.
3. Configurez les autorisations nécessaires :

   - Pour importer des données à partir d'Amazon S3, le cluster de bases de données Babelfish doit être autorisé à accéder au compartiment. Nous vous recommandons d'utiliser un rôle Gestion des identités et des accès AWS (IAM) et d'associer une politique IAM à ce rôle pour votre cluster. Pour ce faire, suivez les étapes de Utilisation d'un rôle IAM pour accéder à un compartiment Amazon S3.
   - Pour exporter des données à partir de votre cluster de bases de données Babelfish, votre cluster doit avoir accès au compartiment Amazon S3. Comme pour l'importation, nous recommandons d'utiliser une politique et un rôle IAM. Pour ce faire, suivez les étapes de Configuration de l'accès à un compartiment Amazon S3.

Vous pouvez désormais utiliser Amazon S3 avec l'extension `aws_s3` et votre cluster de bases de données Babelfish.

Pour importer des données à partir d'Amazon S3 vers Babelfish et pour exporter des données Babelfish vers Amazon S3

1. Utilisez l'extension `aws_s3` avec votre cluster de bases de données Babelfish.

   Dans ce cas, veillez à référencer les tables telles qu'elles existent dans le cadre de PostgreSQL. En d'autres termes, si vous souhaitez importer vers une table Babelfish nommée `[database].[schema].[tableA]`, référez-vous à cette table en tant que `database_schema_tableA` dans la fonction`aws_s3` :

- Pour obtenir un exemple d'utilisation de fonction `aws_s3` pour importer des données, consultez Importation de données d'Amazon S3 vers votre cluster de base de données Aurora PostgreSQL.
- Pour obtenir des exemples d'utilisation de fonctions `aws_s3` pour exporter des données, consultez Exportation de données de requête à l'aide de la fonction aws_s3.query_export_to_s3.

2. Veillez à référencer les tables Babelfish à l'aide de la dénomination PostgreSQL lorsque vous utilisez l'extension `aws_s3` et Amazon S3, comme indiqué dans la table suivante.

Table Babelfish	Table Aurora PostgreSQL
database.*schema*.*table*	*database*_*schema*_*table*

Pour en savoir plus sur l'utilisation d'Amazon S3 avec Aurora PostgreSQL, consultez les sections Importation de données Amazon S3 dans une d'un cluster de base de données Aurora PostgreSQL et Exportation de données à partir d'un cluster de bases de données Aurora PostgreSQL vers Amazon S3.

Utiliser Babelfish avec AWS Lambda

Après le chargement de l'extension `aws_lambda` dans votre cluster de bases de données Babelfish, mais avant l'appel des fonctions Lambda, accordez à Lambda l'accès à votre cluster de bases de données en suivant cette procédure.

Pour configurer l'accès à votre cluster de bases de données Babelfish afin qu'il fonctionne avec Lambda

Cette procédure utilise le AWS CLI pour créer la politique et le rôle IAM, et les associer au cluster de base de données Babelfish.

1. Créez une politique IAM qui autorise l'accès à Lambda à partir de votre cluster de bases de données Babelfish.

```
aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
 "Version": "2012-10-17",
    "Statement": [
        {
```

```
        "Sid": "AllowAccessToExampleFunction",
        "Effect": "Allow",
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
        }
    ]
}'
```

2. Créez un rôle IAM que la politique peut endosser lors de l'exécution.

```
aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. Attachez la stratégie au rôle.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. Attacher le rôle à votre cluster de bases de données Babelfish

```
aws rds add-role-to-db-cluster \
       --db-cluster-identifier my-cluster-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

Une fois ces tâches terminées, vous pouvez appeler vos fonctions Lambda. Pour plus d'informations et des exemples de configuration d'un cluster AWS Lambda AWS Lambda de base de données Aurora PostgreSQL avec, consultez. Étape 2 : configurer IAM pour votre instance de base de données Aurora PostgreSQL pour PostgreSQL et AWS Lambda

Pour appeler une fonction Lambda à partir de votre cluster de bases de données Babelfish

AWS Lambda prend en charge les fonctions écrites en Java, Node.js, Python, Ruby et dans d'autres langages. Si la fonction renvoie du texte lorsqu'elle est appelée, vous pouvez l'appeler à partir de votre cluster de bases de données Babelfish. L'exemple suivant est une fonction Python d'espace réservé qui renvoie un message de salutation.

```
lambda_function.py
import json
def lambda_handler(event, context):
    #TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
```

Actuellement, Babelfish ne prend pas en charge JSON. Si votre fonction renvoie JSON, utilisez un encapsuleur pour gérer le JSON. Par exemple, disons que la fonction `lambda_function.py` illustrée ci-dessus est stockée dans Lambda en tant que `my-function`.

1. Connectez-vous à votre cluster de bases de données Babelfish à l'aide du client `psql` (ou du client pgAdmin). Pour plus d'informations, consultez Utilisation de psql pour se connecter au cluster de bases de données.
2. Créez l'encapsuleur. Cet exemple utilise le langage procédural de PostgreSQL pour SQL, `PL/pgSQL`. Pour en savoir plus, consultez PL/pgSQL–SQL Procedural Language.

```
create or replace function master_dbo.lambda_wrapper()
returns text
language plpgsql
as
$$
declare
   r_status_code integer;
   r_payload text;
begin
   SELECT payload INTO r_payload
     FROM aws_lambda.invoke(  aws_commons.create_lambda_function_arn('my-function',
 'us-east-1')
                            ,'{"body": "Hello from Postgres!"}'::json );
   return r_payload ;
end;
```

```
$$;
```

La fonction peut désormais être exécutée à partir du port TDS Babelfish (1433) ou du port PostgreSQL (5433).

a. Pour appeler cette fonction depuis votre port PostgreSQL :

```
SELECT * from aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function', 'us-east-1'), '{"body": "Hello from Postgres!"}'::json );
```

La sortie est similaire à ce qui suit :

```
status_code |                        payload                         |
 executed_version | log_result
-------------+--------------------------------------------------------
+------------------+------------
         200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
         |
(1 row)
```

b. Pour appeler cette fonction à partir du port TDS, connectez-vous au port à l'aide du client de ligne de commande `sqlcmd` de SQL Server. Pour en savoir plus, consultez Utilisation d'un client SQL Server pour se connecter au cluster de bases de données. Une fois connecté, exécutez les opérations suivantes :

```
1> select lambda_wrapper();
2> go
```

La commande renvoie un résultat semblable à ce qui suit :

```
{"statusCode": 200, "body": "\"Hello from Lambda!\""}
```

Pour en savoir plus sur l'utilisation de Lambda avec Aurora PostgreSQL, consultez Invocation d'une AWS Lambda fonction depuis une instance de base de données Aurora PostgreSQL pour PostgreSQL. Pour plus d'informations sur l'utilisation des fonctions Lambda, consultez Mise en route avec Lambda dans le Guide du développeur AWS Lambda .

Utilisation de pg_stat_statements dans Babelfish

Babelfish pour Aurora PostgreSQL prend en charge l'extension `pg_stat_statements` à partir de la version 3.3.0. Pour plus d'informations, consultez pg_stat_statements.

Pour plus d'informations sur la version de cette extension prise en charge par Aurora PostgreSQL, consultez Extension versions.

Création de l'extension pg_stat_statements

Pour l'activer sur `pg_stat_statements`, vous devez activer le calcul de l'identifiant de la requête. Il s'active automatiquement si `compute_query_id` est défini sur on ou `auto` dans le groupe de paramètres. La valeur par défaut du paramètre `compute_query_id` est `auto`. Vous devez également créer cette extension pour activer cette fonctionnalité. Utilisez la commande suivante pour installer l'extension à partir du point de terminaison T-SQL :

```
1>EXEC sp_execute_postgresql 'CREATE EXTENSION pg_stat_statements WITH SCHEMA sys';
```

Vous pouvez accéder aux statistiques des requêtes à l'aide de la requête suivante :

```
postgres=>select * from pg_stat_statements;
```

> **Note**
>
> Pendant l'installation, si vous ne fournissez pas le nom du schéma de l'extension, celle-ci sera créée par défaut dans le schéma public. Pour y accéder, vous devez utiliser des crochets avec un qualificateur de schéma, comme indiqué ci-dessous :

```
postgres=>select * from [public].pg_stat_statements;
```

Vous pouvez également créer l'extension à partir du point de terminaison PSQL.

Autorisation de l’extension

Par défaut, vous pouvez consulter les statistiques des requêtes effectuées dans votre base de données T-SQL sans avoir besoin d’autorisation.

Pour accéder aux statistiques des requêtes créées par d’autres utilisateurs, vous devez disposer du rôle `pg_read_all_stats` PostgreSQL. Suivez les étapes mentionnées ci-dessous pour créer la commande GRANT pg_read_all_stats.

1. Dans T-SQL, utilisez la requête suivante qui renvoie le nom du rôle PG interne.

   ```
   SELECT rolname FROM pg_roles WHERE oid = USER_ID();
   ```

2. Connectez-vous à la base de données Babelfish pour Aurora PostgreSQL avec le privilège rds_superuser et utilisez la commande suivante :

   ```
   GRANT pg_read_all_stats TO <rolname_from_above_query>
   ```

Exemple

À partir du point de terminaison T-SQL :

```
1>SELECT rolname FROM pg_roles WHERE oid = USER_ID();
2>go
```

```

rolname
-------
master_dbo
(1 rows affected)
```

À partir du point de terminaison PSQL :

```
babelfish_db=# grant pg_read_all_stats to master_dbo;
```

```
GRANT ROLE
```

Vous pouvez accéder aux statistiques des requêtes à l’aide de la vue pg_stat_statements :

```
1>create table t1(cola int);
2>go
1>insert into t1 values (1),(2),(3);
2>go
```

```
(3 rows affected)
```

```
1>select userid, dbid, queryid, query from pg_stat_statements;
2>go
```

```
userid dbid queryid             query
------ ---- -------             -----
37503 34582 6487973085327558478 select * from t1
37503 34582 6284378402749466286 SET QUOTED_IDENTIFIER OFF
37503 34582 2864302298511657420 insert into t1 values ($1),($2),($3)
10    34582 NULL                <insufficient privilege>
37503 34582 5615368793313871642 SET TEXTSIZE 4096
37503 34582 639400815330803392  create table t1(cola int)
(6 rows affected)
```

Réinitialisation des statistiques des requêtes

Vous pouvez utiliser `pg_stat_statements_reset()` pour réinitialiser les statistiques recueillies jusqu'à présent par pg_stat_statements. Pour plus d'informations, consultez pg_stat_statements. Elle est actuellement prise en charge uniquement via le point de terminaison PSQL. Connectez-vous à Babelfish pour Aurora PostgreSQL avec le privilège `rds_superuser` et utilisez la commande suivante :

```
SELECT pg_stat_statements_reset();
```

Limitations

- Actuellement, `pg_stat_statements()` n'est pas pris en charge via le point de terminaison T-SQL. La vue `pg_stat_statements` est la méthode recommandée pour recueillir les statistiques.

- Certaines requêtes peuvent être réécrites par l'analyseur T-SQL implémenté par le moteur Aurora PostgreSQL. La vue `pg_stat_statements` affichera la requête réécrite, et non la requête d'origine.

  Exemple

  ```
  select next value for [dbo].[newCounter];
  ```

  La requête ci-dessus est réécrite comme suit dans la vue pg_stat_statements.

  ```
  select nextval($1);
  ```

- Selon le flux d'exécution des instructions, certaines requêtes peuvent ne pas être suivies par pg_stat_statements et ne seront pas visibles dans la vue. Elle comprend les instructions suivantes : `use dbname`, `goto`, `print`, `raise error`, `set`, `throw`, `declare cursor`.
- Pour les instructions CREATE LOGIN et ALTER LOGIN, query et queryid ne s'afficheront pas. Elle indiquera que les privilèges sont insuffisants.
- La vue `pg_stat_statements` contient toujours les deux entrées ci-dessous, car elles sont exécutées en interne par le client `sqlcmd`.
  - SET QUOTED_IDENTIFIER OFF
  - SET TEXTSIZE 4096

Utilisation de pgvector dans Babelfish

pgvector est une extension open source qui vous permet de rechercher des données similaires directement dans votre base de données Postgres. Babelfish prend désormais en charge cette extension à partir des versions 15.6 et 16.2. Pour plus d'informations, consultez la documentation open source de pgvector.

Prérequis

Pour activer la fonctionnalité pgvector, installez l'extension dans le schéma sys en utilisant l'une des méthodes suivantes :

- Exécutez la commande suivante dans le client sqlcmd :

  ```
  exec sys.sp_execute_postgresql 'CREATE EXTENSION vector WITH SCHEMA sys';
  ```

- Connectez-vous à `babelfish_db` et exécutez la commande suivante dans le client psql :

```
CREATE EXTENSION vector WITH SCHEMA sys;
```

> **Note**
>
> Après avoir installé l'extension pgvector, le type de données vectorielles ne sera disponible que dans les nouvelles connexions à la base de données que vous établirez. Les connexions existantes ne reconnaîtront pas ce nouveau type de données.

Fonctionnalités prises en charge

Babelfish étend les fonctionnalités T-SQL pour prendre en charge les éléments suivants :

- Stockage

  Babelfish prend désormais en charge la syntaxe compatible avec les types de données vectoriels, améliorant ainsi sa compatibilité T-SQL. Pour en savoir plus sur le stockage de données avec pgvector, consultez Stockage.

- Interrogation

  Babelfish étend la prise en charge des expressions T-SQL pour inclure les opérateurs de similarité vectorielle. Cependant, pour toutes les autres requêtes, la syntaxe T-SQL standard est toujours requise.

  > **Note**
  >
  > T-SQL ne prend pas en charge le type Array, et les pilotes de base de données ne disposent d'aucune interface pour les gérer. Pour contourner le problème, Babelfish utilise des chaînes de texte (varchar/nvarchar) pour stocker les données vectorielles. Par exemple, lorsque vous demandez une valeur vectorielle [1,2,3], Babelfish renvoie une chaîne '[1,2,3]' comme réponse. Vous pouvez analyser et diviser cette chaîne au niveau de l'application selon vos besoins.

  Pour en savoir plus sur l'interrogation des données avec pgvector, consultez Interrogation.

- Indexation

  La fonction `Create Index` de T-SQL prend désormais en charge la syntaxe `USING INDEX_METHOD`. Vous pouvez désormais définir l'opérateur de recherche de similarité à utiliser sur une colonne spécifique lors de la création d'un index.

  La grammaire est également étendue pour prendre en charge les opérations de similarité vectorielle sur la colonne requise (vérifier la grammaire column_name_list_with_order_for_vector).

```
CREATE [UNIQUE] [clustered] [COLUMNSTORE] INDEX <index_name> ON <table_name> [USING
 vector_index_method] (<column_name_list_with_order_for_vector>)
Where column_name_list_with_order_for_vector is:
    <column_name> [ASC | DESC] [VECTOR_COSINE_OPS | VECTOR_IP_OPS | VECTOR_L2_OPS]
 (COMMA simple_column_name [ASC | DESC] [VECTOR_COSINE_OPS | VECTOR_IP_OPS |
 VECTOR_L2_OPS])
```

  Pour en savoir plus sur l'indexation des données avec pgvector, consultez Indexation.

- Performances
  - Utilisez `SET BABELFISH_STATISTICS PROFILE ON` pour déboguer les plans de requête du point de terminaison T-SQL.
  - Augmentez `max_parallel_workers_get_gather` à l'aide de la fonction `set_config` prise en charge dans T-SQL.
  - Utilisez `IVFFlat` pour les recherches approximatives. Pour de plus amples informations, veuillez consulter IVFFlat.

  Pour améliorer les performances avec pgvector, consultez Performances.

Limitations

- Babelfish ne prend pas en charge la recherche en texte intégral pour la recherche hybride. Pour plus d'informations, consultez Recherche hybride.
- Babelfish ne prend actuellement pas en charge la fonctionnalité de réindexation. Cependant, vous pouvez toujours utiliser le point de terminaison PostgreSQL pour procéder à la réindexation. Pour plus d'informations, consultez Exécution de l'opération VACUUM.

Utilisation du machine learning Amazon Aurora avec Babelfish

Vous pouvez étendre les fonctionnalités de votre cluster de bases de données Babelfish pour Aurora PostgreSQL en l'intégrant au machine learning Amazon Aurora. Cette intégration fluide vous donne accès à une gamme de services puissants tels qu'Amazon Comprehend, Amazon SageMaker AI ou Amazon Bedrock, chacun étant conçu pour répondre à des besoins d'apprentissage automatique distincts.

En tant qu'utilisateur de Babelfish, vous pouvez utiliser les connaissances existantes en matière de syntaxe et de sémantique T-SQL lorsque vous travaillez avec le machine learning Aurora. Suivez les instructions fournies dans la AWS documentation d'Aurora PostgreSQL. Pour de plus amples informations, veuillez consulter Utilisation du machine learning Amazon Aurora avec Aurora PostgreSQL.

Prérequis

- Avant d'essayer de configurer votre cluster de bases de données Babelfish pour Aurora PostgreSQL pour utiliser le machine learning Aurora, vous devez comprendre les exigences et prérequis suivants. Pour plus d'informations, consultez Exigences pour l'utilisation du machine learning Aurora avec Aurora PostgreSQL.
- Assurez-vous d'installer l'extension `aws_ml` à l'aide du point de terminaison Postgres ou de la procédure de stockage `sp_execute_postgresql`.

```
exec sys.sp_execute_postgresql 'Create Extension aws_ml'
```

> **Note**
>
> Actuellement, Babelfish ne prend pas en charge les opérations en cascade avec `sp_execute_postgresql` dans Babelfish. Comme `aws_ml` repose sur `aws_commons`, vous devrez l'installer séparément à l'aide du point de terminaison Postgres.

```
create extension aws_common;
```

Gestion de la syntaxe et de la sémantique T-SQL avec des fonctions **`aws_ml`**

Les exemples suivants expliquent comment la syntaxe et la sémantique T-SQL sont appliquées aux services Amazon ML :

Example : aws_bedrock.invoke_model — requête simple utilisant les fonctions Amazon Bedrock

```
aws_bedrock.invoke_model(
   model_id      varchar,
   content_type  text,
   accept_type   text,
   model_input   text)
Returns Varchar(MAX)
```

L'exemple suivant montre comment invoquer un modèle Anthropic Claude 2 pour Bedrock en utilisant invoke_model.

```
SELECT aws_bedrock.invoke_model (
    'anthropic.claude-v2', -- model_id
    'application/json', -- content_type
    'application/json', -- accept_type
    '{"prompt": "\n\nHuman:
    You are a helpful assistant that answers questions directly
    and only using the information provided in the context below.
    \nDescribe the answerin detail.\n\nContext: %s \n\nQuestion:
    %s \n\nAssistant:","max_tokens_to_sample":4096,"temperature"
    :0.5,"top_k":250,"top_p":0.5,"stop_sequences":[]}' -- model_input
);
```

Example : aws_comprehend.detect_sentiment — requête simple utilisant les fonctions Amazon Comprehend

```
aws_comprehend.detect_sentiment(
   input_text varchar,
   language_code varchar,
   max_rows_per_batch int)
Returns table (sentiment varchar, confidence real)
```

L’exemple suivant montre comment invoquer le service Amazon Comprehend.

```
select sentiment from aws_comprehend.detect_sentiment('This is great', 'en');
```

Example : aws_sagemaker.invoke_endpoint — Une requête simple utilisant les fonctions Amazon SageMaker

```
aws_sagemaker.invoke_endpoint(
  endpoint_name varchar,
  max_rows_per_batch int,
  VARIADIC model_input "any") -- Babelfish inherits PG's variadic parameter type
Rerurns Varchar(MAX)
```

Comme model_input est marqué comme VARIADIC et comme il est de type « any », les utilisateurs peuvent transmettre une liste de n’importe quelle longueur et de n’importe quel type de données à la fonction qui servira d’entrée au modèle. L'exemple suivant montre comment appeler le SageMaker service Amazon.

```
SELECT CAST (aws_sagemaker.invoke_endpoint(
    'sagemaker_model_endpoint_name',
    NULL,
    arg1, arg2 -- model inputs are separate arguments )
AS INT) -- cast the output to INT
```

Pour plus d’informations sur l’utilisation du machine learning Aurora avec Aurora PostgreSQL, consultez Utilisation du machine learning Amazon Aurora avec Aurora PostgreSQL.

Limitations

- Bien que Babelfish n’autorise pas la création de tableaux, il peut tout de même gérer les données représentant des tableaux. Lorsque vous utilisez des fonctions comme `aws_bedrock.invoke_model_get_embeddings` qui renvoient des tableaux, les résultats sont fournis sous forme de chaîne contenant les éléments du tableau.

Babelfish prend en charge les serveurs liés

Babelfish for Aurora PostgreSQL prend en charge les serveurs liés en utilisant l'extension `tds_fdw` PostgreSQL dans la version 3.1.0. Pour pouvoir utiliser des serveurs liés, vous devez installer l'extension `tds_fdw`. Pour plus d'informations sur l'extension `tds_fdw`, consultez Utilisation des encapsuleurs de données externes pris en charge pour Amazon Aurora PostgreSQL.

Installation de l'extension **`tds_fdw`**

Vous pouvez installer l'extension `tds_fdw` à l'aide des méthodes suivantes.

Utilisation de CREATE EXTENSION à partir du point de terminaison PostgreSQL

1. Connectez-vous à votre instance de base de données PostgreSQL sur la base de données Babelfish sur le port PostgreSQL. Utilisez un compte qui possède le rôle rds_superuser.

   ```
   psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=test --dbname=babelfish_db --password
   ```

2. Installez l'extension `tds_fdw`. Il s'agit d'une procédure d'installation unique. Vous n'avez pas besoin de réinstaller lorsque le cluster de base de données redémarre.

   ```
   babelfish_db=> CREATE EXTENSION tds_fdw;
   CREATE EXTENSION
   ```

Appel d'une procédure **`sp_execute_postgresql`** stockée à partir du point de terminaison TDS

Babelfish prend en charge l'installation de l'extension `tds_fdw` en appelant la procédure `sp_execute_postgresql` de la version 3.3.0. Vous pouvez exécuter des instructions PostgreSQL à partir du point de terminaison T-SQL sans quitter le port T-SQL. Pour plus d'informations, consultez Utilisation des procédures Babelfish pour Aurora PostgreSQL.

1. Connectez-vous à votre instance de base de données PostgreSQL dans la base de données Babelfish sur le port T-SQL.

   ```
   sqlcmd -S your-DB-instance.aws-region.rds.amazonaws.com -U test -P password
   ```

2. Installez l'extension `tds_fdw`.

```
1>EXEC sp_execute_postgresql N'CREATE EXTENSION tds_fdw';
2>go
```

Fonctionnalités prises en charge

Babelfish prend en charge l'ajout de points de terminaison distants RDS for SQL Server ou Babelfish for Aurora PostgreSQL en tant que serveur lié. Vous pouvez également ajouter d'autres instances SQL Server distantes en tant que serveurs liés. Ensuite, utilisez `OPENQUERY()` pour récupérer des données à partir de ces serveurs liés. Depuis Babelfish version 3.2.0, les noms en quatre parties sont également pris en charge.

Les procédures stockées et les vues de catalogue suivantes sont prises en charge afin d'utiliser les serveurs liés.

Procédures stockées

- sp_addlinkedserver : Babelfish ne prend pas en charge le paramètre `@provstr`.
- sp_addlinkedsrvlogin
  - Vous devez fournir un nom d'utilisateur et un mot de passe distants explicites pour vous connecter à la source de données distante. Vous ne pouvez pas vous connecter avec les informations d'identification personnelles de l'utilisateur. Babelfish prend uniquement en charge `@useself = false`.
  - Babelfish ne prend pas en charge le paramètre `@locallogin`, car la configuration de l'accès au serveur distant spécifique à la connexion locale n'est pas prise en charge.
- sp_linkedservers
- sp_helplinkedsrvlogin
- sp_dropserver
- sp_droplinkedsrvlogin : Babelfish ne prend pas en charge le paramètre `@locallogin`, car la configuration de l'accès au serveur distant spécifique à la connexion locale n'est pas prise en charge.
- sp_serveroption : Babelfish prend en charge les options de serveur suivantes :
  - query timeout (à partir de Babelfish version 3.2.0)
  - connect timeout (à partir de Babelfish version 3.3.0)
- sp_testlinkedserver (à partir de Babelfish version 3.3.0)

- sp_enum_oledb_providers (à partir de Babelfish version 3.3.0)

Affichages du catalogue

- sys.servers
- sys.linked_logins

Utilisation du chiffrement en transit pour la connexion

La connexion depuis le serveur source Babelfish for Aurora PostgreSQL vers le serveur cible distant utilise le chiffrement en transit (TLS/SSL) selon la configuration de la base de données du serveur distant. Si le serveur distant n'est pas configuré pour le chiffrement, le serveur Babelfish qui émet la requête à la base de données distante revient au mode non chiffré.

Pour appliquer le chiffrement des connexions

- Si le serveur lié cible est une instance RDS for SQL Server, définissez `rds.force_ssl = on` pour l'instance SQL Server cible. Pour plus d'informations sur la configuration SSL/TLS pour RDS for SQL Server, consultez Utilisation de SSL avec une instance DB Microsoft SQL Server.
- Si le serveur lié cible est un cluster Babelfish for Aurora PostgreSQL, définissez `babelfishpg_tds.tds_ssl_encrypt = on` et `ssl = on` pour le serveur cible. Pour plus d'informations sur SSL/TLS, consultez Paramètres SSL Babelfish et connexions client.

Ajout de Babelfish en tant que serveur lié depuis SQL Server

Babelfish for Aurora PostgreSQL peut être ajouté en tant que serveur lié à partir d'un serveur SQL Server. Sur une base de données SQL Server, vous pouvez ajouter Babelfish en tant que serveur lié à l'aide du fournisseur Microsoft OLE DB pour ODBC : MSDASQL.

Il existe deux manières de configurer Babelfish en tant que serveur lié à partir de SQL Server à l'aide du fournisseur MSDASQL :

- Fourniture de la chaîne de connexion ODBC en tant que chaîne du fournisseur.
- Spécifiez le nom de la source de données (DSN) système de la source de données ODBC lors de l'ajout du serveur lié.

Limites

- OPENQUERY() fonctionne uniquement pour SELECT et ne fonctionne pas pour DML.
- Les noms d'objets en quatre parties ne fonctionnent que pour la lecture et ne fonctionnent pas pour modifier la table distante. UPDATE peut référencer une table distante dans la clause FROM sans la modifier.
- L'exécution de procédures stockées sur les serveurs liés Babelfish n'est pas prise en charge.
- La mise à niveau de la version majeure de Babelfish peut ne pas fonctionner si des objets dépendent de `OPENQUERY()` ou d'objets référencés par des noms en quatre parties. Vous devez vous assurer que tous les objets faisant référence à `OPENQUERY()` ou à des noms en quatre parties sont supprimés avant une mise à niveau de la version majeure.
- Les types de données suivants ne fonctionnent pas comme prévu par rapport au serveur Babelfish distant : `nvarchar(max)`, `varchar(max)`, `varbinary(max)`, `binary(max)` et `time`. Nous vous recommandons d'utiliser la fonction CAST pour les convertir dans les types de données pris en charge.

exemple

Dans l'exemple suivant, une instance Babelfish for Aurora PostgreSQL se connecte à une instance de RDS for SQL Server dans le cloud.

```
EXEC master.dbo.sp_addlinkedserver @server=N'rds_sqlserver', @srvproduct=N'',
 @provider=N'SQLNCLI', @datasrc=N'myserver.CB2XKFSFFMY7.US-WEST-2.RDS.AMAZONAWS.COM';
EXEC master.dbo.sp_addlinkedsrvlogin
 @rmtsrvname=N'rds_sqlserver',@useself=N'False',@locallogin=NULL,@rmtuser=N'username',@rmtpassw
```

Lorsque le serveur lié est en place, vous pouvez utiliser T-SQL OPENQUERY() ou une dénomination standard en quatre parties pour référencer une table, une vue ou d'autres objets pris en charge sur le serveur distant :

```
SELECT * FROM OPENQUERY(rds_sqlserver, 'SELECT * FROM TestDB.dbo.t1');
SELECT * FROM rds_sqlserver.TestDB.dbo.t1;
```

Pour supprimer le serveur lié et toutes les connexions associées :

```
EXEC master.dbo.sp_dropserver @server=N'rds_sqlserver', @droplogins=N'droplogins';
```

Résolution des problèmes

Vous pouvez utiliser le même groupe de sécurité pour les serveurs source et distant afin de leur permettre de communiquer entre eux. Le groupe de sécurité doit autoriser uniquement le trafic entrant sur le port TDS (1433 par défaut) et l'adresse IP source du groupe de sécurité peut être définie comme identifiant du groupe de sécurité lui-même. Pour plus d'informations sur la définition des règles de connexion à une instance à partir d'une autre instance dotée du même groupe de sécurité, consultez Règles pour la connexion à des instances à partir d'une instance avec le même groupe de sécurité.

Si l'accès n'est pas configuré correctement, un message d'erreur similaire à l'exemple suivant s'affiche lorsque vous essayez d'interroger le serveur distant.

```
TDS client library error: DB #: 20009, DB Msg: Unable to connect: server is unavailable
 or does not exist (mssql2019.aws-region.rds.amazonaws.com), OS #: 110, OS Msg:
 Connection timed out, Level: 9
```

Utilisation de la recherche en texte intégral dans Babelfish

À partir de la version 4.0.0, Babelfish assure une prise en charge limitée de la recherche en texte intégral (FTS). FTS est une fonctionnalité puissante des bases de données relationnelles qui permet une recherche et une indexation efficaces des données contenant beaucoup de texte. Elle vous permet d'effectuer des recherches de texte complexes et d'extraire rapidement des résultats pertinents. FTS est particulièrement utile pour les applications qui traitent de gros volumes de données textuelles, telles que les systèmes de gestion de contenu, les plateformes de commerce électronique et les archives de documents.

Comprendre les fonctionnalités de recherche en texte intégral prises en charge par Babelfish

Babelfish prend en charge les fonctionnalités de recherche en texte intégral suivantes :

- Clause CONTAINS :
  - Prise en charge de base de la clause CONTAINS.

```
CONTAINS (
     {
        column_name
     }
     , '<contains_search_condition>'
)
```

> **Note**
>
> À l'heure actuelle, seule l'anglais est pris en charge.

  - Gestion et traduction complètes des chaînes de recherche `simple_term`.
- Clause `FULLTEXT INDEX` :
  - Prend uniquement en charge l'instruction `CREATE FULLTEXT INDEX ON table_name(column_name [...n]) KEY INDEX index_name`.
  - Prend en charge l'instruction `DROP FULLTEXT INDEX` complète.

> **Note**
>
> Pour réindexer l'index du texte intégral, vous devez le supprimer et en créer un autre sur la même colonne.

- Caractères spéciaux dans les conditions de recherche :
  - Babelfish garantit que les occurrences uniques de caractères spéciaux dans les chaînes de recherche sont gérées efficacement.

> **Note**
>
> Bien que Babelfish identifie désormais les caractères spéciaux dans la chaîne de recherche, il est essentiel de reconnaître que les résultats obtenus peuvent varier par rapport à ceux obtenus avec T-SQL.

- Alias de table dans column_name :
  - Grâce à la prise en charge des alias de table, les utilisateurs peuvent créer des requêtes SQL plus concises et lisibles pour la recherche en texte intégral.

Limites de la recherche en texte intégral Babelfish

- Actuellement, les options suivantes ne sont pas prises en charge dans Babelfish pour la clause `CONTAINS`.
  - Les caractères spéciaux et les langues autres que l'anglais ne sont pas pris en charge. Vous recevrez le message d'erreur générique pour les caractères et les langues non pris en charge

    ```
    Full-text search conditions with special characters or languages other than English
     are not currently supported in Babelfish
    ```

  - Plusieurs colonnes comme `column_list`
  - Attribut PROPERTY
  - `prefix_term`, `generation_term`, `generic_proximity_term`, `custom_proximity_term` et `weighted_term`
  - Les opérateurs booléens ne sont pas pris en charge et vous recevez le message d'erreur suivant lorsqu'ils sont utilisés :

    ```
    boolean operators not supported
    ```

  - Les noms d'identifiants comprenant des points ne sont pas pris en charge.
- Actuellement, les options suivantes ne sont pas prises en charge dans Babelfish pour la clause `CREATE FULLTEXT INDEX`.
  - [TYPE COLUMN type_column_name]
  - [LANGUAGE language_term]
  - [STATISTICAL_SEMANTICS]
  - options de groupe de fichiers de catalogue
  - avec options
- La création d'un catalogue de texte intégral n'est pas prise en charge. La création d'un index de texte intégral ne nécessite pas de catalogue de texte intégral.
- `CREATE FULLTEXT INDEX` ne prend pas en charge les noms d'identifiant contenant des points.
- Babelfish ne prend actuellement pas en charge les caractères spéciaux consécutifs dans les chaînes de recherche. Vous recevrez le message d'erreur suivant lorsque vous en utilisez :

```
Consecutive special characters in the full-text search condition are not currently
 supported in Babelfish
```

Babelfish prend en charge les types de données géospatiales

À partir des versions 3.5.0 et 4.1.0, Babelfish prend en charge les deux types de données spatiales suivants :

- Type de données géométriques : ce type de données est destiné au stockage de données planaires ou euclidiennes (monde en deux dimensions).
- Type de données géographiques : ce type de données est destiné au stockage de données ellipsoïdales ou terrestres, telles que les coordonnées GPS de latitude et de longitude.

Ces types de données permettent le stockage et la manipulation de données spatiales, mais avec des limites.

Comprendre les types de données géospatiales dans Babelfish

- Les types de données géospatiales sont pris en charge dans divers objets de base de données tels que les vues, les procédures et les tables.
- Supporte le type de données ponctuelles pour stocker les données de localisation sous forme de points définis par la latitude, la longitude et un identifiant de système de référence spatiale (SRID) valide. Un point peut contenir des valeurs Z (altitude), M (mesure) et peut être vide.
- Supporte le type de données linestring (à partir de la version 5.4.0) défini par une séquence de points et les segments de ligne qui les relient et par un identifiant de système de référence spatiale (SRID) valide. Une chaîne linéaire peut contenir des points avec des valeurs Z (altitude), M (mesure) et peut être vide.
- Les applications qui se connectent à Babelfish via des pilotes tels que JDBC, ODBC, DOTNET et PYTHON peuvent utiliser cette fonctionnalité géospatiale.

Fonctions des types de données géométriques prises en charge dans Babelfish

- STGeomFromText (***`geometry_tagged_text`***, SRID) — Crée une instance de géométrie à l'aide d'une représentation en texte connu (WKT).

- STPointFromText (***point_tagged_text***, SRID) — Crée une instance de point à l'aide de la représentation WKT.
- Point (X, Y, SRID) : crée une instance de point en utilisant les valeurs flottantes des coordonnées x et y.
- <geometry_instance>. STAsText () — Extrait la représentation WKT de l'instance de géométrie.
- <geometry_instance>. STAsBinary () — Extrait la représentation WKB de l'instance de géométrie.
- <geometry_instance>. STArea () — Calcule la surface totale de l'instance de géométrie.
- <geometry_instance>. STSrid () — Extrait l'identifiant de référence spatiale (SRID) de l'instance de géométrie.
- <geometry_instance>. STDimension () — Récupère la dimension spatiale de l'instance de géométrie.
- <geometry_instance>. STIsEmpty () — Vérifie si l'instance de géométrie est vide.
- <geometry_instance>. STIsFermé () — Vérifie si l'instance de géométrie est fermée.
- <geometry_instance>. STIsValid () — Vérifie si l'instance de géométrie est valide.
- <geometry_instance>. STDistance (other_geometry) — Calcule la distance entre deux instances de géométrie.
- <geometry_instance>. STEquals (other_geometry) — Vérifie si l'instance de géométrie représente le même ensemble de points qu'une autre instance de géométrie.
- <geometry_instance>. STContains (other_geometry) — Vérifie si l'instance de géométrie contient l'instance other_geometry.
- <geometry_instance>. STDisjoint (other_geometry) — Vérifie si deux instances de géométrie n'ont aucun point commun.
- <geometry_instance>. STIntersects (other_geometry) — Vérifie si deux instances de géométrie se croisent spatialement.
- <geometry_instance>.STX : extrait la coordonnée X (longitude) de l’instance de géométrie.
- <geometry_instance>.STY : extrait la coordonnée Y (latitude) de l’instance de géométrie.

À partir des versions 4.7.0 et 5.3.0, Babelfish inclut le support des fonctions de données spatiales suivantes :

- .M <geometry_instance>— Extrait la coordonnée M de l'instance de géométrie.
- .Z <geometry_instance>— Extrait la coordonnée Z de l'instance de géométrie.
- .hasM <geometry_instance>— Vérifie si l'instance de géométrie possède au moins une valeur M.

- .hasZ <geometry_instance>— Vérifie si l'instance de géométrie possède au moins une valeur Z.

À partir des versions 5.4.0, Babelfish inclut le support de la fonction de données spatiales suivante :

- STLineFromText (***linestring_tagged_text***, SRID) — Crée une instance de chaîne de lignes à l'aide d'une représentation WKT.

Fonctions de type de données géographiques prises en charge dans Babelfish

- STGeomFromText (***geography_tagged_text***, SRID) — Crée une instance de géographie à l'aide d'une représentation WKT.
- STPointFromText (***point_tagged_text***, SRID) — Crée une instance de point à l'aide de la représentation WKT.
- Point (Lat, Long, SRID) : crée une instance de point en utilisant des valeurs flottantes de latitude et de longitude.
- <geography_instance>. STAsText () — Extrait la représentation WKT de l'instance de géographie.
- <geography_instance>. STAsBinary () — Extrait la représentation WKB de l'instance de géographie.
- <geography_instance>. STArea () — Calcule la surface totale de l'instance géographique.
- <geography_instance>. STSrid () — Extrait l'identifiant de référence spatiale (SRID) de l'instance de géographie.
- <geography_instance>. STDimension () — Récupère la dimension spatiale de l'instance de géographie.
- <geography_instance>. STIsEmpty () — Vérifie si l'instance de géographie est vide.
- <geography_instance>. STIsFermé () — Vérifie si l'instance de géographie est fermée.
- <geography_instance>. STIsValid () — Vérifie si l'instance de géographie est valide.
- <geography_instance>. STDistance (other_geography) — Calcule la distance entre deux instances géographiques.
- <geography_instance>. STEquals (other_geography) — Vérifie si l'instance de géographie représente le même ensemble de points qu'une autre instance de géographie.
- <geography_instance>. STContains (other_geography) — Vérifie si l'instance de géographie contient l'instance other_geography.
- <geography_instance>. STDisjoint (other_geography) — Vérifie si deux instances géographiques n'ont aucun point commun.

- <geography_instance>. STIntersects (other_geography) — Vérifie si deux instances géographiques se croisent spatialement.
- <geography_instance>.Lat : extrait la valeur de latitude pour l'instance de géographie.
- <geography_instance>.Long : extrait la valeur de longitude pour l'instance de géographie.

À partir des versions 4.7.0 et 5.3.0, Babelfish inclut le support des fonctions de données spatiales suivantes :

- .M <geography_instance>— Extrait la coordonnée M de l'instance géographique.
- .Z <geography_instance>— Extrait la coordonnée Z de l'instance géographique.
- .hasM <geography_instance>— Vérifie si l'instance de géographie possède au moins une valeur M.
- .hasZ <geography_instance>— Vérifie si l'instance de géographie possède au moins une valeur Z.

À partir des versions 5.4.0, Babelfish inclut le support de la fonction de données spatiales suivante :

- STLineFromText (***linestring_tagged_text***, SRID) — Crée une instance de chaîne de lignes à l'aide d'une représentation WKT.

Limites de Babelfish pour les types de données géospatiales

- Les types de géométrie et de géographie autres que les instances de points et de chaînes de lignes ne sont actuellement pas pris en charge :
  - CircularString
  - CompoundCurve
  - Polygone
  - CurvePolygon
  - MultiPoint
  - MultiLineString
  - MultiPolygon
  - GeometryCollection
- Actuellement, l'indexation spatiale n'est pas prise en charge pour les types de données géospatiales.

- Seules les fonctions répertoriées sont actuellement prises en charge pour ces types de données. Pour plus d'informations, consultez Fonctions des types de données géométriques prises en charge dans Babelfish et Fonctions de type de données géographiques prises en charge dans Babelfish.
- STDistance la sortie de la fonction pour les données géographiques peut présenter des variations de précision mineures par rapport à T-SQL. Cela est dû à l'implémentation sous-jacente de PostGIS. Pour plus d'informations, consultez ST_Distance.
- STIsLa sortie de fonction valide pour les données de géométrie et de géographie peut présenter certains écarts par rapport à T-SQL. De ce fait, les fonctions - STDistance,, STContains, STInstersects, STDisjoint, STDimension STArea, STEquals peuvent également différer de T-SQL dans certains cas (renvoie le résultat au lieu de générer une erreur). Cela est dû à l'implémentation sous-jacente de PostGIS. Pour plus d'informations, consultez ST_ IsValid.
- Pour des performances optimales, utilisez des types de données géospatiales intégrés, sans créer de couches d'abstraction supplémentaires dans Babelfish.
- Dans Babelfish, les noms des fonctions géospatiales sont utilisés comme mots-clés, et les opérations spatiales ne sont effectuées que si elles sont utilisées de la manière prévue.

Comprendre le partitionnement dans Babelfish

À partir de la version 4.3.0, Babelfish inclut le partitionnement des tables et des index avec une prise en charge limitée. Les sections suivantes fournissent des détails sur la création de fonctions de partition, la définition de schémas de partition et l'implémentation de tables et d'index partitionnés dans Babelfish.

Rubriques

- Présentation du partitionnement dans Babelfish
- Limites et solutions de contournement

Présentation du partitionnement dans Babelfish

- Fonctions de partition :
  - `CREATE PARTITION FUNCTION:` définit le mode de partitionnement d'une table ou d'un index en spécifiant le type de colonne de partitionnement et la plage de valeurs de chaque partition.
  - `DROP PARTITION FUNCTION:` supprime une fonction de partition existante.
- Schémas de partition :

  - `CREATE PARTITION SCHEME:` définit le mappage entre les partitions et les groupes de fichiers.

    > **Note**
    >
    > Dans Babelfish, les groupes de fichiers sont traités comme des objets factices et ne représentent pas des emplacements de stockage physiques.

  - `DROP PARTITION SCHEME:` supprime un schéma de partition existant.
- Fonction système :
  - `$PARTITION:` cette fonction système renvoie le numéro de partition auquel une valeur spécifiée dans une colonne de partitionnement appartiendrait dans une table partitionnée spécifiée.
- Tables et index partitionnés :
  - `CREATE TABLE ... ON partition_scheme_name (partition_column_name):` crée une table partitionnée basée sur un schéma de partition et une colonne de partitionnement spécifiés.
  - `CREATE INDEX ... ON partition_scheme_name (partition_column_name):` crée un index partitionné basé sur un schéma de partition et une colonne de partitionnement spécifiés.
- Vues système pour les métadonnées de partitionnement :

  Les vues système suivantes sont ajoutées pour fournir des métadonnées relatives au partitionnement :
  - `sys.destination_data_spaces`
  - `sys.partitions`
  - `sys.partition_functions`
  - `sys.partition_parameters`
  - `sys.partition_range_values`
  - `sys.partition_schemes`

Limites et solutions de contournement

Les fonctionnalités de partitionnement suivantes de SQL Server ne sont pas encore prises en charge par Babelfish :

- `ALTER PARTITION FUNCTION` et `ALTER PARTITION SCHEME`.

> **Note**
>
> Babelfish ne prend pas en charge les opérations de division et de fusion. Définissez toutes les partitions dans les fonctions de partition lors de leur création, car vous ne pourrez ni ajouter, ni supprimer de partitions ultérieurement.

- Colonnes calculées en tant que colonnes de partitionnement.
- Utilitaire `INSERT BULK` et `BCP` pour les tables partitionnées.
- Option de limite `LEFT` pour les fonctions de partition.
- Type de données `SQL_VARIANT` pour les fonctions de partition.
- `TRUNCATE TABLE ... WITH PARTITION`.
- `ALTER TABLE ... SWITCH PARTITION`.
- Index partitionnés non alignés tels que le schéma de partition et la colonne de partition qui diffèrent de la table partitionnée.
- La migration DMS à partir de la source Babelfish n'est prise en charge que pour les tâches de chargement complet sur les tables partitionnées.
- Babelfish ne prend pas en charge ces options de syntaxe, mais propose des solutions de contournement :
  - Utilisation d'un schéma de partition avec des contraintes ou des index dans l'instruction CREATE TABLE.
  - ALTER TABLE ... ADD CONSTRAINT ... ON partition_scheme_name (partition_column_name).

> **Note**
>
> Vous pouvez configurer la trappe de secours `babelfishpg_tsql.escape_hatch_storage_on_partition` pour ignorer le schéma de partition. Cela permet à l'analyseur d'ignorer l'option de schéma de partition utilisée avec les contraintes ou les index. Le système dorsal créera des contraintes ou des index individuels pour chaque partition.

Résolution des problèmes liés à Babelfish

Vous trouverez ci-dessous des idées de résolution et des solutions de contournement pour certains problèmes de cluster de bases de données Babelfish.

Rubriques

Échec de connexion

Les causes les plus courantes des échecs de connexion à un nouveau cluster de bases de données Aurora doté de Babelfish sont les suivantes :

- Le groupe de sécurité n'autorise pas l'accès – Si vous rencontrez des difficultés pour vous connecter à une instance Babelfish, vérifiez que vous avez ajouté votre adresse IP au groupe de sécurité Amazon EC2 par défaut. Vous pouvez utiliser https://checkip.amazonaws.com/ pour déterminer votre adresse IP, puis l'ajouter à votre règle de trafic entrant pour le port TDS et le port PostgreSQL. Pour en savoir plus, consultez Ajout de règles à un groupe de sécurité dans le Guide de l'utilisateur Amazon EC2.
- Mismatching SSL configurations (Configurations SSL non concordantes) — si le paramètre `rds.force_ssl` est activé (défini sur 1) sur Aurora PostgreSQL, les clients doivent se connecter à Babelfish via SSL. Si votre client n'est pas configuré correctement, vous verrez apparaître un message d'erreur comme celui-ci :

  ```
  Cannot connect to your-Babelfish-DB-cluster, 1433
  --------------------
  ADDITIONAL INFORMATION:
  no pg_hba_conf entry for host "256.256.256.256", user "your-user-name",
  "database babelfish_db", SSL off (Microsoft SQL Server, Error: 33557097)
  ...
  ```

  Cette erreur indique un possible problème de configuration SSL entre votre client local et le cluster de base de données Babelfish. De plus, le cluster exige que les clients utilisent SSL (son paramètre `rds.force_ssl` est réglé sur 1). Pour plus d'informations sur la configuration de SSL, consultez Using SSL with a PostgreSQL DB instance (Utiliser SSL avec une instance de base de données PostgreSQL) dans le Guide de l'utilisateur Amazon RDS.

Si vous utilisez SQL Server Management Studio (SSMS) pour vous connecter à Babelfish et que vous voyez cette erreur, vous pouvez sélectionner les options Encrypt connection (Chiffrer la connexion) et Trust server certificate (Faire confiance au certificat de connexion du serveur) dans le volet Connection Properties (Propriétés de la connexion) et réessayer. Ces paramètres gèrent l'exigence de connexion SSL pour SSMS.

Pour en savoir plus sur la résolution des problèmes de connexion Aurora, consultez Impossible de se connecter à l'instance de base de données Amazon RDS.

Désactivation de Babelfish

Lorsque vous n'avez plus besoin de Babelfish, vous pouvez désactiver la fonctionnalité Babelfish.

Tenez compte des considérations suivantes :

- Dans certains cas, vous pouvez désactiver Babelfish avant de terminer une migration vers Aurora PostgreSQL. Si vous le désactivez et que votre DDL dépend de types de données SQL Server, ou que vous utilisez une fonctionnalité T-SQL dans votre code, votre code échoue.
- Si, après avoir approvisionné une instance Babelfish, vous désactivez l'extension Babelfish, vous ne pourrez plus approvisionner cette même base de données sur le même cluster.

Pour désactiver Babelfish, modifiez votre groupe de paramètres en définissant`rds.babelfish_status` sur `OFF`. Vous pouvez continuer à utiliser vos types de données SQL Server lorsque Babelfish est désactivé, en définissant `rds.babelfish_status` sur `datatypesonly`.

Si vous désactivez Babelfish dans un groupe de paramètres, tous les clusters qui utilisent ce groupe de paramètres perdent la fonctionnalité Babelfish.

Pour en savoir plus sur la modification des groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora. Pour en savoir plus sur les paramètres propres à Babelfish, consultez Paramètres du groupe de paramètres de cluster de bases de données pour Babelfish.

Gestion des mises à jour de version de Babelfish pour Aurora PostgreSQL

Babelfish est une option disponible avec Aurora PostgreSQL 13.4 et versions ultérieures. Les mises à jour de Babelfish sont disponibles avec certaines nouvelles versions du moteur de base de données Aurora PostgreSQL. Pour plus d'informations, consultez Notes de mise à jour d'Aurora PostgreSQL.

> **Note**
>
> Les clusters de bases de données Babelfish exécutés sur n'importe quelle version d'Aurora PostgreSQL 13 ne peuvent pas être mis à niveau vers Aurora PostgreSQL 14.3, 14.4 et 14.5. De plus, Babelfish ne prend pas en charge la mise à niveau directe de la version 13.x vers la version 15.x. Vous devez d'abord mettre à niveau votre cluster de bases de données 13.x vers les versions 14.6 et ultérieures, puis passer à la version 15.x.

Pour obtenir la liste des fonctionnalités prises en charge dans les différentes versions de Babelfish, consultez Fonctionnalités prises en charge dans Babelfish, classées par version.

Pour obtenir la liste des fonctionnalités non prises en charge actuellement, consultez Fonctionnalités non prises en charge dans Babelfish.

Vous pouvez obtenir une liste des versions d'Aurora PostgreSQL compatibles avec Babelfish en interrogeant votre Région AWS à l'aide de la commande describe-db-engine-versionsAWS CLI, comme suit.

Pour Linux, macOS ou Unix :

```
$ aws rds describe-db-engine-versions --region us-east-1 \
    --engine aurora-postgresql \
    --query '*[]|[?SupportsBabelfish==`true`].[EngineVersion]' \
    --output text
```

Pour plus d'informations, consultez describe-db-engine-versions dans la Référence des commandes de l'AWS CLI.

Dans les rubriques suivantes, vous pouvez apprendre à identifier la version de Babelfish exécutée sur votre cluster de bases de données Aurora PostgreSQL et à effectuer une mise à niveau vers une nouvelle version.

Table des matières

Identification de votre version de Babelfish

Vous pouvez interroger Babelfish pour rechercher des informations sur la version de Babelfish, la version d'Aurora PostgreSQL et la version compatible de Microsoft SQL Server. Vous pouvez utiliser le port TDS ou le port PostgreSQL.

Pour utiliser le port TDS pour rechercher des informations de version

1. Utilisez `sqlcmd` ou `ssms` pour vous connecter au point de terminaison pour votre cluster de bases de données Babelfish.

   ```
   sqlcmd -S bfish_db.cluster-123456789012.aws-region.rds.amazonaws.com,1433 -U
       login-id -P password -d db_name
   ```

2. Pour identifier la version de Babelfish, exécutez la requête suivante :

   ```
   1> SELECT CAST(serverproperty('babelfishversion') AS VARCHAR)
   2> GO
   ```

La requête renvoie des résultats semblables à ce qui suit :

```
serverproperty
------------------------------
3.4.0

(1 rows affected)
```

3. Pour identifier la version du cluster de bases de données Aurora PostgreSQL, exécutez la requête suivante :

```
1> SELECT aurora_version() AS aurora_version
2> GO
```

La requête renvoie des résultats semblables à ce qui suit :

```
aurora_version


------------------------------------------------
15.5.0

(1 rows affected)
```

4. Pour identifier la version compatible de Microsoft SQL Server, exécutez la requête suivante :

```
1> SELECT @@VERSION AS version
2> GO
```

La requête renvoie des résultats semblables à ce qui suit :

```
Babelfish for Aurora PostgreSQL with SQL Server Compatibility - 12.0.2000.8
Dec 7 2023 09:43:06
Copyright (c) Amazon Web Services
PostgreSQL 15.5 on x86_64-pc-linux-gnu (Babelfish 3.4.0)

(1 rows affected)
```

Vous pouvez exécuter l'exemple de requête suivant qui montre une différence mineure entre Babelfish et Microsoft SQL Server. Sur Babelfish, la requête renvoie 1, tandis que sur Microsoft SQL Server, la requête renvoie NULL.

```
SELECT CAST(serverproperty('babelfish') AS VARCHAR) AS runs_on_babelfish
```

Vous pouvez également utiliser le port PostgreSQL pour obtenir des informations de version, comme indiqué dans la procédure suivante.

Pour utiliser le port PostgreSQL pour rechercher des informations de version

1. Utilisez `psql` ou `pgAdmin` pour vous connecter au point de terminaison pour votre cluster de bases de données Babelfish.

   ```
   psql host=bfish_db.cluster-123456789012.aws-region.rds.amazonaws.com
        port=5432 dbname=babelfish_db user=sa
   ```

2. Activez la fonction étendue (`\x`) de `psql` pour obtenir une sortie plus lisible.

   ```
   babelfish_db=> \x
   babelfish_db=> SELECT
   babelfish_db=> aurora_version() AS aurora_version,
   babelfish_db=> version() AS postgresql_version,
   babelfish_db=> sys.version() AS Babelfish_compatibility,
   babelfish_db=> sys.SERVERPROPERTY('BabelfishVersion') AS Babelfish_Version;
   ```

   La requête renvoie un résultat semblable à ce qui suit :

   ```
   -[ RECORD 1 ]-----------
   +-------------------------------------------------------------------------------------------
   aurora_version          | 15.5.0
   postgresql_version      | PostgreSQL 15.5 on x86_64-pc-linux-gnu, compiled by
    x86_64-pc-linux-gnu-gcc (GCC) 9.5.0, 64-bit
   babelfish_compatibility | Babelfish for Aurora Postgres with SQL Server
    Compatibility - 12.0.2000.8                    +
                           | Dec 7 2023 09:43:06
                                              +
                           | Copyright (c) Amazon Web Services
                                               +
                           | PostgreSQL 15.5 on x86_64-pc-linux-gnu (Babelfish 3.4.0)
   ```

```
babelfish_version       | 3.4.0
```

Mise à niveau de votre cluster Babelfish vers une nouvelle version

De nouvelles versions de Babelfish sont disponibles avec certaines nouvelles versions du moteur de base de données Aurora PostgreSQL après la version 13.4. Chaque nouvelle version de Babelfish possède son propre numéro de version. Comme avec Aurora PostgreSQL, Babelfish utilise le schéma de dénomination *majeurmineurcorrectif* pour les versions. Par exemple, la première version de Babelfish, version 1.0.0, est devenue disponible dans le cadre d'Aurora PostgreSQL 13.4.0.

Babelfish ne nécessite pas de processus d'installation distinct. Comme indiqué dans Création d'un cluster de bases de données Babelfish pour Aurora PostgreSQL, Turn on Babelfish (Activer Babelfish) est une option que vous choisissez lorsque vous créez un cluster de bases de données Aurora PostgreSQL.

De même, vous ne pouvez pas mettre à niveau Babelfish indépendamment du cluster de bases de données Aurora compatible. Pour mettre à niveau un cluster de bases de données Babelfish for Aurora PostgreSQL existant vers une nouvelle version de Babelfish, vous mettez à niveau le cluster de bases de données Aurora PostgreSQL vers une nouvelle version prenant en charge la version de Babelfish que vous souhaitez utiliser. La procédure à suivre pour la mise à niveau dépend de la version d'Aurora PostgreSQL qui prend en charge votre déploiement de Babelfish, comme suit.

Mises à niveau de version majeure.

Vous devez mettre à niveau les versions suivantes d'Aurora PostgreSQL vers Aurora PostgreSQL versions 14.6 et ultérieures avant la mise à niveau vers Aurora PostgreSQL version 15.2.

- Aurora PostgreSQL versions 13.8 et ultérieures
- Aurora PostgreSQL versions 13.7.1 et mineures ultérieures
- Aurora PostgreSQL versions 13.6.4 et mineures ultérieures

Vous pouvez mettre à niveau Aurora PostgreSQL versions 14.6 et ultérieures vers Aurora PostgreSQL versions 15.2 et ultérieures.

La mise à niveau du cluster de bases de données Aurora PostgreSQL vers une nouvelle version majeure implique plusieurs tâches préliminaires. Pour de plus amples informations, consultez Réalisation d'une mise à niveau de version majeure. Pour réussir la mise à niveau de votre cluster de bases de données Babelfish for Aurora PostgreSQL, vous devez créer un groupe

de paramètres de cluster de bases de données personnalisé pour la nouvelle version d'Aurora PostgreSQL. Ce nouveau groupe de paramètres doit contenir les mêmes paramètres Babelfish que ceux du cluster que vous mettez à niveau. Pour plus d'informations et pour consulter une table des sources et cibles de mise à niveau des versions majeures, consultez Mise à niveau de Babelfish vers une nouvelle version majeure.

Mises à niveau des versions mineures et correctifs

Les versions mineures et les correctifs ne nécessitent pas la création d'un nouveau groupe de paramètres de cluster de bases de données pour la mise à niveau. Les versions mineures et les correctifs peuvent utiliser le processus de mise à niveau des versions mineures, qu'il soit appliqué automatiquement ou manuellement. Pour plus d'informations et pour consulter une table des sources et cibles de versions, consultez Mise à niveau de Babelfish vers une nouvelle version mineure.

> **Note**
>
> Avant d'effectuer une mise à niveau majeure ou mineure, appliquez toutes les tâches de maintenance en attente à votre cluster Babelfish for Aurora PostgreSQL.

Rubriques

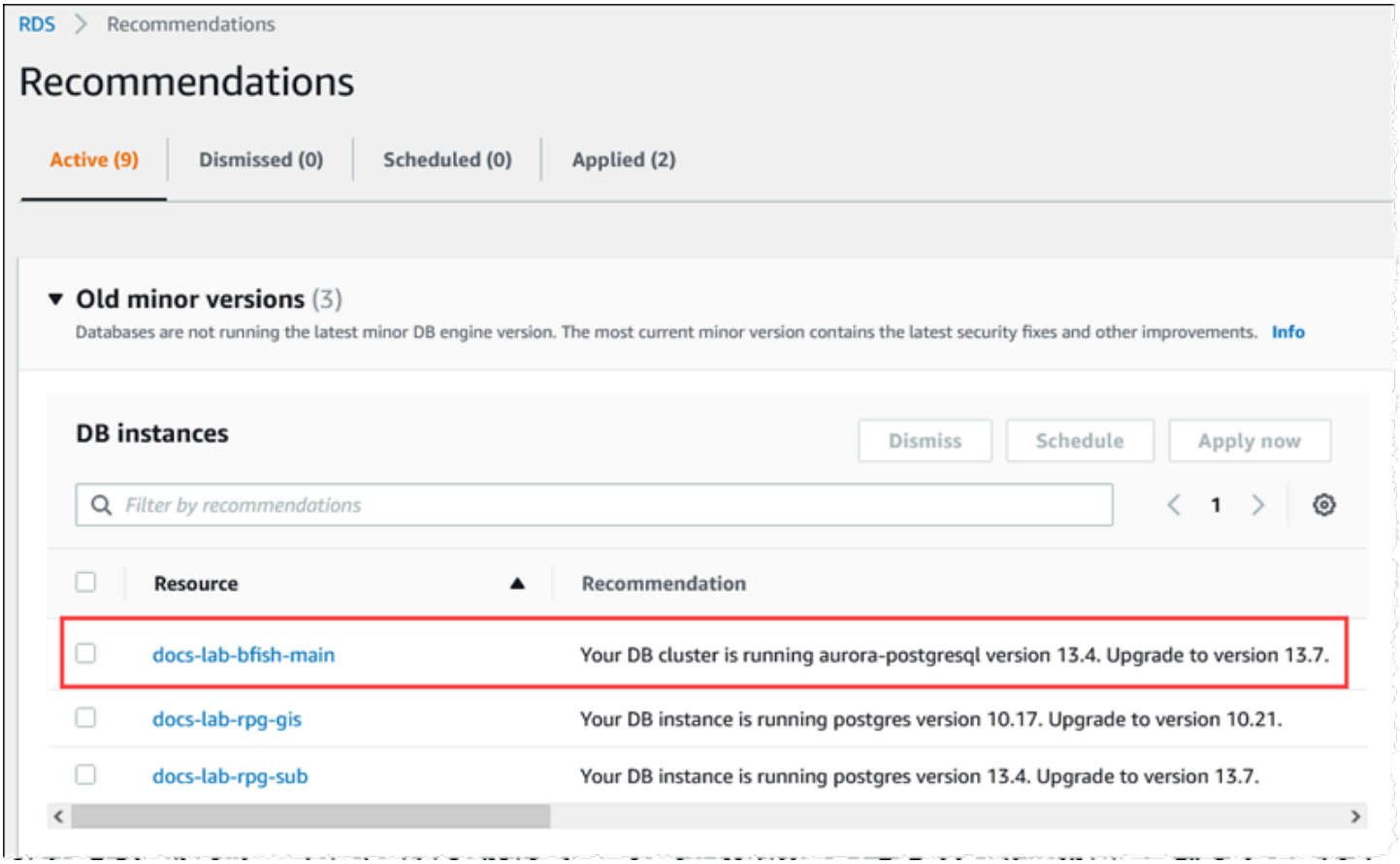
- Mise à niveau de Babelfish vers une nouvelle version mineure
- Mise à niveau de Babelfish vers une nouvelle version majeure

Mise à niveau de Babelfish vers une nouvelle version mineure

Une mise à jour de version mineure vise à assurer la rétrocompatibilité. Cependant, dans certains cas, les correctifs de sécurité critiques ou les corrections de bogues majeures ne sont pas toujours entièrement rétrocompatibles. Une version de correctif Aurora inclut des corrections importantes apportées à une version mineure après sa publication. Par exemple, l'étiquette de version de la première version d'Aurora PostgreSQL 13.4 était Aurora PostgreSQL 13.4.0. Plusieurs correctifs pour cette version mineure ont été publiés à ce jour, notamment Aurora PostgreSQL 13.4.1, 13.4.2 et 13.4.4. Vous pouvez trouver les correctifs disponibles pour chaque version d'Aurora PostgreSQL dans la liste Versions de correctifs en haut des notes de mise à jour d'Aurora PostgreSQL pour cette version. Pour obtenir un exemple, consultez PostgreSQL 14.3 dans Notes de mise à jour d'Aurora PostgreSQL.

Si votre cluster de bases de données Aurora PostgreSQL est configuré avec l'option Auto minor version upgrade (Mise à niveau automatique de la version mineure), votre cluster de bases de données Babelfish pour Aurora PostgreSQL est automatiquement mis à niveau pendant la fenêtre de maintenance du cluster. Pour en savoir plus sur la mise à niveau automatique de version mineure (AmVU) et sur son utilisation, consultez Mises à niveau automatiques des versions mineures pour les clusters de bases de données Aurora. Si votre cluster n'utilise pas AmVU, vous pouvez mettre à niveau manuellement votre cluster de bases de données Babelfish pour Aurora PostgreSQL vers de nouvelles versions mineures, soit en répondant aux tâches de maintenance, soit en modifiant le cluster pour utiliser la nouvelle version.

Lorsque vous choisissez une version d'Aurora PostgreSQL à installer et que vous consultez un cluster de bases de données Aurora PostgreSQL existant dans la AWS Management Console, la version affiche uniquement les chiffres *majeurs*`mineurs`. Par exemple, l'image suivante tirée de la console d'un cluster de bases de données Babelfish pour Aurora PostgreSQL existant avec Aurora PostgreSQL 13.4 recommande de mettre à niveau le cluster vers la version 13.7, une nouvelle version mineure d'Aurora PostgreSQL.



Pour obtenir les détails complets de la version, y compris le niveau de *correctif*, vous pouvez interroger le cluster de bases de données Aurora PostgreSQL à l'aide de la fonction `aurora_version` d'Aurora PostgreSQL. Pour plus d'informations, consultez aurora_version dans le Référence sur les fonctions Aurora PostgreSQL. Vous trouverez un exemple d'utilisation de cette fonction dans la procédure To use the PostgreSQL port to query for version information de Identification de votre version de Babelfish.

La table suivante présente les versions d'Aurora PostgreSQL et de Babelfish, ainsi que les versions cibles disponibles, qui peuvent prendre en charge le processus de mise à niveau des versions mineures.

Versions sources actuelles	Cibles de mise à niveau les plus récentes
Aurora PostgreSQL (Babelfish)	Aurora PostgreSQL (Babelfish)
17.4 (5.1)	17.5 (5.2)
16.8 (4.5)	16.9 (4.6)
16.6 (4.4.0)	16.9 (4.6), 16.8 (4.5.0)
16.4 (4.3.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0)
16.3 (4.2.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0)
16.2 (4.1.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0)
16.1 (4.0.0)	16.9 (4.6), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0)
15.12 (3.9.0)	15.13 (3.10)
15.10 (3.8.0)	15.13 (3.10), 15.12 (3.9.0)
15.8 (3.7.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0)
15.7 (3.6.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0)

Versions sources actuelles	Cibles de mise à niveau les plus récentes
15.6 (3.5.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0)
15.5 (3.4.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0)
15.4 (3.3.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0)
15.12 (3.9.0), 15.3 (3.2.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0)
15.2 (3.1.0)	15.13 (3.10), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0)
14.17 (2.12.0)	14.18 (2.13.0)
14.15 (2.11.0)	14.18 (2.13.0), 14.17 (2.12.0)
14.13 (2.10.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0)
14.12 (2.9.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0)
14.11 (2.8.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0)
14.10 (2.7.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0)
14.9 (2.6.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0)

Versions sources actuelles	Cibles de mise à niveau les plus récentes
14.8 (2.5.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0)
14.7 (2.4.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0)
14.6 (2.3.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0), 14.7 (2.4.0)
14.5 (2.2.0)	14.18 (2.13.0), 14.17 (2.12.0), 14.15 (2.11.0), 14.13 (2.10.0), 14.12 (2.9.0), 14.11 (2.8.0), 14.10 (2.7.0), 14.9 (2.6.0), 14.8 (2.5.0), 14.7 (2.4.0), 14.6 (2.3.0)
14.3 (2.1.0)	14.18 (2.13.0), 14.6 (2.3.0)
13.8 (1.4.0)	13.9 (1.5.0)
13.7 (1.3.0)	13.9 (1.5.0), 13.8 (1.4.0)

Mise à niveau de Babelfish vers une nouvelle version majeure

Pour une mise à niveau de version majeure, vous devez d'abord mettre à niveau votre cluster de bases de données Babelfish pour Aurora PostgreSQL vers une version prenant en charge la mise à niveau de la version majeure. Pour ce faire, appliquez des mises à niveau de correctifs ou de versions mineures à votre cluster de bases de données. Pour plus d'informations, consultez Mise à niveau de Babelfish vers une nouvelle version mineure.

La table suivante présente les versions d'Aurora PostgreSQL et de Babelfish qui peuvent prendre en charge une mise à niveau d'une version majeure.

Versions sources actuelles	Cibles de mise à niveau les plus récentes
Aurora PostgreSQL (Babelfish)	Aurora PostgreSQL (Babelfish)
16.9 (4.6.0)	17.5 (5.2.0)
16.8 (4.5.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.6 (4.4.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.4 (4.3.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.3 (4.2.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.2 (4.1.0)	17.5 (5.2.0), 17.4 (5.1.0)
16.1 (4.0.0)	17.5 (5.2.0), 17.4 (5.1.0)
15.13 (3.10)	17.5 (5.2.0) 16.9 (4.6.0)
15.12 (3.9.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0)
15.10 (3.8.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0)
15.8 (3.7.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0)
15.7 (3.6.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0)
15.6 (3.5.0)	17.5 (5.2.0), 17.4 (5.1.0)

Versions sources actuelles	Cibles de mise à niveau les plus récentes
	16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0)
15.5 (3.4.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.4 (3.3.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.3 (3.2.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
15.2 (3.1.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0)
14.18 (2.13.0)	17.5 (5.2.0) 16.9 (4.6.0) 15.13 (3.10.0)
14.17 (2.12.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0) 15.13 (3.10.0), 15.12 (3.9.0)
14.15 (2.11.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0)

Versions sources actuelles	Cibles de mise à niveau les plus récentes
14.13 (2.10.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0)
14.12 (2.9.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0)
14.11 (2.8.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0)
14.10 (2.7.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0)

Versions sources actuelles	Cibles de mise à niveau les plus récentes
14.9 (2.6.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0)
14.8 (2.5.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0)
14.7 (2.4.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0), 15.2 (3.1.0)
14.6 (2.3.0)	17.5 (5.2.0), 17.4 (5.1.0) 16.9 (4.6.0), 16.8 (4.5.0), 16.6 (4.4.0), 16.4 (4.3.0), 16.3 (4.2.0), 16.2 (4.1.0), 16.1 (4.0.0) 15.13 (3.10.0), 15.12 (3.9.0), 15.10 (3.8.0), 15.8 (3.7.0), 15.7 (3.6.0), 15.6 (3.5.0), 15.5 (3.4.0), 15.4 (3.3.0), 15.3 (3.2.0), 15.2 (3.1.0)
13.9 (1.5.0)	14.6 (2.3.0)

Versions sources actuelles	Cibles de mise à niveau les plus récentes
13.8 (1.4.0)	14.6 (2.3.0)
13.7 (1.3.0)	14.6 (2.3.0)

Avant la mise à niveau de Babelfish vers une nouvelle version majeure

Une mise à niveau peut entraîner de brèves interruptions. Nous vous recommandons ainsi d'effectuer ou de planifier vos mises à niveau pendant votre fenêtre de maintenance ou pendant les périodes de faible utilisation.

Avant d'effectuer une mise à niveau de version majeure

1. Identifiez la version Babelfish de votre cluster de bases de données Aurora PostgreSQL existant à l'aide des commandes décrites dans Identification de votre version de Babelfish. Les informations relatives aux versions d'Aurora PostgreSQL et de Babelfish sont gérées par PostgreSQL. Suivez donc les étapes décrites dans la procédure To use the PostgreSQL port to query for version information pour plus de détails.
2. Vérifiez si votre version prend en charge la mise à niveau de la version majeure. Pour obtenir la liste des versions qui prennent en charge la fonction de mise à niveau des versions majeures, consultez Mise à niveau de Babelfish vers une nouvelle version mineure et effectuez les tâches préalables à la mise à niveau nécessaires.

   Par exemple, si votre version de Babelfish s'exécute sur un cluster de bases de données Aurora PostgreSQL 13.5 et que vous souhaitez effectuer une mise à niveau vers Aurora PostgreSQL 15.2, appliquez d'abord toutes les versions mineures et tous les correctifs pour mettre à niveau votre cluster vers Aurora PostgreSQL versions 14.6 ou ultérieures. Lorsque votre cluster est à la version 14.6 ou ultérieure, poursuivez le processus de mise à niveau de la version majeure.
3. Créez un instantané manuel de votre cluster de bases de données Babelfish actuel en tant que sauvegarde. La sauvegarde vous permet de restaurer le cluster à sa version Aurora PostgreSQL, à sa version Babelfish et de restaurer toutes les données dans l'état où elles se trouvaient avant la mise à niveau. Pour plus d'informations, consultez Création d'un instantané de cluster de bases de données. Veillez à conserver votre groupe de paramètres de cluster de bases de données personnalisé existant pour pouvoir le réutiliser si vous décidez de restaurer ce cluster à son état antérieur à la mise à niveau. Pour plus d'informations, consultez Restauration

à partir d'un instantané de cluster de bases de données et Considérations relatives au groupe de paramètres.

4. Préparez un groupe de paramètres de cluster de bases de données personnalisé pour la version de base de données Aurora PostgreSQL cible. Dupliquez les paramètres Babelfish de votre cluster de bases de données Babelfish pour Aurora PostgreSQL actuel. Pour obtenir la liste de tous les paramètres de Babelfish, consultez Paramètres du groupe de paramètres de cluster de bases de données pour Babelfish. Pour une mise à niveau de version majeure, les paramètres suivants nécessitent les mêmes paramètres que le cluster de bases de données source. Pour que la mise à niveau réussisse, tous les paramètres doivent être identiques.

   - rds.babelfish_status
   - babelfishpg_tds.tds_default_numeric_precision
   - babelfishpg_tds.tds_default_numeric_scale
   - babelfishpg_tsql.database_name
   - babelfishpg_tsql.default_locale
   - babelfishpg_tsql.migration_mode
   - babelfishpg_tsql.server_collation_name

   > **Warning**
   >
   > Si les paramètres Babelfish du groupe de paramètres de cluster de bases de données personnalisé pour la nouvelle version d'Aurora PostgreSQL ne correspondent pas aux valeurs des paramètres du cluster que vous mettez à niveau, l'opération `ModifyDBCluster` échoue. Un message d'erreur `InvalidParameterCombination` apparaît dans la AWS Management Console ou dans la sortie de la commande `modify-db-cluster` de l'AWS CLI.

5. Utilisez la AWS Management Console ou l'AWS CLI pour créer le groupe de paramètres personnalisé pour le cluster de bases de données. Choisissez la famille Aurora PostgreSQL applicable à la version d'Aurora PostgreSQL que vous souhaitez mettre à niveau.

   > **Tip**
   >
   > Les groupes de paramètres sont gérés au niveau de la Région AWS. Lorsque vous travaillez avec l'AWS CLI, vous pouvez configurer avec une région par défaut au lieu de spécifier la `--region` dans la commande. Pour en savoir plus sur l'AWS CLI, consultez

Quick setup (Configuration rapide) dans le Guide de l'utilisateur de l'AWS Command Line Interface.

Réalisation d'une mise à niveau de version majeure

1. Mettez à niveau un cluster de bases de données Aurora PostgreSQL vers une nouvelle version majeure. Pour plus d'informations, consultez Mise à niveau du moteur Aurora PostgreSQL vers une nouvelle version majeure.
2. Redémarrez l'instance d'enregistreur du cluster afin que les paramètres puissent prendre effet.

Après la mise à niveau vers une nouvelle version majeure

Après la mise à niveau d'une version majeure vers une nouvelle version d'Aurora PostgreSQL, la valeur `IDENTITY` des tables comportant une colonne `IDENTITY` peut être plus grande (+32) qu'elle ne l'était avant la mise à niveau. Il en résulte que lorsque la ligne suivante est insérée dans de telles tables, la valeur de la colonne d'identité générée passe au chiffre +32 et commence la séquence à partir de là. Cette condition n'affectera pas négativement les fonctions de votre cluster de bases de données Babelfish. Toutefois, si vous le souhaitez, vous pouvez réinitialiser l'objet de séquence en fonction de la valeur maximale de la colonne. Pour ce faire, connectez-vous au port T-SQL sur votre instance d'enregistreur Babelfish avec `sqlcmd` ou un autre client SQL Server. Pour plus d'informations, consultez Utilisation d'un client SQL Server pour se connecter au cluster de bases de données.

```
sqlcmd -S bfish-db.cluster-123456789012.aws-region.rds.amazonaws.com,1433 -U
     sa -P ******** -d dbname
```

Lorsque vous êtes connecté, utilisez la commande SQL suivante pour générer des instructions que vous pouvez utiliser pour amorcer l'objet de séquence associé. Cette commande SQL fonctionne à la fois pour les configurations Babelfish avec une seule ou plusieurs bases de données. Pour plus d'informations sur ces deux modèles de déploiement, consultez Utilisation de Babelfish avec une ou plusieurs bases de données.

```
DECLARE @schema_prefix NVARCHAR(200) = ''
IF current_setting('babelfishpg_tsql.migration_mode') = 'multi-db'
    SET @schema_prefix = db_name() + '_'
SELECT 'SELECT setval(pg_get_serial_sequence(''' + @schema_prefix +
 schema_name(tables.schema_id)
```

```
    + '.' + tables.name + ''', ''' + columns.name + '''),(select max(' + columns.name +
 ')
    FROM ' + schema_name(tables.schema_id) + '.' + tables.name + '));
    'FROM sys.tables tables JOIN sys.columns
    columns ON tables.object_id = columns.object_id
    WHERE columns.is_identity = 1
GO
```

La requête génère une série d'instructions SELECT que vous pouvez ensuite exécuter pour réinitialiser la valeur IDENTITY maximale et combler toute lacune. Ce qui suit montre la sortie obtenue lors de l'utilisation de l'exemple de base de données SQL Server, Northwind, exécuté sur un cluster Babelfish.

```
------------------------------------------------------------
SELECT setval(pg_get_serial_sequence('northwind_dbo.categories', 'categoryid'),(select
 max(categoryid)
    FROM dbo.categories));

SELECT setval(pg_get_serial_sequence('northwind_dbo.orders', 'orderid'),(select
 max(orderid)
    FROM dbo.orders));


SELECT setval(pg_get_serial_sequence('northwind_dbo.products', 'productid'),(select
 max(productid)
    FROM dbo.products));


SELECT setval(pg_get_serial_sequence('northwind_dbo.shippers', 'shipperid'),(select
 max(shipperid)
    FROM dbo.shippers));


SELECT setval(pg_get_serial_sequence('northwind_dbo.suppliers', 'supplierid'),(select
 max(supplierid)
    FROM dbo.suppliers));


(5 rows affected)
```

Exécutez les instructions une par une pour réinitialiser les valeurs de séquence.

Exemple : mise à niveau du cluster de bases de données Babelfish vers une version majeure

Dans cet exemple, vous trouverez la série de commandes AWS CLI expliquant comment mettre à niveau un cluster de bases de données Aurora PostgreSQL 13.6.4 exécutant Babelfish version 1.2.2 vers Aurora PostgreSQL 14.6. Vous créez d'abord un groupe de paramètres de cluster de bases de données personnalisé pour Aurora PostgreSQL 14. Vous modifiez ensuite les valeurs des paramètres pour qu'elles correspondent à celles de votre source Aurora PostgreSQL version 13. Enfin, vous effectuez la mise à niveau en modifiant le cluster source. Pour plus d'informations, consultez Paramètres du groupe de paramètres de cluster de bases de données pour Babelfish. Dans cette rubrique, vous trouverez également des informations sur la mise à niveau avec la AWS Management Console.

Utilisez la commande de l'interface de ligne de commande create-db-cluster-parameter-group pour créer le groupe de paramètres de cluster de bases de données pour la nouvelle version.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster-parameter-group \
    --db-cluster-parameter-group-name docs-lab-babelfish-apg-14 \
    --db-parameter-group-family aurora-postgresql14 \
    --description 'New custom parameter group for upgrade to new major version' \
    --region us-west-1
```

Lorsque vous exécutez cette commande, le groupe de paramètres du cluster de bases de données personnalisé est créé dans la Région AWS. Vous voyez des résultats similaires à ce qui suit.

```
{
    "DBClusterParameterGroup": {
        "DBClusterParameterGroupName": "docs-lab-babelfish-apg-14",
        "DBParameterGroupFamily": "aurora-postgresql14",
        "Description": "New custom parameter group for upgrade to new major version",
        "DBClusterParameterGroupArn": "arn:aws:rds:us-west-1:111122223333:cluster-
pg:docs-lab-babelfish-apg-14"
    }
}
```

Pour plus d'informations, consultez Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

Utilisez la commande de l'interface de ligne de commande modify-db-cluster-parameter-group pour modifier les paramètres afin qu'ils correspondent au cluster source.

Pour Windows :

```
aws rds modify-db-cluster-parameter-group --db-cluster-parameter-group-name docs-lab-
babelfish-apg-14 ^
  --parameters
 "ParameterName=rds.babelfish_status,ParameterValue=on,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tds.tds_default_numeric_precision,ParameterValue=38,ApplyMethod=pen
reboot" ^
 "ParameterName=babelfishpg_tds.tds_default_numeric_scale,ParameterValue=8,ApplyMethod=pending-
reboot" ^
 "ParameterName=babelfishpg_tsql.database_name,ParameterValue=babelfish_db,ApplyMethod=pending-
reboot" ^
 "ParameterName=babelfishpg_tsql.default_locale,ParameterValue=en-
US,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tsql.migration_mode,ParameterValue=single-
db,ApplyMethod=pending-reboot" ^
 "ParameterName=babelfishpg_tsql.server_collation_name,ParameterValue=sql_latin1_general_cp1_ci
reboot"
```

La réponse ressemble à ce qui suit.

```
{
    "DBClusterParameterGroupName": "docs-lab-babelfish-apg-14"
}
```

Utilisez la commande de l'interface de ligne de commande modify-db-cluster pour modifier le cluster afin d'utiliser la nouvelle version et le nouveau groupe de paramètres du cluster de bases de données personnalisé. Vous spécifiez également l'argument `--allow-major-version-upgrade`, comme indiqué dans l'exemple suivant.

```

aws rds modify-db-cluster \
--db-cluster-identifier docs-lab-bfish-apg-14 \
--engine-version 14.6 \
--db-cluster-parameter-group-name docs-lab-babelfish-apg-14 \
--allow-major-version-upgrade \
--region us-west-1 \
--apply-immediately
```

Utilisez la commande de l'interface de ligne de commande reboot-db-instance pour redémarrer l'instance d'enregistreur du cluster, afin que les paramètres puissent prendre effet.

```
aws rds reboot-db-instance \
--db-instance-identifier docs-lab-bfish-apg-14-instance-1\
--region us-west-1
```

Utilisation du paramètre de version du produit Babelfish

Un nouveau paramètre GUC (Grand Unified Configuration) appelé `babelfishpg_tds.product_version` est introduit à partir des versions 2.4.0 et 3.1.0 de Babelfish. Ce paramètre vous permet de définir le numéro de version du produit SQL Server comme sortie de Babelfish.

Le paramètre est une chaîne d'identifiant de version en 4 parties, et chaque partie doit être séparée par « . ».

Syntaxe

```
Major.Minor.Build.Revision
```

- Version majeure : un nombre compris entre 11 et 16.
- Version majeure : un nombre compris entre 0 et 255.
- Version de build : un nombre compris entre 0 et 65 535.
- Révision : 0 et tout nombre positif.

Configuration du paramètre de version du produit Babelfish

Vous devez utiliser le groupe de paramètres du cluster pour définir le paramètre babelfishpg_tds.product_version dans la console. Pour plus d'informations sur la modification du paramètre de cluster de bases de données, consultez Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

Lorsque vous définissez le paramètre de version du produit sur une valeur non valide, la modification ne prend pas effet. Bien que la console puisse afficher la nouvelle valeur, le paramètre conserve la valeur précédente. Consultez le fichier journal du moteur pour avoir plus de détails sur les messages d'erreur.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
--db-cluster-parameter-group-name mydbparametergroup \
--parameters
 "ParameterName=babelfishpg_tds.product_version,ParameterValue=15.2.4000.1,ApplyMethod=immediat
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
--db-cluster-parameter-group-name mydbparametergroup ^
--parameters
 "ParameterName=babelfishpg_tds.product_version,ParameterValue=15.2.4000.1,ApplyMethod=immediat
```

Requêtes et paramètres concernés

Requête/paramètre	Résultat	Heure effective
SELECT @@VERSION	Renvoie la version de SQL Server définie par l'utilisateur (valeur babelfishpg_tsql.version = Default)	Immediatement
SELECT SERVERPROPERTY('ProductVersion')	Renvoie la version de SQL Server définie par l'utilisateur	Immediatement
SELECT SERVERPROPERTY('ProductMajorVersion')	Renvoie la version majeure de SQL Server définie par l'utilisateur	Immediatement
Jetons de VERSION dans le message de réponse PRELOGIN	Le serveur renvoie des messages PRELOGIN avec la version de SQL Server définie par l'utilisateur	Prend effet lorsqu'un utilisateur crée une nouvelle session
SQLServerVersion dans LoginAck lors de l'utilisation de JDBC	DatabaseMetaData.getDatabaseProductVersion()	Prend effet lorsqu'un utilisateur crée une nouvelle session

Requête/paramètre	Résultat	Heure effective
	renvoie la version de SQL Server définie par l’utilisateur	

Interface avec le paramètre babelfishpg_tsql.version

Vous pouvez définir la sortie de @@VERSION à l’aide des paramètres babelfishpg_tsql.version et babelfishpg_tds.product_version. Les exemples suivants illustrent l’interface entre ces deux paramètres.

- Lorsque le paramètre babelfishpg_tsql.version est « default » et que babelfishpg_tds.product_version est 15.0.2000.8.
  - Sortie de @@version : 15.0.2000.8.
- Lorsque le paramètre babelfishpg_tsql.version est défini sur 13.0.2000.8 et que le paramètre babelfishpg_tds.product_version est 15.0.2000.8.
  - Sortie de @@version : 13.0.2000.8.

Référence Babelfish for Aurora PostgreSQL

Rubriques

Fonctionnalités prises en charge dans Babelfish, classées par version

Dans les tableaux suivants, vous trouverez la liste de toutes les fonctionnalités prises en charge par les différentes versions de Babelfish. Pour afficher les listes des fonctionnalités non prises en charge, consultez Fonctionnalités non prises en charge dans Babelfish. Pour plus d'informations sur les différentes versions de Babelfish, consultez Notes de mise à jour d'Aurora PostgreSQL.

Fonctionnalités Aurora et PostgreSQL

Dans le tableau suivant, vous trouverez les fonctionnalités Aurora et PostgreSQL prises en charge par les différentes versions de Babelfish.

Fonctionnalités Aurora et PostgreSQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Services ML d'Aurora	O	O	O	O	O	O	O	O	N	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
Authentification de base de données avec Kerberos à l'aide d'Directory Service	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	N	Non
Vidage et restauration	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non

Fonctionnalités Aurora et PostgreSQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Extension pg_stat_statement	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	C	N	N	N	N	N	Non
pgvector	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
Correctifs sans durée d'indisponibilité (ZDP)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	Non

Fonctionnalité ou syntaxe T-SQL

Dans le tableau suivant, vous trouverez la syntaxe ou les fonctionnalités T-SQL prises en charge par les différentes versions de Babelfish.

Fonctionnalité ou syntaxe T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Références de nom d'objet en 4 parties pour les instructions SELECT	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	C	N	N	N	N	N	Non
Syntaxe ALTER AUTHORIZATION pour changer de propriétaire de base de données.	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
ALTER DATABASE *<db_name>*	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non

Fonctionnalité ou syntaxe T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
MODIFY NAME = *<new_db_name>*																																	
ALTER FUNCTION	C	C	C	C	C	N	N	N	N	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
ALTER PROCEDURE	C	C	C	C	C	C	C	N	N	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
ALTER ROLE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
ALTER USER...WITH LOGIN	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
ALTER VIEW	C	N	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
Mot clé AS dans CREATE FUNCTION	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
Clause AT TIME ZONE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
Instance Babelfish en tant que serveur lié	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
Opérateurs de comparaison !< et !>	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non

Fonctionnalité ou syntaxe T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
CREATE INDEX ... ON partition _scheme_name (partition_column_ name)	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
CREATE au lieu de déclencheurs (DML) dans les vues SQL Server	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
CREATE PARTITION FUNCTION	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
CREATE PARTITION SCHEME	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
CREATE ROLE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
CREATE TABLE ... ON partition _scheme_name (partition_column_ name)	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
CREATE TRIGGER	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
CREATE OR ALTER VIEW	C	N	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non

Fonctionnalité ou syntaxe T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Création d’index uniques	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
Exécution de procédures entre bases de données	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non
Références de base de données croisées SELECT, INSERT, UPDATE, DELETE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
Paramètres de type curseur pour les paramètres d’entrée uniquement (pas de sortie)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
Migration de données à l’aide de l’utilitaire client bcp	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
Types de données TIMESTAMP, ROWVERSION (pour plus d’informations sur l’utilisation, consultez Fonctionnalités à implémentation limitée)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Fonctionnalité ou syntaxe T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Mot clé DEFAULT dans les appels aux procédures stockées et aux fonctions	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
DBCC CHECKIDENT	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
DROP DATABASE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	Non
DROP IF EXISTS (pour les objets SCHEMA, DATABASE et USER)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
DROP INDEX ON schema.table.index	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
DROP INDEX schema.table.index	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
DROP PARTITION FUNCTION	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
DROP PARTITION SCHEME	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
DROP ROLE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Fonctionnalité ou syntaxe T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
ENABLE/DISABLE TRIGGER	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
FULL TEXT SEARCH	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
Recherche en texte intégral avec clause CONTAINS	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
Types de données spatiales de géométrie et de géographie	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
GRANT	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
GRANT l’autorisation [,...n] ON SCHEMA	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
GUC babelfish pg_tds.product_version	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	Non
Identifiants avec point de début	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
Déclencheurs INSTEAD OF sur les tables	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Fonctionnalité ou syntaxe T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Déclencheurs INSTEAD OF sur les vues	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
KILL	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
$PARTITION.partition_function_name(partition_column_value)	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
PIVOT (utilisation dans la définition de vue depuis 4.4.0 et 3.8.0, expression de table commune depuis 3.4.0, jointure depuis 3.4.0)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
REVOKE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
REVOKE l'autorisation [,...n] ON SCHEMA	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
Clauses SELECT... OFFSET... FETCH	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
SELECT FOR JSON AUTO	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non

Fonctionnalité ou syntaxe T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SET BABELFISH _SHOWPLAN_ALL ON (et OFF)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
SET BABELFISH _STATISTICS PROFILE ON (OFF)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
SET CONTEXT_I NFO	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
SET LOCK_TIME OUT	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
SET NO_BROWSE TABLE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non
SET rowcount	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
SET SHOWPLAN_ ALL	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non
SET STATISTICS IO	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non
SET TRANSACTI ON ISOLATION LEVEL REPEATABL E READ	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non

Fonctionnalité ou syntaxe T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
Syntaxe SET TRANSACTION ISOLATION LEVEL	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
SSMS : connexion à l’aide de la boîte de dialogue de connexion de l’explorateur d’objets	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
SSMS : migration des données grâce à l’assistant d’importation/exportation	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
SSMS : prise en charge partielle de l’explorateur d’objets	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
STDEV	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	N	N	N	N	N	N	N	Non
STDEVP	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	N	N	N	N	N	N	N	Non
Les déclencheurs comportant plusieurs actions DML peuvent référencer les tables de transition.	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Fonctionnalité ou syntaxe T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2.1.0
Indicateurs T-SQL (méthodes de jointure, utilisation des index, MAXDOP)	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	Non
Syntaxe entre crochets de T-SQL avec le prédicat LIKE	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	O	O	O	O	O	O	O	N	N	N	N	N	Non
UNPIVOT	O	N	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
Valeurs de chaîne sans guillemets dans les appels de procédure stockée et valeurs par défaut	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
VAR	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	O	O	O	O	O	O	O	N	N	N	N	N	Non
VARP	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	O	O	O	O	O	O	O	N	N	N	N	N	Non

Fonctions intégrées T-SQL

Dans le tableau suivant, vous trouverez les fonctionnalités intégrées T-SQL prises en charge par les différentes versions de Babelfish.

Fonctions intégrées T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
APP_NAME	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	Non
ATN2	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	Non
CHARINDEX	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
CHOOSE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
COL_LENGTH	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
COL_NAME	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
COLUMNS_UPDATED	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
COLUMNPROPERTY (CharMaxLen, AllowsNull uniquement)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
CONCAT_WS	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
CONTEXT_INFO																																	

Fonctions intégrées T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	O	O	O	O	O	O	O	N	N	N	N	N	N	Non
CURSOR_STATUS	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
DATABASE_ PRINCIPAL_ID	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	O	O	O	O	O	O	O	N	N	N	N	N	N	Non
DATEADD	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	Non
DATEDIFF	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	Non
DATEDIFF_BIG	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	N	Non
DATEFROMPARTS	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
DATENAME	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	Non
DATEPART	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	Non
DATETIMEF ROMPARTS	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui

Fonctions intégrées T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
DATETIME2 FROMPARTS	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	O	O	O	O	O	O	O	N	N	N	N	N	Non
DATETIMEO FFSETFROM PARTS	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	Non
DATETRUNC	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	N	O	O	O	O	O	O	N	N	N	N	N	N	Non
DATE_BUCKET	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	N	O	O	O	O	O	O	N	N	N	N	N	N	Non
EOMONTH	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	N	O	O	O	O	O	O	N	N	N	N	N	N	Non
EXECUTE AS CALLER	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	Non
fn_listextendedpro perty	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	O	O	O	O	O	O	O	N	N	N	N	N	Non
FOR JSON	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	Non
FULLTEXTS ERVICEPROPERTY	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
HAS_DBACCESS	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui

Fonctions intégrées T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	20	1.0
HAS_PERMS_BY_NAME	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
HOST_NAME	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non
HOST_ID	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	N	N	N	N	N	N	N	Non
IDENTITY	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
IS_MEMBER	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
IS_ROLEMEMBER	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
IS_SRVROLEMEMBER	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
ISJSON	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
JSON_MODIFY	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	Non
JSON_QUERY	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Fonctions intégrées T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
JSON_VALUE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
NEXT VALUE FOR	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	Non
OBJECT_DEFINITION	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	Non
OBJECT_SCHEMA_NAME	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	Non
OPENJSON	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
OPENQUERY	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
ORIGINAL_LOGIN	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
PARSENAME	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	N	N	N	N	N	N	N	Non
PATINDEX	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
ROWCOUNT_BIG	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	N	Non

Fonctions intégrées T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SCHEMA_NAME	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
SESSION_CONTEXT	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	Non
SESSION_USER	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
SID_BINARY (renvoie toujours NULL)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non
SMALLDATETIMEFROMPARTS	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
SQUARE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
STR	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	Non
STRING_AGG	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	C	C	C	C	N	N	N	N	N	N	N	N	Non
STRING_SPLIT	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
SUSER_SID	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Fonctions intégrées T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
SUSER_SNAME	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
SWITCHOFFSET	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
SYSTEM_USER	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non
TIMEFROMPARTS	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	N	N	N	N	N	N	N	Non
TODATETIMEOFFSET	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
TO_CHAR	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	Non
TRIGGER_NESTLEVEL (uniquement sans arguments)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
TRY_CONVERT	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	Non
TYPE_ID	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non

Fonctions intégrées T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
TYPE_NAME	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
UPDATE	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui

Catalogues T-SQL INFORMATION_SCHEMA

Dans le tableau suivant, vous trouverez les catalogues T-SQL INFORMATION_SCHEMA pris en charge par les différentes versions de Babelfish.

Catalogues T-SQL INFORMATION_SCHEMA	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
CHECK_CONSTRAINTS	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Non
COLUMN_DOMAIN_USAGE	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Non
COLUMNS	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
CONSTRAINT_COLUMN_USAGE	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
DOMAINS	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui

Catalogues T-SQL INFORMATION_SCHEMA	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
KEY_COLUMN_USAGE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
ROUTINES	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non
TABLES	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
TABLE_CONSTRAINTS	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
VUES	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Variables @@ définies par le système T-SQL

Dans le tableau suivant, vous trouverez les variables @@ définies par le système T-SQL prises en charge par les différentes versions de Babelfish.

Variables @@ définies par le système T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
@@CURSOR_ROWS	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@DATEFIRST	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Variables @@ définies par le système T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
@@DBTS	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@ERROR	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@ERROR=213	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@FETCH_STATUS	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@IDENTITY	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@LANGUAGE	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@LOCK_TIMEOUT	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@MAX_CONNECTIONS	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@MAX_PRECISION	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Variables @@ définies par le système T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	21.0
@@MICROSOFTVERSION	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@NESTLEVEL	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@PROCID	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@ROWCOUNT	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@SERVERNAME	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@SERVICENAME	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@SPID	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@TRANCOUNT	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
@@VERSION (notez la différence de format décrite dans Différences T-SQL dans Babelfish.	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Procédures stockées du système T-SQL

Dans le tableau suivant, vous trouverez les procédures stockées du système T-SQL prises en charge par les différentes versions de Babelfish.

Procédures stockées du système T-SQL	5.5.4	5.4.4	5.4.3	5.3.3	5.3.2	5.2.2	5.2.1	5.1.0
sp_addextendedproperty	C	C	C	C	C	C	N	Non
sp_addlinkedserver	C	C	C	C	C	C	N	Non
sp_addlinkedsrvlogin	C	C	C	C	C	C	N	Non
sp_addrole	C	C	C	C	C	C	N	Non
sp_addrolemember	C	C	C	C	C	C	N	Non
sp_babelfish_volatility	C	C	C	C	C	C	N	Non
sp_column_privileges	C	C	C	C	C	C	C	Oui
sp_columns	C	C	C	C	C	C	C	Oui
sp_columns_100								

Procédures stockées du système T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2.0 1.0
	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_column s_managed	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_cursor	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_cursor_list	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_cursorclose	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_cursorexecute	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_cursorfetch	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_cursoropen	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_cursoroption	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_cursorprepare	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Procédures stockées du système T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_cursorprepexec	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C		Oui
sp_cursorunprepare	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C		Oui
sp_databases	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C		Oui
sp_datatype_info	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C		Oui
sp_datatype_info_100	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C		Oui
sp_describe_cursor	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C		Oui
sp_describe_first_result_set	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C		Oui
sp_describe_undeclared_parameters	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C		Oui
sp_dropextendedproperty	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
sp_droplinkedsrvlogin	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	Non

Procédures stockées du système T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_droprole	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	Non
sp_droprolemember	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	Non
sp_dropserver	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	Non
sp_enum_oledb_providers	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
sp_execute	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_execute_postgresql(CREATE, ALTER, DROP)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
sp_executesql	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_fkeys	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_getapplock	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_helpdb	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Procédures stockées du système T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_helpdbfixedrole	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	Non
sp_helplinkedsrvlogin	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	Non
sp_helprole	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_helprolemember	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_helpsrvrolemember	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non
sp_helpuser	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_linkedservers	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	Non
sp_oledb_ro_usrname	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_pkeys	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_prefix	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non

Procédures stockées du système T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_prepare	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sp_procedure_params_100_managed	O	O	O	O	O	O	O	O	N	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
sp_releaseapplock	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sp_rename	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	Non
sp_renamedb	O	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
sp_rename (pour l’object_type ’DATABASE’)	O	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
sp_reset_connection	O	O	O	O	O	N	N	N	N	O	O	O	N	N	N	N	N	N	N	O	O	O	N	N	N	N	N	N	N	N	N	Non
sp_serveroption (option connect_timeout)	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	O	O	O	O	O	O	O	N	N	N	N	N	Non
sp_set_session_context	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	Non

Procédures stockées du système T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	201.0
sp_special_columns	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_sproc_columns	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_sproc_columns_100	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_statistics	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_statistics_100	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_stored_procedures	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_table_privileges	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_tablecollations_100	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_tables	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_testlinkedserver	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	C	N	N	N	N	N	Non

Procédures stockées du système T-SQL	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sp_unprepare	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sp_updateextendedproperty	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
sp_who	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
xp_qv	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Propriétés T-SQL prises en charge au niveau de la fonction système CONNECTIONPROPERTY

Dans le tableau suivant, vous trouverez les fonctionnalités T-SQL prises en charge au niveau des fonctions système CONNECTIONPROPERTY par les différentes versions de Babelfish.

Propriétés T-SQL prises en charge au niveau de la fonction système CONNECTIONPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
auth_scheme	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
client_net_address	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
local_net_address																																	

Propriétés T-SQL prises en charge au niveau de la fonction système CONNECTIONPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
local_tcp_port	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
net_transport	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
protocol_type	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
Physical_net_transport	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Propriétés T-SQL prises en charge au niveau de la fonction système OBJECTPROPERTY

Dans le tableau suivant, vous trouverez les fonctionnalités T-SQL prises en charge au niveau des fonctions système OBJECTPROPERTY par les différentes versions de Babelfish.

Propriétés T-SQL prises en charge au niveau de la fonction système OBJECTPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
IsInlineFunction																																

Propriétés T-SQL prises en charge au niveau de la fonction système OBJECTPROPERTY	5.5	5.4	4.4	4.3	3.3	3.2	2.2	2.1.0
	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui
IsScalarFunction	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui
IsTableFunction	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui

Propriétés T-SQL prises en charge au niveau de la fonction SERVERPROPERTY

Dans le tableau suivant, vous trouverez les fonctionnalités T-SQL prises en charge au niveau des fonctions système SERVERPROPERTY par les différentes versions de Babelfish.

Propriétés T-SQL prises en charge au niveau de la fonction SERVERPROPERTY	5.5	5.4	4.4	4.3	3.3	3.2	2.2	2.1.0
Babelfish	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui
Classement	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui
CollationID	Oui	Oui	Oui	Oui	Oui	Oui	Oui	Oui

Propriétés T-SQL prises en charge au niveau de la fonction SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
Edition	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
EditionID	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
EngineEdition	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
InstanceName	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Non
IsAdvancedAnalyticsInstalled	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
IsBigDataCluster	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
IsFullTextInstalled	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
IsIntegratedSecurityOnly	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
IsLocalDB	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui

Propriétés T-SQL prises en charge au niveau de la fonction SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	21.0
IsPolyBaseInstalled	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
IsSingleUser	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
IsXTPSupported	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
Japanese_CI_AI	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
Japanese_CI_AS	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
Japanese_CS_AS	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
LicenseType	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
MachineName	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Non
ProductLevel	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	Non

Propriétés T-SQL prises en charge au niveau de la fonction SERVERPROPERTY	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
ProductMajorVersion	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
ProductMinorVersion	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
ProductUpdateLevel	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	Non
ProductVersion	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
ServerName	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Vues SQL Server prises en charge par Babelfish

Dans le tableau suivant, vous trouverez les vues SQL Server prises en charge par les différentes versions de Babelfish.

Vues SQL Server prises en charge par Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
information_schema .key_column_usage	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	C	C	C	C	C	C	N	N	N	N	N	N	Non

Vues SQL Server prises en charge par Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	1.0
information_schema .routines	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	Non
information_schema .schemata	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	Non
information_schema .sequences	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	Non
sys.all_columns	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.all_objects	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.all_parameters	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	Non
sys.all_sql_modules	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.all_views	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.columns	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui

Vues SQL Server prises en charge par Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2 1.0
sys.configurations	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sys.data_spaces	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sys.database_files	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sys.database_mirroring	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sys.database_principals	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sys.database_role_members	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sys.databases	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sys.destination_data_spaces	C	C	C	C	C	C	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
sys.dm_exec_connections	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui

Vues SQL Server prises en charge par Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sys.dm_exec_sessions	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.dm_hard_database_replica_states	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.dm_os_host_info	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.dm_os_sys_info	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
sys.endpoints	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.extended_properties	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	O	O	O	O	O	O	O	N	N	N	N	N	N	Non
sys.indexes	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.partitions	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
sys.partition_functions	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non

Vues SQL Server prises en charge par Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sys.partition_parameters	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
sys.partition_range_values	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
sys.partition_schemes	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
sys.schemas	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.server_principals	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.server_role_members	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Non
sys.sql_modules	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.sysconfigures	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui
sys.syscurconfigs	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	Oui

Vues SQL Server prises en charge par Babelfish	5	5	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	1.0
sys.syslogins	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
sys.sysprocesses	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sys.sysusers	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C	C	N	N	N	N	N	N	Non
sys.table_types	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sys.tables	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sys.types	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	N	Non
sys.xml_schema_collections	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
syslanguages	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	Oui
sysobjects.crdate	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	N	N	Non

Fonctionnalités non prises en charge dans Babelfish

Dans le tableau et les listes suivants, vous trouverez des fonctionnalités qui ne sont actuellement pas prises en charge par Babelfish. Les mises à jour de Babelfish sont incluses dans les versions d'Aurora PostgreSQL. Pour plus d'informations, consultez Notes de mise à jour d'Aurora PostgreSQL.

Rubriques

Fonctionnalité non prise en charge actuellement

Le tableau contient des informations sur certaines fonctionnalités qui ne sont pas prises en charge pour le moment.

Fonctionnalité ou syntaxe	Description
Modules d'assemblage et routines SQL Common Language Runtime (CLR)	La fonctionnalité associée aux modules d'assemblage et aux routines CLR n'est pas prise en charge.
Attributs des colonnes	ROWGUIDCOL, SPARSE, FILESTREAM et MASKED ne sont pas pris en charge.
Bases de données contenues	Les bases de données contenues dont les identifiants sont authentifiés au niveau de la base de données plutôt qu'au niveau du serveur ne sont pas prises en charge.

Fonctionnalité ou syntaxe	Description
DDL couvrant plusieurs bases de données	L’exécution d’instructions DDL qui traitent des objets ou y font référence dans plusieurs bases de données n’est pas encore prise en charge.
Curseurs (actualisables)	Les curseurs actualisables ne sont pas pris en charge.
Curseurs (globaux)	Les curseurs GLOBAL ne sont pas pris en charge.
Curseur (comportements de récupération)	Les comportements de récupération de curseur suivants ne sont pas pris en charge : FETCH PRIOR, FIRST, LAST, ABSOLUTE et RELATIVE
Paramètres de sortie de type curseur	Les variables et paramètres de type curseur ne sont pas pris en charge pour les paramètres de sortie (une erreur est générée).
Options de curseur	SCROLL, KEYSET, DYNAMIC, FAST_FORWARD, SCROLL_LOCKS, OPTIMISTIC, TYPE_WARNING et FOR UPDATE
Chiffrement des données	Le chiffrement des données n’est pas pris en charge.
Applications de la couche Données (DAC)	Les opérations d’importation et d’exportation d’applications de la couche Données (DAC) avec des fichiers de package DAC (.dacpac) ou de sauvegarde DAC (.bacpac) ne sont pas prises en charge.
Commandes DBCC	Les commandes DBCC (Microsoft SQL Server Database Console) ne sont pas prises en charge. DBCC CHECKIDENT est pris en charge dans Babelfish 3.4.0 et versions ultérieures.
DROP IF EXISTS	Cette syntaxe n’est pas prise en charge pour les objets USER et SCHEMA. Elle est prise en charge pour les objets TABLE, VIEW, PROCEDURE, FUNCTION et DATABASE.
Chiffrement	Les fonctions et instructions intégrées ne prennent pas en charge le chiffrement.

Fonctionnalité ou syntaxe	Description
Connexions ENCRYPT_CLIENT_CERT	Les connexions par certificat client ne sont pas prises en charge.
Instruction EXECUTE AS	Cette instruction n’est pas prise en charge.
Clause EXECUTE AS SELF	Cette clause n’est pas prise en charge dans les fonctions, procédures ou déclencheurs.
Clause EXECUTE AS USER	Cette clause n’est pas prise en charge dans les fonctions, procédures ou déclencheurs.
Contraintes de clé étrangère faisant référence au nom de base de données	Les contraintes de clé étrangère qui font référence au nom de base de données ne sont pas prises en charge.
FORMAT	Les types définis par l’utilisateur ne sont pas pris en charge.
Déclarations de fonctions avec plus de 100 paramètres	Les déclarations de fonctions contenant plus de 100 paramètres ne sont pas prises en charge.
Appels de fonction qui incluent DEFAULT comme valeur de paramètre	DEFAULT n’est pas une valeur de paramètre prise en charge pour un appel de fonction. DEFAULT en tant que valeur de paramètre pour un appel de fonction est pris en charge à partir de Babelfish 3.4.0 et versions ultérieures.
Fonctions définies en externe	Les fonctions externes, y compris les fonctions SQL CLR, ne sont pas prises en charge.
Tables temporaires globales (tables dont le nom commence par ##)	Les tables temporaires globales ne sont pas prises en charge.
Fonctionnalités graphiques	Aucune des fonctionnalités graphiques de SQL n’est prise en charge.

Fonctionnalité ou syntaxe	Description
Procédures stockées étendues générales	Les procédures stockées du système qui fournissent une interface entre une instance de SQL Server et des programmes externes pour diverses activités de maintenance ne sont pas prises en charge. Cela inclut `xp_cmdshell` et les autres procédures stockées du système. Pour plus d'informations, consultez Procédures stockées étendues générales.
Identifiants (variables ou paramètres) comportant plusieurs caractères @ de début	Les identifiants qui commencent par plusieurs caractères @ ne sont pas pris en charge.
Identifiants, noms de table ou de colonne contenant des caractères @ ou]]	Les noms de table ou de colonne contenant un signe @ ou des crochets ne sont pas pris en charge.
Index en ligne	Les index en ligne ne sont pas pris en charge.
Appel d'une procédure dont le nom figure dans une variable	L'utilisation d'une variable comme nom de procédure n'est pas prise en charge.
Vues matérialisées	Les vues matérialisées ne sont pas prises en charge.
Clause NOT FOR REPLICATION	Cette syntaxe est acceptée et ignorée.
Fonctions d'échappement ODBC	Les fonctions d'échappement ODBC ne sont pas prises en charge.
Appels de procédure incluant DEFAULT comme valeur de paramètre	DEFAULT n'est pas une valeur de paramètre prise en charge. DEFAULT en tant que valeur de paramètre pour un appel de fonction est pris en charge à partir de Babelfish 3.4.0 et versions ultérieures.
Déclarations de procédure comportant plus de 100 paramètres	Les déclarations comportant plus de 100 paramètres ne sont pas prises en charge.

Fonctionnalité ou syntaxe	Description
Procédures définies en externe	Les procédures définies en externe, y compris les procédures SQL CLR, ne sont pas prises en charge.
Gestion des versions des procédures	La gestion des versions des procédures n'est pas prise en charge.
Procédures WITH RECOMPILE	WITH RECOMPILE (lorsqu'il est utilisé conjointement avec les instructions DECLARE et EXECUTE) n'est pas pris en charge.
Références à des objets distants	L'exécution de procédures stockées sur les serveurs liés Babelfish n'est pas prise en charge. Les noms d'objets en quatre parties ne fonctionnent que pour la lecture et ne fonctionnent pas pour modifier la table distante. UPDATE peut référencer une table distante dans la clause FROM sans la modifier. Pour plus d'informations, consultez Babelfish prend en charge les serveurs liés.
Sécurité de niveau ligne	La sécurité de niveau ligne avec CREATE SECURITY POLICY et les fonctions de valeur de table en ligne n'est pas prise en charge.
Fonctionnalité Service Broker	La fonctionnalité Service Broker n'est pas prise en charge.
SESSIONPROPERTY	Propriétés non prises en charge : ANSI_NULLS, ANSI_PADDING, ANSI_WARNINGS, ARITHABORT, CONCAT_NULL_YIELDS_NULL et NUMERIC_ROUNDABORT
SET LANGUAGE	Cette syntaxe n'est prise en charge qu'avec la valeur `english` ou `us_english` .
SP_CONFIGURE	Cette procédure stockée fournie par le système n'est pas prise en charge.
Mot-clé SQL `SPARSE`	Le mot-clé SPARSE est accepté et ignoré.

Fonctionnalité ou syntaxe	Description
Syntaxe du constructeur de valeurs de table (clause FROM)	La syntaxe non prise en charge concerne une table dérivée construite avec la clause FROM.
Tables temporelles	Les tables temporelles ne sont pas prises en charge.
Les procédures temporaires ne sont pas automatiquement supprimées	Cette fonctionnalité n'est pas prise en charge.
Déclencheurs définis en externe	Aucun de ces déclencheurs n'est pris en charge, pas même SQL Common Language Runtime (CLR).

Paramètres non pris en charge

Les paramètres suivants ne sont pas pris en charge :

- SET ANSI_NULL_DFLT_OFF ON
- SET ANSI_NULL_DFLT_ON OFF
- SET ANSI_PADDING OFF
- SET ANSI_WARNINGS OFF
- SET ARITHABORT OFF
- SET ARITHIGNORE ON
- SET CURSOR_CLOSE_ON_COMMIT ON
- SET NUMERIC_ROUNDABORT ON
- SET PARSEONLY ON (la commande ne fonctionne pas comme prévu)
- SET FMTONLY ON (la commande ne fonctionne pas comme prévu. Elle supprime uniquement l'exécution des instructions SELECT, mais pas les autres.)

Commandes non prises en charge

Certaines fonctionnalités des commandes suivantes ne sont pas prises en charge :

- ADD SIGNATURE

- ALTER DATABASE, ALTER DATABASE SET
- BACKUP/RESTORE DATABASE/LOG
- BACPAC et DACPAC FILES RESTORE
- CREATE, ALTER, DROP AUTHORIZATION. ALTER AUTHORIZATION est pris en charge pour les objets de base de données.
- CREATE, ALTER, DROP AVAILABILITY GROUP
- CREATE, ALTER, DROP BROKER PRIORITY
- CREATE, ALTER, DROP COLUMN ENCRYPTION KEY
- CREATE, ALTER, DROP DATABASE ENCRYPTION KEY
- CREATE, ALTER, DROP, BACKUP CERTIFICATE
- CREATE AGGREGATE
- CREATE CONTRACT
- CHECKPOINT

Noms de colonnes ou attributs non pris en charge

Les noms de colonne suivants ne sont pas pris en charge :

- $IDENTITY
- $ROWGUID
- IDENTITYCOL

Types de données non pris en charge

Les types de données suivants ne sont pas pris en charge :

- HIERARCHYID

Types d’objets non pris en charge

Les types d’objets suivants ne sont pas pris en charge :

- COLUMN MASTER KEY
- CREATE, ALTER EXTERNAL DATA SOURCE
- CREATE, ALTER, DROP DATABASE AUDIT SPECIFICATION

- CREATE, ALTER, DROP EXTERNAL LIBRARY
- CREATE, ALTER, DROP SERVER AUDIT
- CREATE, ALTER, DROP SERVER AUDIT SPECIFICATION
- CRÉER, MODIFIER, SUPPRIMER UNE CLÉ OPEN/CLOSE SYMÉTRIQUE
- CREATE, DROP DEFAULT
- CREDENTIAL
- CRYPTOGRAPHIC PROVIDER
- DIAGNOSTIC SESSION
- Vues indexées
- SERVICE MASTER KEY
- SYNONYM

Fonctions non prises en charge

Les fonctions intégrées suivantes ne sont pas prises en charge :

Fonctions d’agrégation

- APPROX_COUNT_DISTINCT
- CHECKSUM_AGG
- GROUPING_ID
- STRING_AGG avec la clause WITHIN GROUP

Fonctions cryptographiques

- Fonction CERTENCODED
- Fonction CERTID
- Fonction CERTPROPERTY

Fonctions de métadonnées

- COLUMNPROPERTY
- TYPEPROPERTY
- Fonction SERVERPROPERTY — les propriétés suivantes ne sont pas prises en charge :

- BuildClrVersion
- ComparisonStyle
- ComputerNamePhysicalNetBIOS
- HadrManagerStatus
- InstanceDefaultDataPath
- InstanceDefaultLogPath
- IsClustered
- IsHadrEnabled
- LCID
- NumLicenses
- ProcessID
- ProductBuild
- ProductBuildType
- ProductUpdateReference
- ResourceLastUpdateDateTime
- ResourceVersion
- ServerName
- SqlCharSet
- SqlCharSetName
- SqlSortOrder
- SqlSortOrderName
- FilestreamShareName
- FilestreamConfiguredLevel
- FilestreamEffectiveLevel

Security functions

- CERTPRIVATEKEY
- LOGINPROPERTY

Déclarations, opérateurs, autres fonctions

- Fonction EVENTDATA
- GET_TRANSMISSION_STATUS
- OPENXML

Syntaxe non prise en charge

La syntaxe suivante n’est pas prise en charge :

- ALTER DATABASE
- ALTER DATABASE SCOPED CONFIGURATION
- ALTER DATABASE SCOPED CREDENTIAL
- ALTER DATABASE SET HADR
- ALTER INDEX
- ALTER PARTITION FUNCTION
- ALTER PARTITION SCHEME
- ALTER SCHEMA
- ALTER SERVER CONFIGURATION
- Clause ALTER BACKUP/RESTORE SERVICE, SERVICE MASTER KEY
- BEGIN CONVERSATION TIMER
- BEGIN DISTRIBUTED TRANSACTION
- BEGIN DIALOG CONVERSATION
- BULK INSERT
- CREATE COLUMNSTORE INDEX
- CREATE EXTERNAL FILE FORMAT
- CREATE EXTERNAL TABLE
- CREATE, ALTER, DROP APPLICATION ROLE
- CREATE, ALTER, DROP ASSEMBLY
- CREATE, ALTER, DROP ASYMMETRIC KEY
- CREATE, ALTER, DROP CREDENTIAL
- CREATE, ALTER, DROP CRYPTOGRAPHIC PROVIDER

- CREATE, ALTER, DROP ENDPOINT
- CREATE, ALTER, DROP EVENT SESSION
- CREATE, ALTER, DROP EXTERNAL LANGUAGE
- CREATE, ALTER, DROP EXTERNAL RESOURCE POOL
- CREATE, ALTER, DROP FULLTEXT CATALOG
- CREATE, ALTER, DROP FULLTEXT INDEX
- CREATE, ALTER, DROP FULLTEXT STOPLIST
- CREATE, ALTER, DROP MESSAGE TYPE
- CRÉATION, MODIFICATION, SUPPRESSION, CLÉ OPEN/CLOSE, BACKUP/RESTORE PRINCIPALE
- CREATE, ALTER, DROP QUEUE
- CREATE, ALTER, DROP RESOURCE GOVERNOR
- CREATE, ALTER, DROP RESOURCE POOL
- CREATE, ALTER, DROP ROUTE
- CREATE, ALTER, DROP SEARCH PROPERTY LIST
- CREATE, ALTER, DROP SECURITY POLICY
- CREATE, ALTER, DROP SELECTIVE XML INDEX clause
- CREATE, ALTER, DROP SERVICE
- CREATE, ALTER, DROP SPATIAL INDEX
- CREATE, ALTER, DROP TYPE
- CREATE, ALTER, DROP XML INDEX
- CREATE, ALTER, DROP XML SCHEMA COLLECTION
- CREATE/DROP RULE
- CREATE, DROP WORKLOAD CLASSIFIER
- CREATE, ALTER, DROP WORKLOAD GROUP
- ALTER TRIGGER
- Clause CREATE TABLE... GRANT
- Clause CREATE TABLE... IDENTITY
- CREATE USER : cette syntaxe n’est pas prise en charge. L’instruction CREATE USER de PostgreSQL ne crée pas d’utilisateur équivalent à celui créé avec la syntaxe CREATE USER de SQL Server. Pour plus d’informations, consultez Différences T-SQL dans Babelfish.

- REJETER
- END, MOVE CONVERSATION
- EXECUTE with AS LOGIN or AT option
- GET CONVERSATION GROUP
- GROUP BY ALL clause
- GROUP BY CUBE clause
- GROUP BY ROLLUP clause
- INSERT... DEFAULT VALUES
- MERGE
- READTEXT
- REVERT
- SELECT TOP... WITH TIES
- SELECT... FOR BROWSE
- SELECT... FOR XML AUTO
- SELECT... FOR XML EXPLICIT
- SELECT... FOR XML PATH
- SEND
- SET DATEFORMAT
- SET DEADLOCK_PRIORITY
- SET FMTONLY
- SET FORCEPLAN
- SET NUMERIC_ROUNDABORT ON
- SET OFFSETS
- SET REMOTE_PROC_TRANSACTIONS
- SET SHOWPLAN_TEXT
- SET SHOWPLAN_XML
- SET STATISTICS
- SET STATISTICS PROFILE
- SET STATISTICS TIME
- SET STATISTICS XML

- SHUTDOWN statement
- UPDATE STATISTICS
- UPDATETEXT
- Using EXECUTE to call a SQL function
- VIEW... CHECK OPTION clause
- VIEW... VIEW_METADATA clause
- WAITFOR DELAY
- WAITFOR TIME
- WAITFOR, RECEIVE
- WITH XMLNAMESPACES construct
- WRITETEXT
- XPATH expressions

Utilisation des procédures Babelfish pour Aurora PostgreSQL

Présentation

Vous pouvez utiliser la procédure suivante pour les instances de base de données Amazon RDS exécutant Babelfish pour Aurora PostgreSQL afin d'améliorer les performances des requêtes :

- sp_babelfish_volatility
- sp_execute_postgresql

sp_babelfish_volatility

La volatilité des fonctions PostgreSQL aide l'optimiseur à mieux exécuter les requêtes, ce qui, lorsqu'il est utilisé dans des parties de certaines clauses, a un impact significatif sur les performances des requêtes.

Syntaxe

```
sp_babelfish_volatility 'function_name', 'volatility'
```

Arguments

function_name (facultatif)

Vous pouvez spécifier la valeur de cet argument avec un nom en deux parties comme `schema_name.function_name` ou uniquement `function_name`. Si vous spécifiez uniquement `function_name`, le nom du schéma est le schéma par défaut pour l'utilisateur actuel.

volatility (facultatif)

Les valeurs PostgreSQL valides de volatilité sont `stable`, `volatile` ou `immutable`. Pour plus d'informations, consultez https://www.postgresql.org/docs/current/xfunc-volatility.html.

> **Note**
>
> Quand `sp_babelfish_volatility` est appelée avec `function_name` qui possède plusieurs définitions, elle génère une erreur.

Jeu de résultats

Si les paramètres ne sont pas mentionnés, le jeu de résultats s'affiche sous les colonnes suivantes :`schemaname`, `functionname`, `volatility`.

Notes d’utilisation

La volatilité des fonctions PostgreSQL aide l'optimiseur à mieux exécuter les requêtes, ce qui, lorsqu'il est utilisé dans des parties de certaines clauses, a un impact significatif sur les performances des requêtes.

Exemples

Les exemples suivants montrent comment créer des fonctions simples et expliquent ensuite comment utiliser `sp_babelfish_volatility` sur ces fonctions à l'aide de différentes méthodes.

```
1> create function f1() returns int as begin return 0 end
2> go
```

```
1> create schema test_schema
2> go
```

```
1> create function test_schema.f1() returns int as begin return 0 end
2> go
```

L'exemple suivant montre la volatilité des fonctions :

```
1> exec sp_babelfish_volatility
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           volatile
test_schema f1           volatile
```

L'exemple suivant montre comment modifier la volatilité des fonctions :

```
1> exec sp_babelfish_volatility 'f1','stable'
2> go
1> exec sp_babelfish_volatility 'test_schema.f1','immutable'
```

```
2> go
```

Lorsque vous spécifiez uniquement function_name, cela affiche le nom du schéma, le nom de la fonction et la volatilité de cette fonction. L'exemple suivant affiche la volatilité des fonctions après la modification des valeurs :

```
1> exec sp_babelfish_volatility 'test_schema.f1'
2> go

schemaname  functionname volatility
----------- ------------ ----------
test_schema f1           immutable
```

```
1> exec sp_babelfish_volatility 'f1'
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           stable
```

Lorsque vous ne spécifiez aucun argument, cela affiche la liste des fonctions (nom du schéma, nom de la fonction, volatilité des fonctions) présentes dans la base de données actuelle :

```
1> exec sp_babelfish_volatility
2> go

schemaname  functionname volatility
----------- ------------ ----------
dbo         f1           stable
test_schema f1           immutable
```

sp_execute_postgresql

Vous pouvez exécuter des instructions PostgreSQL du point de terminaison T-SQL. Cela simplifie vos applications, car vous n'avez pas besoin de quitter le port T-SQL pour exécuter ces instructions.

Syntaxe

```
sp_execute_postgresql [ @stmt = ] statement
```

Arguments

instruction [@stmt]

L'argument est de type varchar. Cet argument accepte les instructions en langage PG.

> **Note**
>
> Vous ne pouvez passer qu'une seule instruction en langage PG comme argument, sinon l'erreur suivante se produira.

```
1>exec sp_execute_postgresql 'create extension pg_stat_statements; drop extension
 pg_stat_statements'
2>go
```

```

Msg 33557097, Level 16, State 1, Server BABELFISH, Line 1
expected 1 statement but got 2 statements after parsing
```

Notes d'utilisation

CREATE EXTENSION

Crée et charge une nouvelle extension dans la base de données actuelle.

```
1>EXEC sp_execute_postgresql 'create extension [ IF NOT EXISTS ] <extension name>
 [ WITH ] [SCHEMA schema_name] [VERSION version]';
2>go
```

L'exemple suivant montre comment créer une extension :

```
1>EXEC sp_execute_postgresql 'create extension pg_stat_statements with schema sys
 version "1.10"';
2>go
```

Utilisez la commande suivante pour accéder aux objets de l'extension :

```
1>select * from pg_stat_statements;
2>go
```

> **Note**
>
> Si le nom du schéma n'est pas fourni explicitement lors de la création de l'extension, les extensions sont installées par défaut dans le schéma public. Vous devez fournir le qualificateur de schéma pour accéder aux objets de l'extension, comme mentionné ci-dessous :

```
1>select * from [public].pg_stat_statements;
2>go
```

Extensions prises en charge

Les extensions suivantes disponibles avec Aurora PostgreSQL sont compatibles avec Babelfish.

- `pg_stat_statements`
- `tds_fdw`
- `fuzzystrmatch`

Limites

- Les utilisateurs doivent posséder le rôle sysadmin sur T-SQL et rds_superuser sur postgres pour installer les extensions.
- Les extensions ne peuvent pas être installées dans des schémas créés par l'utilisateur, ni dans des schémas dbo et guest pour les bases de données master, tempdb et msdb.
- L'option CASCADE n'est pas prise en charge.

ALTER EXTENSION

Vous pouvez effectuer une mise à niveau vers une nouvelle version de l'extension à l'aide de l'instruction ALTER.

```
1>EXEC sp_execute_postgresql 'alter extension <extension name> UPDATE TO
 <new_version>';
2>go
```

Limites

- Vous pouvez mettre à niveau la version de votre extension uniquement à l'aide de l'instruction ALTER Extension. Les autres opérations ne sont pas prises en charge.

DROP EXTENSION

Supprime l'extension spécifiée. Vous pouvez également utiliser les options `if exists` ou `restrict` pour supprimer l'extension.

```
1>EXEC sp_execute_postgresql 'drop extension <extension name>';
2>go
```

Limites

- L'option CASCADE n'est pas prise en charge.

Babelfish supporte les méthodes de type de données XML

Depuis la version 5.4.0, Babelfish supporte désormais les procédures stockées sp_xml_preparedocument et sp_xml_removedocument, la fonction rowset OPENXML () et la méthode de type de données XML .VALUE (). Grâce à ces fonctions et procédures, les requêtes sur les données XML deviennent beaucoup plus faciles.

Comprendre les procédures et méthodes XML

- sp_xml_preparedocument — La procédure sp_xml_preparedocument analyse un texte XML fourni en entrée et renvoie un descripteur à ce document. Ce descripteur est valide pendant la session ou jusqu'à ce qu'il soit supprimé par sp_xml_removedocument.
- sp_xml_removedocument — La procédure sp_xml_removedocument invalide le descripteur créé par la procédure sp_xml_preparedocument.
- OPENXML () — OPENXML fournit une vue en ligne d'un document XML. Comme OPENXML est un fournisseur d'ensembles de lignes et qu'il renvoie un ensemble de lignes, nous pouvons utiliser OPENXML dans la clause FROM comme nous pouvons utiliser n'importe quelle autre fonction de table, de vue ou de valeur de table.
- VALUE () — La méthode de type de données XML VALUE () est utilisée pour extraire une valeur d'une instance XML stockée dans une colonne, un paramètre ou une variable de type XML.

Limitations des procédures et méthodes XML de Babelfish

- Babelfish ne supporte que la syntaxe XPATH 1.0 pour le deuxième argument (c'est-à-dire ROWPATTERN) de OPENXML ().
- Les méta-propriétés et le drapeau 8 ne sont actuellement pas pris en charge dans OPENXML ().
- Babelfish ne supporte que la syntaxe XPATH 1.0 pour le premier argument (c'est-à-dire XQuery) de la méthode de type de données VALUE ().

Performance et mise à l’échelle d’Amazon Aurora PostgreSQL

La section suivante explique la gestion des performances et de la mise à l’échelle d’un cluster de bases de données Amazon Aurora PostgreSQL. Elle inclue également des informations relatives à d’autres tâches de maintenance.

Rubriques

Dimensionnement des instances de base de données Aurora PostgreSQL

Vous pouvez dimensionner les instances de base de données Aurora PostgreSQL de deux façons : le dimensionnement d'instance et le dimensionnement en lecture. Pour plus d'informations sur le dimensionnement en lecture, consultez Dimensionnement en lecture.

Vous pouvez mettre à l'échelle votre cluster de bases de données Aurora PostgreSQL DB en modifiant la classe d'instance de base de données pour chaque instance du cluster de bases de données. Aurora PostgreSQL prend en charge plusieurs classes d'instance de base de données optimisées pour Aurora. N'utilisez pas les classes d'instance db.t2 ou db.t3 pour des clusters Aurora d'une taille supérieure à 40 téraoctets (To).

> **Note**
>
> Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à la production. Pour plus de détails sur les classes d'instance T, consultez Types de classes d'instance de base de données.

La mise à l'échelle n'est pas instantanée. La modification apportée à une autre classe d'instance de base de données peut prendre 15 minutes ou plus. Si vous utilisez cette approche pour modifier la classe d'instance de base de données, vous appliquez le changement lors de la prochaine fenêtre de maintenance planifiée (plutôt qu'immédiatement) pour éviter d'affecter les utilisateurs.

Au lieu de modifier directement la classe d'instance de base de données, vous pouvez réduire la durée d'indisponibilité en utilisant les fonctions de haute disponibilité d'Amazon Aurora. Tout d'abord,

ajoutez un réplica Aurora à votre cluster. Lorsque vous créez le réplica, choisissez la taille de classe d'instance de base de données que vous souhaitez utiliser pour votre cluster. Lorsque le réplica Aurora est synchronisé avec le cluster, effectuez un basculement vers le réplica nouvellement ajouté. Pour en savoir plus, consultez Réplicas Aurora et Basculement rapide avec Amazon Aurora PostgreSQL.

Pour obtenir les spécifications détaillées des classes d'instance de base de données prises en charge par Aurora PostgreSQL, consultez Moteurs de base de données pris en charge pour les classes d'instance de base de données.

Nombre maximal de connexions à une instance de base de données Aurora PostgreSQL

Un cluster de bases de données Aurora PostgreSQL alloue des ressources en fonction de la classe d'instance de base de données et de sa mémoire disponible. Chaque connexion au cluster de bases de données consomme des quantités incrémentielles de ces ressources, telles que la mémoire et le processeur. La mémoire consommée par connexion varie en fonction du type de requête, du nombre de requêtes et de l'utilisation éventuelle de tables temporaires. Même une connexion inactive consomme de la mémoire et du processeur. En effet, lorsque des requêtes sont exécutées sur une connexion, une plus grande quantité de mémoire est allouée pour chaque requête et elle n'est pas complètement libérée, même lorsque le traitement s'arrête. Nous vous recommandons donc de vous assurer que vos applications ne maintiennent pas des connexions inactives : chacune d'entre elles gaspille des ressources et a un impact négatif sur les performances. Pour plus d'informations, consultez Resources consumed by idle PostgreSQL connections (Ressources consommées par les connexions PostgreSQL inactives).

Le nombre maximal de connexions autorisées par une instance de base de données Aurora PostgreSQL est déterminé par la valeur de paramètre `max_connections` spécifiée dans le groupe de paramètres de cette instance de base de données. Le cadre idéal pour le paramètre `max_connections` prend en charge toutes les connexions client dont votre application a besoin, sans excès de connexions inutilisées, plus au moins 3 connexions supplémentaires pour prendre en charge l'automatisation AWS. Avant de modifier le paramètre `max_connections`, nous vous recommandons de prendre en compte les points suivants :

- Si la valeur `max_connections` est trop faible, l'instance de base de données Aurora PostgreSQL peut ne pas disposer de connexions suffisantes lorsque les clients tentent de se connecter. Si c'est le cas, les tentatives de connexion à l'aide de `psql` génèrent des messages d'erreur tels que les suivants :

```
psql: FATAL: remaining connection slots are reserved for non-replication superuser
 connections
```

- Si la valeur `max_connections` dépasse le nombre de connexions réellement nécessaires, les connexions inutilisées peuvent entraîner une dégradation des performances.

La valeur par défaut de `max_connections` est dérivée de la fonction Aurora PostgreSQL `LEAST` suivante :

`LEAST({DBInstanceClassMemory/9531392},5000)`.

Si vous souhaitez modifier la valeur de `max_connections`, vous devez créer un groupe de paramètres de base de données personnalisé et y modifier sa valeur. Après avoir appliqué votre groupe de paramètres de base de données personnalisé à votre cluster, veillez à redémarrer l'instance principale pour que la nouvelle valeur prenne effet. Pour plus d'informations, consultez Paramètres Amazon Aurora PostgreSQL. et Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

> **Tip**
>
> Si vos applications ouvrent et ferment fréquemment des connexions, ou si elles ont ouvert un grand nombre de connexions de longue durée, nous vous recommandons d'utiliser Proxy Amazon RDS. RDS Proxy est un proxy de base de données entièrement géré et hautement disponible qui utilise le regroupement de connexions pour partager les connexions de base de données de manière sécurisée et efficace. Pour en savoir plus sur RDS Proxy, consultez Proxy Amazon RDS pour Aurora.

Pour obtenir plus de détails sur la façon dont les instances Aurora Serverless v2 gèrent ce paramètre, consultez Nombre maximal de connexions pour Aurora Serverless v2.

Limites de stockage temporaires pour Aurora PostgreSQL

Aurora PostgreSQL stocke les tables et les index dans le sous-système de stockage Aurora. Aurora PostgreSQL utilise un stockage temporaire séparé pour les fichiers temporaires non persistants. Il s'agit notamment des fichiers qui sont utilisés à des fins telles que le tri de grands jeux de données pendant le traitement des requêtes ou les opérations de génération d'index. Pour en savoir plus,

consultez l'article Comment puis-je résoudre les problèmes de stockage local dans les instances compatibles avec Aurora PostgreSQL ?

Ces volumes de stockage locaux sont sauvegardés par Amazon Elastic Block Store et peuvent être étendus en utilisant une classe d'instance de base de données plus grande. Pour plus d'informations sur le stockage, consultez Stockage Amazon Aurora. Vous pouvez également augmenter votre espace de stockage local pour les objets temporaires en utilisant un type d'instance compatible NVMe et des objets temporaires compatibles Aurora Optimized Reads. Pour plus d'informations, consultez Amélioration des performances des requêtes pour Aurora PostgreSQL avec Aurora Optimized Reads.

> **Note**
>
> Vous verrez peut-être des événements `storage-optimization` lors de la mise à l'échelle d'instances de base de données, de db.r5.2xlarge à db.r5.4xlarge par exemple.

Le tableau suivant indique la quantité maximale de stockage temporaire disponible pour chaque classe d'instance de base de données Aurora PostgreSQL. Pour plus d'informations sur la prise en charge d'une classe d'instance de base de données pour Aurora, consultez Classes d'instance de base de données Amazon Aurora.

Classe d'instance de base de données	Stockage temporaire maximal disponible (Gio)
db.x2g.16xlarge	1 829
db.x2g.12xlarge	1 606
db.x2g.8xlarge	1 071
db.x2g.4xlarge	535
db.x2g.2xlarge	268
db.x2g.xlarge	134
db.x2g.large	67
db.r8g.48xlarge	3 072

Classe d’instance de base de données	Stockage temporaire maximal disponible (Gio)
db.r8g.24xlarge	1 536
db.r8g.16xlarge	998
db.r8g.12xlarge	749
db.r8g.8xlarge	499
db.r8g.4xlarge	250
db.r8g.2xlarge	125
db.r8g.xlarge	63
db.r8g.large	31
db.r7g.16xlarge	1 008
db.r7g.12xlarge	756
db.r7g.8xlarge	504
db.r7g.4xlarge	252
db.r7g.2xlarge	126
db.r7g.xlarge	63
db.r7g.large	32
db.r7i.48xlarge	3 072
db.r7i.24xlarge	1 500
db.r7i.16xlarge	1 008
db.r7i.12xlarge	748
db.r7i.8xlarge	504

Classe d’instance de base de données	Stockage temporaire maximal disponible (Gio)
db.r7i.4xlarge	249
db.r7i.2xlarge	124
db.r7i.xlarge	62
db.r7i.large	31
db.r6g.16xlarge	1 008
db.r6g.12xlarge	756
db.r6g.8xlarge	504
db.r6g.4xlarge	252
db.r6g.2xlarge	126
db.r6g.xlarge	63
db.r6g.large	32
db.r6i.32xlarge	1 829
db.r6i.24xlarge	1 500
db.r6i.16xlarge	1 008
db.r6i.12xlarge	748
db.r6i.8xlarge	504
db.r6i.4xlarge	249
db.r6i.2xlarge	124
db.r6i.xlarge	62
db.r6i.large	31

Classe d’instance de base de données	Stockage temporaire maximal disponible (Gio)
db.r5.24xlarge	1 500
db.r5.16xlarge	1 008
db.r5.12xlarge	748
db.r5.8xlarge	504
db.r5.4xlarge	249
db.r5.2xlarge	124
db.r5.xlarge	62
db.r5.large	31
db.r4.16xlarge	960
db.r4.8xlarge	480
db.r4.4xlarge	240
db.r4.2xlarge	120
db.r4.xlarge	60
db.r4.large	30
db.t4g.large	16,5
db.t4g.medium	8,13
db.t3.large	16
db.t3.medium	7,5

> **Note**
>
> Les types d'instances compatibles NVMe peuvent augmenter l'espace temporaire disponible jusqu'à atteindre la taille totale du NVMe. Pour plus d'informations, consultez Amélioration des performances des requêtes pour Aurora PostgreSQL avec Aurora Optimized Reads.

Vous pouvez surveiller le stockage temporaire disponible pour une instance de base de données à l'aide de la métrique CloudWatch `FreeLocalStorage` --> décrite dans CloudWatch Métriques Amazon pour Amazon Aurora. (Cela ne s'applique pas à Aurora Serverless v2).

Pour certaines charges de travail, vous pouvez réduire la quantité de stockage temporaire en allouant plus de mémoire aux processus qui exécutent l'opération. Pour augmenter la mémoire disponible pour une opération, en augmentant les valeurs des paramètres PostgreSQL work_mem ou maintenance_work_mem.

Grandes pages pour Aurora PostgreSQL

Les Huge pages (Grandes pages) sont une fonction de gestion de la mémoire qui réduit la surcharge lorsqu'une instance de base de données fonctionne avec de gros morceaux de mémoire contigus, tels que ceux utilisés par les tampons partagés. Cette fonction PostgreSQL est prise en charge par toutes les versions actuellement disponibles d'Aurora PostgreSQL.

Le paramètre `Huge_pages` est activé par défaut pour toutes les classes d'instance de base de données autres que les classes d'instance t3.medium, db.t3.large, db.t4g.medium, db.t4g.large. Vous ne pouvez pas modifier la valeur du paramètre `huge_pages` ni désactiver cette fonction dans les classes d'instance prises en charge par Aurora PostgreSQL.

Sur les instances de base de données Aurora PostgreSQL qui ne prennent pas en charge la fonctionnalité de mémoire pour les grandes pages, l'utilisation de la mémoire de processus spécifique peut augmenter sans que la charge de travail ne soit modifiée en conséquence.

Le système alloue des segments de mémoire partagée tels que le cache tampon lors du démarrage du serveur. Lorsque la mémoire pour les grandes pages n'est pas disponible, le système ne facture pas ces allocations au processus postmaster. Il inclut plutôt la mémoire dans le processus qui a accédé pour la première fois à chaque page de 4 Ko dans le segment de mémoire partagée.

> **Note**
>
> Les connexions actives partagent la mémoire allouée selon les besoins, quelle que soit la manière dont l'utilisation de la mémoire partagée est suivie d'un processus à l'autre.

Test d'Amazon Aurora PostgreSQL à l'aide des requêtes d'injection d'erreurs

Vous pouvez tester la tolérance aux pannes de votre cluster de bases de données Aurora PostgreSQL à l'aide des requêtes d'injection d'erreurs. Les requêtes d'injection d'erreurs sont émises sous forme de commandes SQL à une instance Amazon Aurora. Les requêtes d'injection de panne vous permettent de créer une panne d'instance afin de tester le basculement et la récupération. Vous pouvez également simuler une panne de réplica Aurora, une panne de disque et une surcharge disque. Les requêtes d'injection de panne sont prises en charge par toutes les versions disponibles d'Aurora PostgreSQL, comme suit.

- Aurora PostgreSQL versions 12, 13, 14, et ultérieures
- Aurora PostgreSQL version 11.7 et ultérieures
- Aurora PostgreSQL version 10.11 et ultérieures

Rubriques

Lorsqu'une requête d'injection d'erreurs spécifie une panne, elle provoque la panne forcée de l'instance de base de données Aurora PostgreSQL. Les autres requêtes d'injection d'erreurs se traduisent par des simulations d'événements d'erreur, mais n'entraînent pas la manifestation de l'événement. Lorsque vous envoyez une requête d'injection d'erreurs, vous pouvez aussi spécifier la durée de la simulation de l'événement d'erreur.

Vous pouvez soumettre une requête d'injection d'erreurs à l'une de vos instances de réplica Aurora en vous connectant au point de terminaison du réplica Aurora. Pour plus d'informations, consultez Connexions de point de terminaison Amazon Aurora.

Test d'un incident d'instance

Vous pouvez forcer l'arrêt d'une instance Aurora PostgreSQL en utilisant la fonction de requête d'injection d'erreurs `aurora_inject_crash()`.

Pour cette requête d'injection d'erreurs, un basculement ne se produit pas. Si vous souhaitez tester un basculement, vous pouvez choisir l'action d'instance Failover (Basculement) pour votre cluster de bases de données dans la console RDS, ou utiliser la commande de l'AWS CLI failover-db-cluster ou l'opération d'API RDS FailoverDBCluster.

Syntaxe

```
SELECT aurora_inject_crash ('instance' | 'dispatcher' | 'node');
```

Options

Cette requête d'injection d'erreurs accepte l'un des types d'incident suivants. Le type d'incident n'est pas sensible à la casse.

'instance'

Simulation d'un incident de la base de données compatible PostgreSQL pour l'instance Amazon Aurora.

'répartiteur '

Simulation d'un incident lié au répartiteur sur l'instance principale pour le cluster de bases de données Aurora. Le répartiteur écrit les mises à jour sur le volume de cluster d'un cluster de bases de données Amazon Aurora.

'node'

Simulation d'un incident de la base de données compatible PostgreSQL et du répartiteur de l'instance Amazon Aurora.

Test d’une défaillance d’un réplica Aurora

Vous pouvez simuler la défaillance d’un réplica Aurora à l’aide de la fonction de requête d’injection d’erreurs `aurora_inject_replica_failure()`.

Une défaillance du réplica Aurora bloque la réplication vers le réplica Aurora ou tous les réplicas Aurora du cluster de bases de données par le pourcentage spécifié pour l’intervalle de temps spécifié. Une fois l’intervalle de temps écoulé, les réplicas Aurora affectés sont automatiquement synchronisés avec l’instance principale.

Syntaxe

```
SELECT aurora_inject_replica_failure(
   percentage_of_failure,
   time_interval,
   'replica_name'
);
```

Options

La requête d’injection d’erreurs accepte les paramètres suivants :

percentage_of_failure

Pourcentage de réplications à bloquer pendant l’événement d’échec. La valeur peut être un nombre double compris entre 0 et 100. Si vous spécifiez 0, aucune réplication n’est bloquée. Si vous spécifiez 100, toute la réplication est bloquée.

time_interval

Durée de simulation de l’échec du réplica Aurora. L’intervalle est exprimé en secondes. Par exemple, si la valeur est de 20, la simulation s’exécute pendant 20 secondes.

> **Note**
>
> Soyez vigilant lorsque vous spécifiez l’intervalle de l’événement d’erreur du réplica Aurora. Si vous spécifiez un intervalle trop long et que votre instance d’enregistreur écrit une importante quantité de données pendant l’événement d’erreur, votre cluster de bases de données Aurora peut considérer que votre réplica Aurora est en panne et le remplacer.

replica_name

Réplica Aurora dans lequel injecter la simulation d'échec. Spécifiez le nom d'un réplica Aurora pour simuler un échec du réplica Aurora unique. Spécifiez une chaîne vide pour simuler des échecs de tous les réplicas Aurora du cluster de bases de données.

Pour identifier les noms de réplica, consultez la colonne `server_id` de la fonction `aurora_replica_status()`. Exemples :

```
postgres=> SELECT server_id FROM aurora_replica_status();
```

Test d'une défaillance disque

Vous pouvez simuler l'échec d'un disque pour un cluster de bases de données Aurora PostgreSQL en utilisant la fonction de requête d'injection d'erreurs `aurora_inject_disk_failure()`.

Pendant la simulation d'un échec du disque, le cluster de bases de données Aurora PostgreSQL marque de façon aléatoire les segments disque comme défectueux. Les demandes adressées à ces segments sont bloquées pendant la durée de la simulation.

Syntaxe

```
SELECT aurora_inject_disk_failure(
   percentage_of_failure,
   index,
   is_disk,
   time_interval
);
```

Options

La requête d'injection d'erreurs accepte les paramètres suivants :

percentage_of_failure

Pourcentage du disque à marquer comme défaillant pendant l'événement d'échec. La valeur peut être un nombre double compris entre 0 et 100. Si vous spécifiez 0, aucune partie du disque n'est marquée comme défaillante. Si vous spécifiez 100, la totalité du disque est marquée comme défaillante.

index

Bloc de données logique spécifique pour lequel simuler l'événement d'erreur. Si vous dépassez la plage de blocs de données logiques ou de données de nœuds de stockage disponibles, vous recevez une erreur vous indiquant la valeur d'index maximale que vous pouvez spécifier. Pour éviter cette erreur, consultez Affichage du statut du volume pour un cluster de bases de données Aurora PostgreSQL.

is_disk

Indique si l'échec de l'injection concerne un bloc logique ou un nœud de stockage. Si vous définissez ce paramètre sur true, cela signifie que les échecs d'injection concernent un bloc logique. Si vous le définissez sur false, cela signifie que les échecs d'injection concernent un nœud de stockage.

time_interval

Durée pendant laquelle l'échec du disque est simulé. L'intervalle est exprimé en secondes. Par exemple, si la valeur est de 20, la simulation s'exécute pendant 20 secondes.

Test d'une surcharge disque

Vous pouvez simuler la congestion d'un disque pour un cluster de bases de données Aurora PostgreSQL en utilisant la fonction de requête d'injection de pannes `aurora_inject_disk_congestion()`.

Pendant la simulation d'une surcharge du disque, le cluster de bases de données Aurora PostgreSQL marque de façon aléatoire les segments disque comme surchargés. Les demandes adressées à ces segments sont retardées entre le délai minimal et le délai maximal spécifiés de la durée de la simulation.

Syntaxe

```
SELECT aurora_inject_disk_congestion(
   percentage_of_failure,
   index,
   is_disk,
   time_interval,
   minimum,
   maximum
);
```

Options

La requête d’injection d’erreurs accepte les paramètres suivants :

percentage_of_failure

Pourcentage du disque à marquer comme surchargé pendant l’événement d’échec. Il s’agit d’une valeur double comprise entre 0 et 100. Si vous spécifiez 0, aucune partie du disque n’est marquée comme surchargée. Si vous spécifiez 100, la totalité du disque est marquée comme surchargée.

index

Bloc logique spécifique de données ou de nœud de stockage à utiliser pour simuler l’événement d’erreur.

Si vous dépassez la plage de blocs de données logiques ou de données de nœuds de stockage disponibles, vous recevez une erreur vous indiquant la valeur d’index maximale que vous pouvez spécifier. Pour éviter cette erreur, consultez Affichage du statut du volume pour un cluster de bases de données Aurora PostgreSQL.

is_disk

Indique si l’échec de l’injection concerne un bloc logique ou un nœud de stockage. Si vous définissez ce paramètre sur true, cela signifie que les échecs d’injection concernent un bloc logique. Si vous le définissez sur false, cela signifie que les échecs d’injection concernent un nœud de stockage.

time_interval

Durée pendant laquelle la congestion du disque est simulée. L’intervalle est exprimé en secondes. Par exemple, si la valeur est de 20, la simulation s’exécute pendant 20 secondes.

minimum, maximum

Durées minimale et maximale du délai de surcharge, en millisecondes. Les valeurs valides vont de 0,0 à 100,0 millisecondes. Les segments de disque marqués comme surchargés sont retardés pendant une durée aléatoire comprise entre la durée minimale et la durée maximale de la simulation. La valeur maximale doit être supérieure à la valeur minimale.

Affichage du statut du volume pour un cluster de bases de données Aurora PostgreSQL

Dans Amazon Aurora, un volume de cluster de base de données se compose d'un ensemble de blocs logiques. Chacun d'eux représente 10 gigaoctets de stockage alloué. Ces blocs sont appelés groupes de protection.

Les données figurant dans chaque groupe de protection sont répliquées sur six périphériques de stockage physiques, appelés nœuds de stockage. Ces nœuds de stockage sont alloués dans trois zones de disponibilité dans la région où se trouve le cluster de bases de données. Chaque nœud de stockage contient à son tour un ou plusieurs blocs de données logiques pour le volume de cluster de base de données. Pour plus d'informations sur les groupes de protection et les nœuds de stockage, consultez Introducing the Aurora Storage Engine sur le blog AWS Database. Pour en savoir plus sur les volumes de cluster Aurora en général, consultez Stockage Amazon Aurora.

Utilisez la fonction `aurora_show_volume_status()` pour renvoyer les variables d'état du serveur suivantes :

- `Disks` — Nombre total de blocs logiques de données pour le volume de cluster de base de données.
- `Nodes` — Nombre total de nœuds de stockage pour le volume de cluster de base de données.

Vous pouvez utiliser la fonction `aurora_show_volume_status()` pour éviter une erreur lors de l'utilisation de la fonction d'injection d'erreurs `aurora_inject_disk_failure()`. La fonction d'injection d'erreurs `aurora_inject_disk_failure()` simule la défaillance de la totalité d'un nœud de stockage ou d'un seul bloc logique de données au sein d'un nœud de stockage. Dans la fonction, vous spécifiez la valeur d'index d'un nœud de stockage ou d'un bloc de données logique spécifique. Toutefois, l'instruction renvoie une erreur si vous spécifiez une valeur d'index supérieure au nombre de nœuds de stockage ou de blocs de données logiques utilisés par le volume de cluster de base de données. Pour en savoir plus sur les requêtes d'injection d'erreurs, consultez Test d'Amazon Aurora PostgreSQL à l'aide des requêtes d'injection d'erreurs.

> **Note**
>
> La fonction `aurora_show_volume_status()` est disponible pour Aurora PostgreSQL version 10.11. Pour de plus amples informations sur les versions d'Aurora PostgreSQL, veuillez consulter Versions Amazon Aurora PostgreSQL et versions du moteur.

Syntaxe

```
SELECT * FROM aurora_show_volume_status();
```

Exemple

```
customer_database=> SELECT * FROM aurora_show_volume_status();
 disks | nodes
-------+-------
    96 |    45
```

Spécifier le disque RAM pour le stats_temp_directory

Vous pouvez utiliser le paramètre Aurora PostgreSQL, `rds.pg_stat_ramdisk_size`, pour spécifier la mémoire système allouée à un disque RAM afin de stocker le code PostgreSQL `stats_temp_directory`. Le paramètre de disque RAM est disponible uniquement dans Aurora PostgreSQL 14 et les versions antérieures.

Sous certaines charges de travail, ce paramètre peut améliorer les performances et réduire les exigences d'I/O. Pour plus d'informations sur `stats_temp_directory`, consultez Run-time Statistics (Statistiques d'exécution) dans la documentation de PostgreSQL. À partir de PostgreSQL version 15, la communauté PostgreSQL est passée à l'utilisation de la mémoire partagée dynamique. Il n'est donc pas nécessaire de définir `stats_temp_directory`.

Pour activer un disque RAM pour votre `stats_temp_directory`, définissez le paramètre `rds.pg_stat_ramdisk_size` à une valeur différente de zéro dans le groupe de paramètres du cluster de base de données utilisé par votre cluster de base de données. Ce paramètre est indiqué en Mo, vous devez donc utiliser une valeur entière. Les expressions, formules et fonctions ne sont pas valides pour le paramètre `rds.pg_stat_ramdisk_size`. Assurez-vous de redémarrer le cluster de bases de données pour que la modification prenne effet. Pour plus d'informations sur la définition des paramètres, consultez Groupes de paramètres pour Amazon Aurora. Pour obtenir plus d'informations sur le redémarrage du cluster de base de données, consultez Redémarrage d'un cluster de bases de données Amazon Aurora ou d'une instance de base de données Amazon Aurora.

Par exemple, la commande AWS CLI suivante définit le paramètre de disque RAM sur 256 Mo.

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name db-cl-pg-ramdisk-testing \
```

```
    --parameters "ParameterName=rds.pg_stat_ramdisk_size, ParameterValue=256,
 ApplyMethod=pending-reboot"
```

Après le redémarrage du cluster de base de données, exécutez la commande suivante pour afficher le statut de `stats_temp_directory` :

```
postgres=> SHOW stats_temp_directory;
```

La commande doit renvoyer les éléments suivants :

```
stats_temp_directory
----------------------------
/rdsdbramdisk/pg_stat_tmp
(1 row)
```

Gestion des fichiers temporaires avec PostgreSQL

Dans PostgreSQL, une requête complexe peut exécuter simultanément plusieurs opérations de tri ou de hachage, chacune utilisant la mémoire de l'instance pour stocker les résultats jusqu'à la valeur spécifiée par le paramètre `work_mem`. Lorsque la mémoire de l'instance n'est pas suffisante, des fichiers temporaires sont créés pour stocker les résultats. Ils sont écrits sur le disque pour terminer l'exécution de la requête. Par la suite, ces fichiers sont automatiquement supprimés une fois la requête terminée. Dans Aurora PostgreSQL, ces fichiers partagent le stockage local avec d'autres fichiers journaux. Vous pouvez surveiller l'espace de stockage local de votre cluster de bases de données Aurora PostgreSQL en observant les métriques Amazon CloudWatch pour `FreeLocalStorage`. Pour plus d'informations, consultez Résoudre les problèmes de stockage local.

Nous recommandons d'utiliser des clusters Lectures optimisées pour Aurora pour les charges de travail impliquant plusieurs requêtes simultanées qui augmentent l'utilisation de fichiers temporaires. Ces clusters utilisent un stockage par bloc local, de type SSD (Solid State Drive), basé sur NVMe (Non-Volatile Memory Express) pour placer les fichiers temporaires. Pour plus d'informations, consultez Amélioration des performances des requêtes pour Aurora PostgreSQL avec Aurora Optimized Reads.

Vous pouvez utiliser les paramètres et fonctions suivants pour gérer les fichiers temporaires dans votre instance.

- **temp_file_limit** : ce paramètre annule toute requête dépassant la taille des fichiers temp_files en Ko. Cette limite empêche toute requête de s'exécuter indéfiniment et de consommer de l'espace disque avec des fichiers temporaires. Vous pouvez estimer la valeur à l'aide des résultats du paramètre `log_temp_files`. Nous vous recommandons d'examiner le comportement de la charge de travail et de définir la limite en fonction de l'estimation. L'exemple suivant présente la manière dont une requête est annulée lorsqu'elle dépasse la limite.

```
postgres=>select * from pgbench_accounts, pg_class, big_table;
```

```
ERROR: temporary file size exceeds temp_file_limit (64kB)
```

- **log_temp_files** : ce paramètre envoie des messages au fichier postgresql.log lorsque les fichiers temporaires d'une session sont supprimés. Ce paramètre produit des journaux lorsqu'une requête est terminée avec succès. Par conséquent, cela peut ne pas aider à résoudre les requêtes actives et de longue durée.

  L'exemple suivant montre que lorsque la requête aboutit, les entrées sont journalisées dans le fichier postgresql.log pendant que les fichiers temporaires sont nettoyés.

```
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.5", size 140353536
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:LOG:
 temporary file: path "base/pgsql_tmp/pgsql_tmp31236.4", size 180428800
2023-02-06 23:48:35 UTC:205.251.233.182(12456):adminuser@postgres:[31236]:STATEMENT:
  select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid limit 10;
```

- **pg_ls_tmpdir** : cette fonction disponible auprès de RDS pour PostgreSQL versions 13 et ultérieures offre une visibilité sur l'utilisation actuelle des fichiers temporaires. La requête terminée n'apparaît pas dans les résultats de la fonction. Dans l'exemple suivant, vous pouvez visualiser les résultats de cette fonction.

```
postgres=>select * from pg_ls_tmpdir();
```

```
      name       |    size    |      modification
-----------------+------------+------------------------
 pgsql_tmp8355.1 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.0 | 1072250880 | 2023-02-06 22:54:43+00
 pgsql_tmp8327.0 | 1072250880 | 2023-02-06 22:54:56+00
 pgsql_tmp8351.1 |  703168512 | 2023-02-06 22:54:56+00
 pgsql_tmp8355.0 | 1072250880 | 2023-02-06 22:54:00+00
 pgsql_tmp8328.1 |  835031040 | 2023-02-06 22:54:56+00
 pgsql_tmp8328.0 | 1072250880 | 2023-02-06 22:54:40+00
(7 rows)
```

```
postgres=>select query from pg_stat_activity where pid = 8355;

query
-------------------------------------------------------------------------------------------
select a.aid from pgbench_accounts a, pgbench_accounts b where a.bid=b.bid order by
 a.bid
(1 row)
```

Le nom du fichier inclut l'ID de traitement (PID) de la session qui a généré le fichier temporaire. Une requête plus avancée, comme dans l'exemple suivant, effectue la somme des fichiers temporaires pour chaque PID.

```
postgres=>select replace(left(name, strpos(name, '.')-1),'pgsql_tmp','') as pid,
 count(*), sum(size) from pg_ls_tmpdir() group by pid;
```

```
 pid  | count |   sum
------+-------+------------------
 8355 |     2 | 2144501760
 8351 |     2 | 2090770432
 8327 |     1 | 1072250880
 8328 |     2 | 2144501760
(4 rows)
```

- **pg_stat_statements** : si vous activez le paramètre pg_stat_statements, vous pouvez consulter l'utilisation moyenne des fichiers temporaires par appel. Vous pouvez identifier le query_id de la requête et l'utiliser pour examiner l'utilisation des fichiers temporaires, comme indiqué dans l'exemple suivant.

```
postgres=>select queryid from pg_stat_statements where query like 'select a.aid from
 pgbench%';
```

```
       queryid
----------------------
 -7170349228837045701
(1 row)
```

```
postgres=>select queryid, substr(query,1,25), calls, temp_blks_read/calls
 temp_blks_read_per_call, temp_blks_written/calls temp_blks_written_per_call from
 pg_stat_statements where queryid = -7170349228837045701;
```

```
       queryid        |          substr           | calls | temp_blks_read_per_call |
 temp_blks_written_per_call
----------------------+---------------------------+-------+-------------------------
+---------------------------
 -7170349228837045701 | select a.aid from pgbench |    50 |                  239226 |
                   388678
(1 row)
```

- **Performance Insights** : dans le tableau de bord Performance Insights, vous pouvez consulter l'utilisation des fichiers temporaires en activant les métriques temp_bytes et temp_files. Vous pouvez ensuite voir la moyenne de ces deux métriques et voir comment elles correspondent à la charge de travail des requêtes. La vue de Performance Insights n'affiche pas spécifiquement les requêtes qui génèrent les fichiers temporaires. Toutefois, lorsque vous associez Performance Insights à la requête indiquée pour `pg_ls_tmpdir`, vous pouvez dépanner, analyser et déterminer les modifications apportées à la charge de travail de vos requêtes.

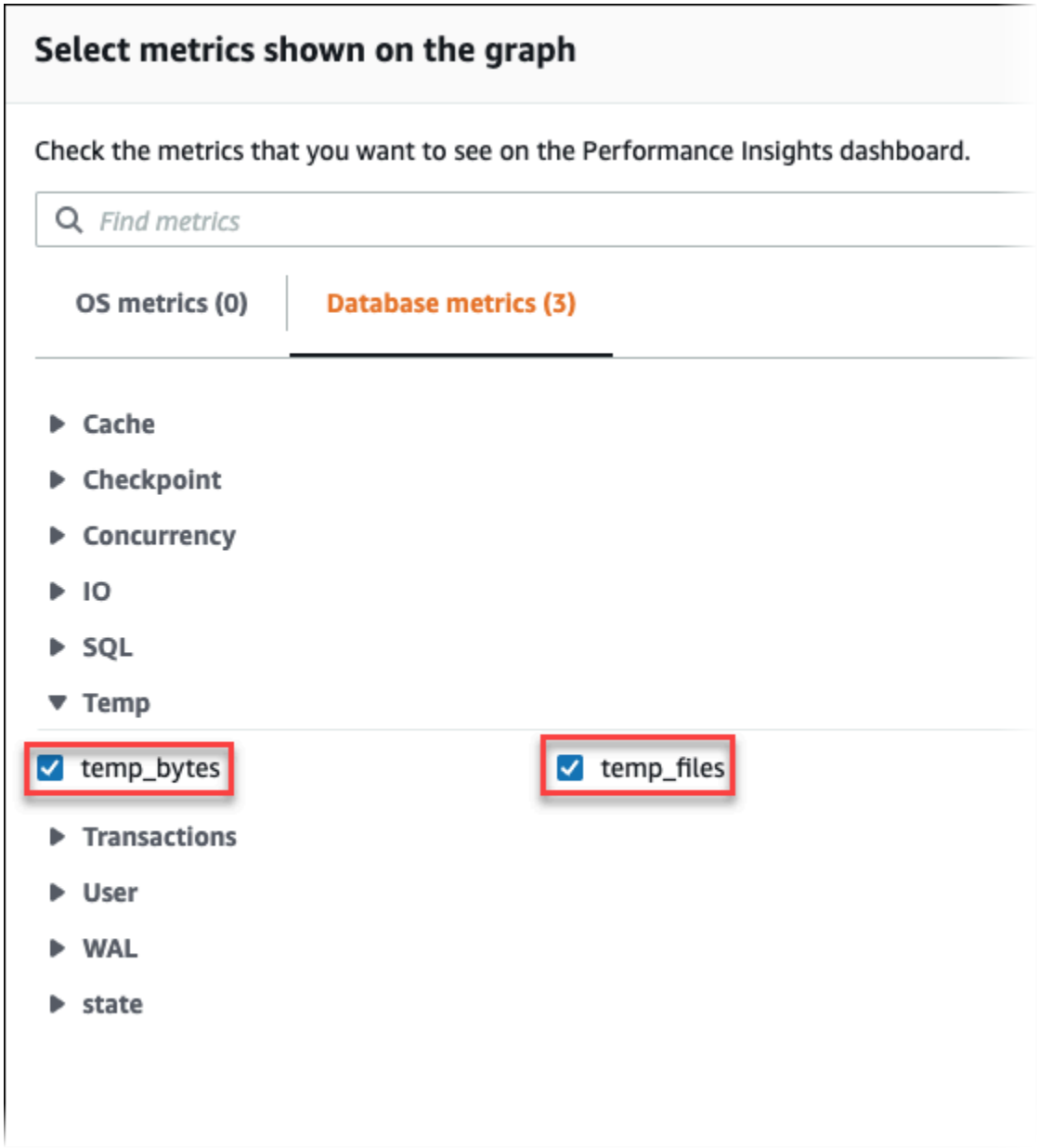
  Pour plus d'informations sur l'analyse des métriques et des requêtes à l'aide de Performance Insights, consultez Analyse des métriques à l'aide du tableau de bord de Performance Insights.

Pour consulter un exemple d’utilisation des fichiers temporaires avec Performance Insights, consultez Affichage de l’utilisation des fichiers temporaires avec Performance Insights

Affichage de l’utilisation des fichiers temporaires avec Performance Insights

Vous pouvez consulter utiliser Performance Insights pour afficher l’utilisation des fichiers temporaires en activant les métriques temp_bytes et temp_files. La vue indiquée dans Performance Insights ne montre pas les requêtes spécifiques qui génèrent des fichiers temporaires. Toutefois, lorsque vous associez Performance Insights à la requête affichée pour `pg_ls_tmpdir`, vous pouvez dépanner, analyser et déterminer les modifications apportées à la charge de travail de vos requêtes.

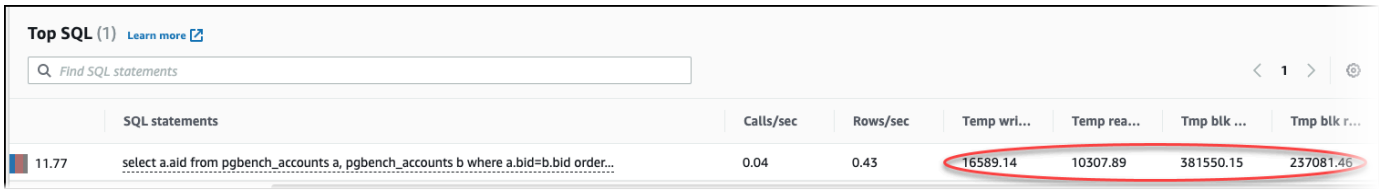
1. Dans le tableau de bord de Performance Insights, choisissez Gérer les métriques.
2. Choisissez Métriques de base de données et sélectionnez les métriques temp_bytes et temp_files comme indiqué dans l’image suivante.



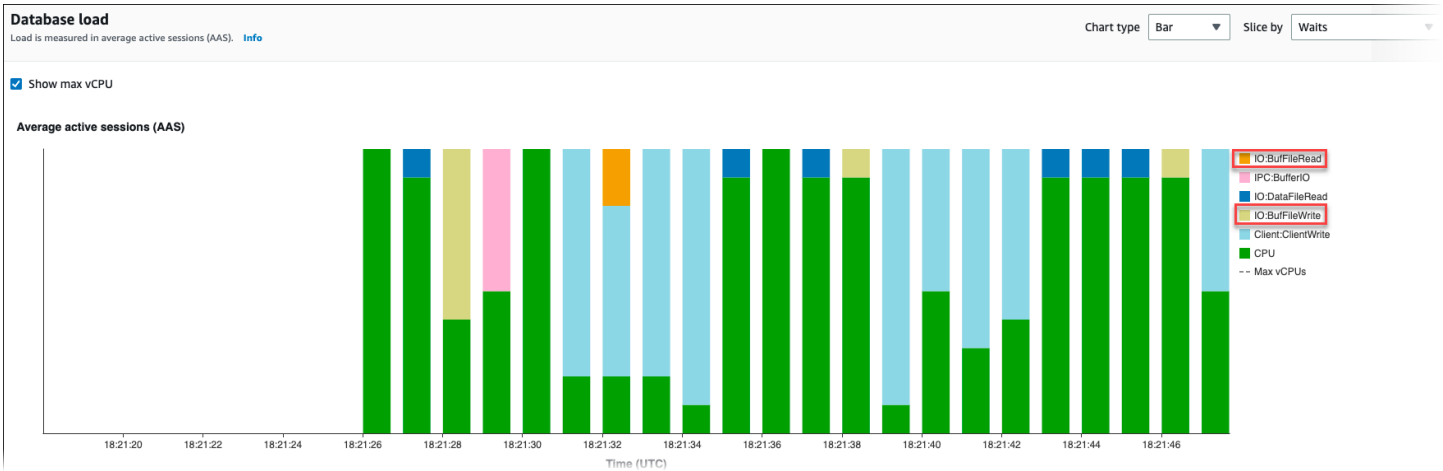
3. Dans l'onglet SQL maximum, cliquez sur l'icône Préférences.
4. Dans la fenêtre Préférences, activez les statistiques suivantes pour qu'elles apparaissent dans l'onglet SQL maximum et choisissez Continuer.

   - Nombre d'écritures temporaires/seconde
   - Nombre de lectures temporaires/seconde
   - Écritures/appels en bloc temporaires
   - Lectures/appels en bloc temporaires

5. Le fichier temporaire est décomposé lorsqu'il est associé à la requête affichée pour `pg_ls_tmpdir`, comme le montre l'exemple suivant.



Les événements `IO:BufFileRead` et `IO:BufFileWrite` se produisent lorsque les requêtes les plus importantes de votre charge de travail créent souvent des fichiers temporaires. Vous pouvez utiliser l'analyse des performances pour identifier les requêtes les plus importantes en attente sur `IO:BufFileRead` et `IO:BufFileWrite` en passant en revue Sessions actives en moyenne (AAS) dans les sections Charge de base de données et Principaux éléments SQL.



Pour plus d'informations sur la façon d'analyser les requêtes les plus importantes et la charge par événement d'attente à l'aide de l'analyse des performances, consultez Présentation de l'onglet Top SQL (Principaux éléments SQL). Vous devez identifier et ajuster les requêtes qui entraînent une augmentation de l'utilisation des fichiers temporaires et des événements d'attente associés. Pour plus d'informations sur ces événements d'attente et les mesures correctives, consultez IO:BufFileRead et IO:BufFileWrite.

> **Note**
>
> Le paramètre `work_mem` contrôle le moment où la mémoire de l'opération de tri est insuffisante et les résultats sont écrits dans des fichiers temporaires. Nous vous recommandons de ne pas modifier la valeur de ce paramètre au-delà de la valeur par défaut, car cela permettrait à chaque session de base de données de consommer davantage de mémoire. En outre, une session unique qui effectue des jointures et des tris complexes peut

effectuer des opérations parallèles au cours desquelles chaque opération consomme de la mémoire.
Il est recommandé de définir ce paramètre au niveau de la session à l'aide de la commande `SET work_mem` lorsque vous disposez d'un rapport volumineux comportant plusieurs jointures et tris. La modification n'est alors appliquée qu'à la session en cours et ne modifie pas la valeur de manière globale.

Réglage des événements d'attente pour Aurora PostgreSQL

Les événements d'attente constituent un outil de réglage important pour Aurora PostgreSQL. Lorsque vous parvenez à déterminer pourquoi les sessions sont en attente de ressources et ce qu'elles font, vous êtes mieux à même de réduire les goulets d'étranglement. Vous pouvez utiliser les informations de cette section pour déterminer les causes possibles et les actions correctives à mettre en œuvre. Avant de plonger dans cette section, nous vous recommandons vivement de comprendre les concepts de base d'Aurora, en particulier les sujets suivants :

- Stockage Amazon Aurora
- Gestion des performances et dimensionnement des clusters de bases de données Aurora

Important

Les événements d'attente présentés dans cette section sont spécifiques à Aurora PostgreSQL. Les informations de réglage fournies dans cette section s'appliquent uniquement à Amazon Aurora, et non à RDS pour PostgreSQL.
Certains événements d'attente mentionnés dans cette section n'ont pas leur équivalent dans les versions open source de ces moteurs de base de données. D'autres événements d'attente portent le même nom que des événements des moteurs open source, mais se comportent différemment. Par exemple, le stockage Amazon Aurora fonctionne différemment du stockage open source, par conséquent les événements d'attente liés au stockage indiquent des conditions de ressources différentes.

Rubriques

- Concepts essentiels à connaître pour le réglage d'Aurora PostgreSQL
- Événements d'attente Aurora PostgreSQL

- Client:ClientRead
- Client:ClientWrite
- CPU
- IO : BufFileRead et IO : BufFileWrite
- IO:DataFileRead
- IO:XactSync
- IPC:DamRecordTxAck
- IPC:parallel wait events
- IPC : ProcArrayGroupUpdate
- Lock:advisory
- Lock:extend
- Lock:Relation
- Lock:transactionid
- Lock:tuple
- LWLock:buffer_content (BufferContent)
- LWLock:buffer_mapping
- LWLock:BufferIO (IPC:BufferIO)
- LWLock:lock_manager
- LWLock:MultiXact
- LWLock:pg_stat_statements
- Timeout:PgSleep

Concepts essentiels à connaître pour le réglage d'Aurora PostgreSQL

Avant de procéder au réglage de votre base de données Aurora PostgreSQL, vous devez savoir ce que sont les événements d'attente et pourquoi ils se produisent. Examinez également l'architecture de base d'Aurora PostgreSQL en termes de mémoire et de disque. Un diagramme d'architecture très utile est disponible dans le wikibook PostgreSQL.

Rubriques

- Événements d'attente Aurora PostgreSQL

- Mémoire d'Aurora PostgreSQL
- Processus d'Aurora PostgreSQL

Événements d'attente Aurora PostgreSQL

Un événement d'attente désigne une ressource pour laquelle une session est en attente. Par exemple, l'événement d'attente `Client:ClientRead` se produit lorsqu'Aurora PostgreSQL attend de recevoir des données du client. Les ressources généralement attendues par une session sont les suivantes :

- Accès monothread à une mémoire tampon, par exemple lorsqu'une session tente de modifier une mémoire tampon
- Ligne verrouillée par une autre session
- Lecture d'un fichier de données
- Écriture de fichier journal

Par exemple, pour répondre à une requête, la session peut effectuer une analyse complète de la table. Si les données ne sont pas déjà en mémoire, la session attend la fin des opérations d'I/O disque. Lorsque les mémoires tampons sont lues en mémoire, la session peut être contrainte d'attendre parce que d'autres sessions accèdent aux mêmes mémoires tampons. La base de données enregistre les attentes à l'aide d'un événement d'attente prédéfini. Ces événements sont regroupés en catégories.

En soi, un événement d'attente n'indique pas un problème de performances. Par exemple, si les données demandées ne sont pas en mémoire, il est nécessaire de les lire sur le disque. Si une session verrouille une ligne pour une mise à jour, une autre session attend que la ligne soit déverrouillée pour pouvoir la mettre à jour. Une validation nécessite d'attendre la fin de l'écriture dans un fichier journal. Les attentes font partie intégrante du fonctionnement normal d'une base de données.

Un grand nombre d'événements d'attente indique généralement un problème de performances. Dans ce cas, vous pouvez utiliser les données des événements d'attente pour déterminer où les sessions passent du temps. Par exemple, si plusieurs heures sont désormais nécessaires à l'exécution d'un rapport qui ne prend habituellement que quelques minutes, vous pouvez identifier les événements d'attente qui contribuent le plus au temps d'attente total. La détermination des causes des principaux événements d'attente peut vous permettre d'apporter des modifications qui auront

pour effet d'améliorer les performances. Par exemple, si votre session est en attente sur une ligne qui a été verrouillée par une autre session, vous pouvez mettre fin à la session à l'origine du verrouillage.

Mémoire d'Aurora PostgreSQL

La mémoire d'Aurora PostgreSQL se décompose en deux parties : la mémoire partagée et la mémoire locale.

Rubriques

Mémoire partagée d'Aurora PostgreSQL

Aurora PostgreSQL alloue de la mémoire partagée au démarrage de l'instance. La mémoire partagée se décompose en sous-zones. Vous trouverez ci-dessous une description des principales sous-zones.

Rubriques

Mémoires tampons partagées

Le groupe de mémoires tampons partagées est une zone de mémoire d'Aurora PostgreSQL qui contient toutes les pages actuellement ou précédemment utilisées par les connexions d'applications. Une page correspond à la version mémoire d'un bloc de disque. Le groupe de mémoires tampons partagées met en cache les blocs de données lus sur le disque. Le groupe réduit la nécessité de relire les données à partir du disque, ce qui améliore l'efficacité de la base de données.

Chaque table et chaque index est stocké sous la forme d'un tableau de pages de taille fixe. Chaque bloc contient plusieurs tuples, qui correspondent à des lignes. Un tuple peut être stocké sur n'importe quelle page.

Le groupe de mémoires tampons partagées dispose d'une mémoire limitée. Si une nouvelle requête requiert une page qui n'est pas en mémoire, et qu'il n'y a plus de mémoire disponible, Aurora PostgreSQL expulse une page moins fréquemment utilisée pour répondre à la requête. La politique d'expulsion est implémentée par un algorithme de balayage horaire.

Le paramètre `shared_buffers` détermine la quantité de mémoire que le serveur consacre à la mise en cache des données.

Mémoires tampons WAL (Write-Ahead Log)

Une mémoire tampon WAL (Write-Ahead Log) contient des données de transaction qu'Aurora PostgreSQL écrit ultérieurement sur un stockage permanent. Le mécanisme WAL permet à Aurora PostgreSQL d'effectuer les opérations suivantes :

- Récupérer des données après une défaillance
- Réduire les I/O disque en évitant les écritures fréquentes sur disque

Lorsqu'un client modifie des données, Aurora PostgreSQL écrit les modifications dans la mémoire tampon WAL. Lorsque le client émet une commande `COMMIT`, le processus d'écriture WAL écrit les données de transaction dans le fichier WAL.

Le paramètre `wal_level` détermine la quantité d'informations écrites dans la mémoire tampon WAL.

Mémoire locale d'Aurora PostgreSQL

Chaque processus backend alloue de la mémoire locale pour le traitement des requêtes.

Rubriques

- Zone de mémoire de travail
- Zone de mémoire des travaux de maintenance
- Zone de mémoire tampon temporaire

Zone de mémoire de travail

La zone de mémoire de travail contient des données temporaires pour les requêtes qui effectuent des opérations de tri et de hachage. Par exemple, une requête contenant une clause `ORDER BY` effectue un tri. Les requêtes utilisent des tables de hachage dans les jointures de hachage et les agrégations.

Le paramètre `work_mem` indique la quantité de mémoire à utiliser par les tables de hachage et les opérations de tri internes avant d'écrire dans des fichiers disque temporaires. La valeur par défaut est de 4 Mo. Plusieurs sessions peuvent s'exécuter simultanément et chacune peut exécuter des opérations de maintenance en parallèle. La mémoire de travail totale utilisée peut donc être un multiple du paramètre `work_mem`.

Zone de mémoire des travaux de maintenance

La zone de mémoire des travaux de maintenance met les données en cache pour les opérations de maintenance. Ces opérations incluent l'opération VACUUM, la création d'un index et l'ajout de clés étrangères.

Le paramètre `maintenance_work_mem` spécifie la quantité maximale de mémoire à utiliser par les opérations de maintenance. La valeur par défaut est de 64 Mo. Une session de base de données ne peut exécuter qu'une seule opération de maintenance à la fois.

Zone de mémoire tampon temporaire

La zone de mémoire tampon temporaire met en cache les tables temporaires pour chaque session de base de données.

Chaque session alloue des mémoires tampons temporaires en fonction des besoins jusqu'à la limite que vous spécifiez. Lorsque la session se termine, le serveur efface le contenu des mémoires tampons.

Le paramètre `temp_buffers` définit le nombre maximal de mémoires tampons temporaires utilisées par chaque session. Avant la première utilisation de tables temporaires au sein d'une session, vous pouvez modifier la valeur `temp_buffers`.

Processus d'Aurora PostgreSQL

Aurora PostgreSQL utilise différents processus.

Rubriques

- Processus postmaster
- Processus backend
- Processus d'arrière-plan

Processus postmaster

Le processus postmaster est le premier qui est lancé lorsque vous démarrez Aurora PostgreSQL. Les principales responsabilités du processus postmaster sont les suivantes :

- Créer et surveiller les processus d'arrière-plan

- Recevoir les requêtes d'authentification des processus clients, et les authentifier avant d'autoriser la base de données à traiter les requêtes

Processus backend

Si le processus postmaster authentifie une requête client, il crée un nouveau processus backend, également appelé processus postgres. Un processus client se connecte à un seul processus backend. Le processus client et le processus backend communiquent directement sans intervention du processus postmaster.

Processus d'arrière-plan

Le processus postmaster crée plusieurs processus qui effectuent différentes tâches backend. Les plus importants sont les suivants :

- Dispositif d'écriture WAL

  Aurora PostgreSQL écrit les données contenues dans la mémoire tampon WAL (Write Ahead Logging) dans les fichiers journaux. Le principe de l'approche WAL est que la base de données ne peut pas écrire les modifications dans les fichiers de données tant que la base de données n'a pas écrit les enregistrements de journal décrivant ces modifications sur le disque. Le mécanisme WAL réduit les I/O disque et permet à Aurora PostgreSQL d'utiliser les journaux pour restaurer la base de données après une défaillance.
- Dispositif d'écriture d'arrière-plan

  Ce processus écrit périodiquement les pages modifiées des mémoires tampons vers les fichiers de données. Une page est considérée comme modifiée lorsqu'un processus backend la modifie en mémoire.
- Démon autovacuum

  Le démon est composé des éléments suivants :

  - Le lanceur autovacuum
  - Les processus employés autovacuum

  Lorsque la fonction autovacuum est activée, elle recherche les tables dans lesquelles un grand nombre de tuples ont été insérés, mis à jour ou supprimés. Les responsabilités du démon sont les suivantes :

  - Récupérer ou réutiliser l'espace disque occupé par les lignes mises à jour ou supprimées

- Mettre à jour les statistiques utilisées par le planificateur
- Protéger contre la perte d'anciennes données en raison du renvoi à la ligne de l'ID de transaction

La fonction autovacuum automatise l'exécution des commandes `VACUUM` et `ANALYZE`. `VACUUM`présente les variantes suivantes : standard et complet. La variante standard s'exécute parallèlement à d'autres opérations de base de données. `VACUUM FULL` requiert un verrou exclusif sur la table sur laquelle il travaille. Ainsi, il ne peut pas fonctionner parallèlement à des opérations qui accèdent à la même table. `VACUUM` génère beaucoup de trafic I/O, ce qui peut nuire aux performances des autres sessions actives.

Événements d'attente Aurora PostgreSQL

Le tableau suivant répertorie les événements d'attente liés à Aurora PostgreSQL, qui révèlent souvent des problèmes de performances, et résume leurs causes et les actions correctives les plus courantes. Les événements d'attente suivants sont un sous-ensemble de la liste disponible dans Événements d’attente Amazon Aurora PostgreSQL.

Événement d'attente	Définition
Client:ClientRead	Cet événement se produit lorsqu'Aurora PostgreSQL attend de recevoir des données du client.
Client:ClientWrite	Cet événement se produit lorsqu'Aurora PostgreSQL attend d'écrire des données sur le client.
CPU	Cet événement se produit lorsqu'un thread est actif dans l'UC ou qu'il est en attente d'UC.
IO : BufFileRead et IO : BufFileWrite	Ces événements se produisent lorsqu'Aurora PostgreSQL crée des fichiers temporaires.
IO:DataFileRead	Cet événement se produit lorsqu'une connexion attend qu'un processus backend lise une page requise à partir du stockage parce que la page n'est pas disponible dans la mémoire partagée.
IO:XactSync	Cet événement se produit lorsque la base de données attend que le sous-système de stockage

Événement d'attente	Définition
	Aurora confirme la validation d'une transaction standard, ou la validation ou restauration d'une transaction préparée.
IPC:DamRecordTxAck	Cet événement se produit lorsqu'Aurora PostgreSQL, dans une session utilisant des flux d'activité de base de données, génère un événement de flux d'activité, puis attend que cet événement devienne durable.
Lock:advisory	Cet événement se produit lorsqu'une application PostgreSQL utilise un verrou pour coordonner l'activité sur plusieurs sessions.
Lock:extend	Cet événement se produit lorsqu'un processus backend attend de verrouiller une relation pour l'étendre alors qu'un autre processus présente un verrou sur cette relation dans le même but.
Lock:Relation	Cet événement se produit lorsqu'une requête attend d'acquérir un verrou sur une table ou une vue actuellement verrouillée par une autre transaction.
Lock:transactionid	Cet événement se produit lorsqu'une transaction attend un verrou de niveau ligne.
Lock:tuple	Cet événement se produit lorsqu'un processus backend attend d'acquérir un verrou sur un tuple.
LWLock:buffer_content (BufferContent)	Cet événement se produit lorsqu'une session attend de lire ou d'écrire une page de données en mémoire alors que celle-ci est verrouillée en écriture dans une autre session.

Événement d'attente	Définition
LWLock:buffer_mapping	Cet événement se produit lorsqu'une session attend d'associer un bloc de données à une mémoire tampon dans le groupe de mémoires tampons partagées.
LWLock:BufferIO (IPC:BufferIO)	Cet événement se produit lorsqu'Aurora PostgreSQL ou RDS for PostgreSQL attend que d'autres processus terminent leurs opérations d'entrée/sortie (I/O) en cas de tentative simultanée d'accès à une page.
LWLock:lock_manager	Cet événement se produit lorsque le moteur Aurora PostgreSQL conserve la zone de mémoire du verrou partagé pour allouer, vérifier et annuler l'allocation d'un verrou parce qu'il est impossible d'utiliser un verrou à chemin d'accès rapide.
LWLock:MultiXact	Ce type d'événement survient lorsqu'Aurora PostgreSQL garde une session ouverte pour effectuer plusieurs transactions qui impliquent la même ligne dans une table. L'événement d'attente indique quel aspect du traitement des transactions multiples génère l'événement d'attente, c'est-à-dire LWLock:MultiXactOffsetSLRU, LWLock:MultiXactOffsetBuffer, LWLock:MultiXactMemberSLRU ou LWLock:MultiXactMemberBuffer.
Timeout:PgSleep	Cet événement se produit lorsqu'un processus serveur a appelé la fonction `pg_sleep` et attend l'expiration du délai de mise en veille.

Client:ClientRead

L'événement `Client:ClientRead` se produit lorsqu'Aurora PostgreSQL attend de recevoir des données du client.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d’attente s’appliquent à Aurora PostgreSQL 10 et versions ultérieures.

Contexte

Un cluster de bases de données Aurora PostgreSQL attend de recevoir des données du client. Le cluster de bases de données Aurora PostgreSQL doit recevoir les données du client avant de pouvoir envoyer plus de données au client. La période pendant laquelle le cluster attend avant de recevoir les données du client est un événement `Client:ClientRead`.

Causes probables de l’augmentation du nombre d’événements d’attente

Les principales causes de l’événement d’attente `Client:ClientRead` sont les suivantes :

Latence réseau accrue

La latence réseau peut être accrue entre le cluster de bases de données Aurora PostgreSQL et le client. Une latence réseau plus élevée augmente le temps de réception des données du client par le cluster de bases de données.

Charge accrue sur le client

Le client peut être soumis à une pression exercée sur l’UC ou à une saturation du réseau. Une augmentation de la charge exercée sur le client peut retarder la transmission des données du client vers le cluster de bases de données Aurora PostgreSQL.

Nombre excessif d’allers-retours réseau

Un grand nombre d’allers-retours réseau entre le cluster de bases de données Aurora PostgreSQL et le client peut retarder la transmission des données du client vers le cluster de bases de données Aurora PostgreSQL.

Opération de copie importante

Lors d'une opération de copie, les données sont transférées du système de fichiers du client vers le cluster de bases de données Aurora PostgreSQL. L'envoi d'une grande quantité de données au cluster de bases de données peut retarder la transmission des données du client vers le cluster de bases de données.

Connexion client inactive

Une connexion à une instance de base de données Aurora PostgreSQL est inactive dans l'état de transaction et attend qu'un client envoie des données supplémentaires ou émette une commande. Cet état peut entraîner une augmentation des événements `Client:ClientRead`.

PgBouncer utilisé pour le regroupement des connexions

PgBouncer possède un paramètre de configuration réseau de bas niveau appelé `pkt_buf`, qui par défaut est défini sur 4 096. Si la charge de travail envoie des paquets de requêtes de plus de 4 096 octets via PgBouncer, nous vous recommandons de remplacer la valeur du paramètre `pkt_buf` par 8 192. Si le nouveau paramètre ne permet pas de réduire le nombre d'événements `Client:ClientRead`, nous vous recommandons d'augmenter le paramètre `pkt_buf` en le définissant sur des valeurs plus élevées, telles que 16 384 ou 32 768. Si le texte de la requête est volumineux, un paramètre plus élevé peut être particulièrement utile.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Placez les clients dans la même zone de disponibilité et le même sous-réseau VPC que le cluster
- Procédez à la mise à l'échelle du client
- Utilisez des instances de la génération actuelle
- Augmentez la bande passante réseau
- Surveillez les valeurs maximales des métriques de performances réseau
- Surveillez les transactions dont l'état est « idle in transaction » (transaction inactive)

Placez les clients dans la même zone de disponibilité et le même sous-réseau VPC que le cluster

Pour réduire la latence réseau et augmenter son débit, placez les clients dans la même zone de disponibilité et le même sous-réseau de cloud privé virtuel (VPC) que le cluster de bases de données

Aurora PostgreSQL. Assurez-vous que les clients sont aussi proches que possible du cluster de bases de données géographiquement parlant.

Procédez à la mise à l'échelle du client

À l'aide d'Amazon CloudWatch ou d'autres métriques d'hôte, déterminez si votre client est actuellement soumis à des contraintes liées à l'UC ou à la bande passante réseau, ou aux deux. Si le client est soumis à des contraintes, mettez-le à l'échelle en conséquence.

Utilisez des instances de la génération actuelle

La classe d'instance de base de données que vous utilisez ne prend peut-être pas en charge les trames jumbo. Si vous exécutez votre application sur Amazon EC2, pensez à utiliser une instance de génération actuelle pour le client. Configurez également l'unité de transmission maximale (MTU) sur le système d'exploitation client. Cette technique peut réduire le nombre d'allers-retours réseau et augmenter le débit du réseau. Pour plus d'informations, consultez Trames jumbo (MTU de 9001) dans le Guide de l'utilisateur Amazon EC2.

Pour plus d'informations sur les classes d'instance de base de données, consultez Classes d'instance de base de données Amazon Aurora. Pour déterminer quelle classe d'instance de base de données est l'équivalent d'un type d'instance Amazon EC2, placez `db.` avant le nom du type d'instance Amazon EC2. Par exemple, l'instance Amazon EC2 `r5.8xlarge` est l'équivalent de la classe d'instance de base de données `db.r5.8xlarge`.

Augmentez la bande passante réseau

Utilisez les métriques Amazon CloudWatch `NetworkReceiveThroughput` et `NetworkTransmitThroughput` pour surveiller le trafic réseau entrant et sortant sur le cluster de bases de données. Ces métriques peuvent vous aider à déterminer si la bande passante réseau est suffisante pour votre charge de travail.

Si la bande passante réseau est insuffisante, augmentez-la. Si le client AWS ou votre instance de base de données atteint les limites de sa bande passante réseau, le seul moyen d'augmenter la bande passante est d'augmenter la taille de l'instance de base de données.

Pour plus d'informations sur les métriques CloudWatch, consultez CloudWatch Métriques Amazon pour Amazon Aurora.

Surveillez les valeurs maximales des métriques de performances réseau

Si vous utilisez des clients Amazon EC2, Amazon EC2 fournit des valeurs maximales pour les métriques de performances réseau, y compris pour la bande passante réseau entrante et sortante

agrégée. Il assure également le suivi des connexions pour garantir un retour optimal des paquets et un accès aux services locaux de liaison pour des services tels que le système de noms de domaine (DNS). Pour surveiller ces valeurs maximales, utilisez un pilote réseau amélioré à jour et surveillez les performances réseau de votre client.

Pour plus d'informations, consultez Surveillance des performances réseau de votre instance Amazon EC2 dans le Guide de l'utilisateur Amazon EC2 et Surveillance des performances réseau de votre instance Amazon EC2 dans le Guide de l'utilisateur Amazon EC2.

Surveillez les transactions dont l'état est « idle in transaction » (transaction inactive)

Déterminez si le nombre de connexions `idle in transaction` est croissant. Pour ce faire, surveillez la colonne `state` de la table `pg_stat_activity`. Vous pouvez identifier la source des connexions en exécutant une requête semblable à la suivante.

```
select client_addr, state, count(1) from pg_stat_activity
where state like 'idle in transaction%'
group by 1,2
order by 3 desc
```

Client:ClientWrite

L'événement `Client:ClientWrite` se produit lorsqu'Aurora PostgreSQL attend d'écrire des données sur le client.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à Aurora PostgreSQL 10 et versions ultérieures.

Contexte

Un processus client doit lire toutes les données reçues d'un cluster de bases de données Aurora PostgreSQL avant que le cluster puisse envoyer plus de données. La période pendant laquelle le cluster attend avant d'envoyer plus de données au client est un événement `Client:ClientWrite`.

Un débit réseau réduit entre le cluster de bases de données Aurora PostgreSQL et le client peut provoquer cet événement. Cet événement peut également se produire lorsque le client est soumis à une pression exercée sur l'UC ou à une saturation du réseau.. On parle de pression exercée sur l'UC lorsque l'UC est entièrement utilisée et que des tâches attendent du temps UC. On parle de saturation du réseau lorsque le réseau situé entre la base de données et le client transporte plus de données qu'il ne peut en gérer.

Causes probables de l'augmentation du nombre d'événements d'attente

Les principales causes de l'événement d'attente `Client:ClientWrite` sont les suivantes :

Latence réseau accrue

La latence réseau peut être accrue entre le cluster de bases de données Aurora PostgreSQL et le client. Une latence réseau plus élevée augmente le temps nécessaire au client pour recevoir les données.

Charge accrue sur le client

Le client peut être soumis à une pression exercée sur l'UC ou à une saturation du réseau. Une augmentation de la charge exercée sur le client retarde la réception des données du cluster de bases de données Aurora PostgreSQL.

Gros volume de données envoyé au client

Le cluster de bases de données Aurora PostgreSQL peut envoyer une grande quantité de données au client. Un client peut ne pas être en mesure de recevoir les données aussi rapidement que le cluster les envoie. Certaines activités comme la copie d'une table volumineuse peuvent entraîner une augmentation des événements `Client:ClientWrite`.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Placez les clients dans la même zone de disponibilité et le même sous-réseau VPC que le cluster
- Utilisez des instances de la génération actuelle
- Réduisez la quantité de données envoyée au client
- Procédez à la mise à l'échelle du client

Placez les clients dans la même zone de disponibilité et le même sous-réseau VPC que le cluster

Pour réduire la latence réseau et augmenter son débit, placez les clients dans la même zone de disponibilité et le même sous-réseau de cloud privé virtuel (VPC) que le cluster de bases de données Aurora PostgreSQL.

Utilisez des instances de la génération actuelle

La classe d'instance de base de données que vous utilisez ne prend peut-être pas en charge les trames jumbo. Si vous exécutez votre application sur Amazon EC2, pensez à utiliser une instance de génération actuelle pour le client. Configurez également l'unité de transmission maximale (MTU) sur le système d'exploitation client. Cette technique peut réduire le nombre d'allers-retours réseau et augmenter le débit du réseau. Pour plus d'informations, consultez Trames jumbo (MTU de 9001) dans le Guide de l'utilisateur Amazon EC2.

Pour plus d'informations sur les classes d'instance de base de données, consultez Classes d'instance de base de données Amazon Aurora. Pour déterminer quelle classe d'instance de base de données est l'équivalent d'un type d'instance Amazon EC2, placez `db.` avant le nom du type d'instance Amazon EC2. Par exemple, l'instance Amazon EC2 `r5.8xlarge` est l'équivalent de la classe d'instance de base de données `db.r5.8xlarge`.

Réduisez la quantité de données envoyée au client

Lorsque cela est possible, ajustez votre application pour réduire la quantité de données que le cluster de bases de données Aurora PostgreSQL envoie au client. Ces ajustements permettent d'éviter les conflits liés à l'UC et au réseau sur le client.

Procédez à la mise à l'échelle du client

À l'aide d'Amazon CloudWatch ou d'autres métriques d'hôte, déterminez si votre client est actuellement soumis à des contraintes liées à l'UC ou à la bande passante réseau, ou aux deux. Si le client est soumis à des contraintes, mettez-le à l'échelle en conséquence.

CPU

Cet événement se produit lorsqu'un thread est actif dans l'UC ou qu'il est en attente d'UC.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à Aurora PostgreSQL 9.6 et versions ultérieures.

Contexte

L'unité centrale (UC) est le composant d'un ordinateur qui exécute les instructions. Par exemple, les instructions de l'UC effectuent des opérations arithmétiques et échangent des données en mémoire. Si une requête augmente le nombre d'instructions qu'elle exécute par le biais du moteur de base de données, le temps passé à exécuter la requête augmente. La planification du temps UC consiste à accorder du temps UC à un processus. La planification est orchestrée par le noyau du système d'exploitation.

Rubriques

Comment savoir quand cette attente se produit

Cet événement d'attente `CPU` indique qu'un processus backend est actif dans l'UC ou qu'il est en attente d'UC. Cela se produit lorsqu'une requête contient les informations suivantes :

- La colonne `pg_stat_activity.state` indique la valeur `active`.

- Les colonnes `wait_event_type` et `wait_event` de `pg_stat_activity` indiquent toutes les deux `null`.

Pour voir les processus backend qui utilisent l'UC ou sont en attente de celle-ci, exécutez la requête suivante.

```
SELECT *
FROM   pg_stat_activity
WHERE  state = 'active'
AND    wait_event_type IS NULL
AND    wait_event IS NULL;
```

Métrique DBLoadCPU

La métrique Performance Insights de l'UC est `DBLoadCPU`. La valeur de `DBLoadCPU` peut être différente de la valeur de la métrique Amazon CloudWatch `CPUUtilization`. Cette dernière est collectée à partir de l'hyperviseur pour une instance de base de données.

Métriques os.cpuUtilization

Les métriques du système d'exploitation Performance Insights fournissent des informations détaillées sur l'utilisation de l'UC. Par exemple, vous pouvez afficher les métriques suivantes :

- `os.cpuUtilization.nice.avg`
- `os.cpuUtilization.total.avg`
- `os.cpuUtilization.wait.avg`
- `os.cpuUtilization.idle.avg`

Performance Insights présente l'utilisation du processeur par le moteur de base de données sous la forme `os.cpuUtilization.nice.avg`.

Cause probable de la planification du temps UC

Du point de vue du système d'exploitation, l'UC est active lorsqu'elle n'exécute pas de thread inactif. L'UC est active lorsqu'elle effectue un calcul, mais elle est également active lorsqu'elle attend des I/O mémoire. Ce type d'I/O domine la charge de travail standard d'une base de données.

Les processus sont susceptibles d'attendre d'être planifiés sur une UC lorsque les conditions suivantes sont réunies :

- La métrique CloudWatch `CPUUtilization` est proche de 100 %.
- La charge moyenne est supérieure au nombre de vCPU, ce qui indique une charge importante. Vous trouverez la métrique `loadAverageMinute` dans la section relative aux métriques du système d'exploitation de Performance Insights.

Causes probables de l'augmentation du nombre d'événements d'attente

Un événement d'attente UC trop fréquent peut révéler un problème de performances dont les principales causes sont les suivantes.

Rubriques

Causes probables des pics soudains

Les causes les plus probables des pics soudains sont les suivantes :

- Votre application a ouvert un trop grand nombre de connexions simultanées à la base de données. Ce scénario est connu sous le nom de « connection storm » (tempête de connexions).
- La charge de travail de votre application a connu l'un des changements suivants :
  - Nouvelles requêtes
  - Augmentation de la taille du jeu de données
  - Maintenance ou création d'index
  - Nouvelles fonctions
  - Nouveaux opérateurs
  - Augmentation des exécutions de requêtes parallèles
- Vos plans d'exécution des requêtes ont changé. Dans certains cas, un changement peut entraîner une augmentation des mémoires tampons. Par exemple, la requête utilise désormais une analyse séquentielle alors qu'elle utilisait auparavant un index. Dans ce cas, les requêtes ont besoin de plus d'UC pour atteindre le même objectif.

Causes probables d'une fréquence élevée sur le long terme

Causes les plus probables de la répétition des événements sur une longue période :

- Un trop grand nombre de processus backend s'exécutent simultanément sur l'UC. Ces processus peuvent être des employés parallèles.
- Les performances des requêtes ne sont pas optimales car elles nécessitent un grand nombre de mémoires tampons.

Cas particuliers

Si aucune des causes probables ne s'avère être la bonne, les situations suivantes peuvent se produire :

- L'UC échange des processus en entrée et en sortie.
- Le changement de contexte d'UC a augmenté.
- Il manque des événements d'attente dans le code Aurora PostgreSQL.

Actions

Si l'événement d'attente CPU domine l'activité de la base de données, cela n'indique pas nécessairement un problème de performance. Ne réagissez à cet événement qu'en cas de dégradation des performances.

Rubriques

- Vérifiez que la base de données n'est pas à l'origine de l'augmentation du nombre d'événements d'attente UC
- Déterminez si le nombre de connexions a augmenté
- Réagissez aux changements de charge de travail

Vérifiez que la base de données n'est pas à l'origine de l'augmentation du nombre d'événements d'attente UC

Examinez la métrique `os.cpuUtilization.nice.avg` dans Performance Insights. Si cette valeur est bien inférieure à l'utilisation de l'UC, cela signifie que des processus non liés à la base de données contribuent majoritairement aux événements d'attente UC.

Déterminez si le nombre de connexions a augmenté

Examinez la métrique `DatabaseConnections` dans Amazon CloudWatch. L'action à entreprendre varie selon que ce nombre augmente ou diminue pendant la période d'augmentation des événements d'attente UC.

Les connexions ont augmenté

Si le nombre de connexions a augmenté, comparez le nombre de processus backend utilisant l'UC au nombre de vCPU. Les scénarios possibles sont les suivants :

- Le nombre de processus backend utilisant l'UC est inférieur au nombre de vCPU.

  Dans ce cas, le nombre de connexions n'est pas un problème. Cependant, vous pouvez toujours essayer de réduire l'utilisation de l'UC.
- Le nombre de processus backend utilisant l'UC est supérieur au nombre de vCPU.

  Dans ce cas, procédez comme suit :
  - Réduisez le nombre de processus backend connectés à votre base de données. Par exemple, implémentez une solution de regroupement des connexions telle que RDS Proxy. Pour en savoir plus, veuillez consulter la section Proxy Amazon RDS pour Aurora.
  - Augmentez la taille de votre instance pour bénéficier d'un plus grand nombre de vCPU.
  - Redirigez certaines charges de travail en lecture seule vers des nœuds de lecture, le cas échéant.

Les connexions n'ont pas augmenté

Examinez les métriques `blks_hit` dans Performance Insights. Recherchez une corrélation entre une augmentation de `blks_hit` et l'utilisation de l'UC. Les scénarios possibles sont les suivants :

- L'utilisation de l'UC et `blks_hit` sont corrélés.

  Dans ce cas, recherchez les principales instructions SQL liées à l'utilisation de l'UC ainsi que les modifications apportées au plan. Vous pouvez utiliser l'une des techniques suivantes :
  - Décrivez les plans manuellement et comparez-les au plan d'exécution attendu.
  - Recherchez une augmentation des accès en bloc par seconde et des accès en bloc locaux par seconde. Dans la section Top SQL (Principaux éléments SQL) du tableau de bord Performance Insights, choisissez Preferences (Préférences).

- L'utilisation de l'UC et `blks_hit` ne sont pas corrélés.

  Dans ce cas, déterminez si l'une des situations suivantes se produit :

  - L'application se connecte et se déconnecte rapidement de la base de données.

    Diagnostiquez ce comportement en activant `log_connections` et `log_disconnections`, puis en analysant les journaux PostgreSQL. Pensez à utiliser l'analyseur de journaux `pgbadger`. Pour de plus amples informations, consultez https://github.com/darold/pgbadger.

  - Le système d'exploitation est surchargé.

    Dans ce cas, Performance Insights montre que les processus backend utilisent l'UC plus longtemps que d'habitude. Recherchez des preuves dans les métriques Performance Insights `os.cpuUtilization` ou dans la métrique CloudWatch `CPUUtilization`. Si le système d'exploitation est surchargé, consultez les métriques de surveillance améliorée pour approfondir le diagnostic. Plus précisément, examinez la liste des processus et le pourcentage d'UC utilisé par chaque processus.

  - Les principales instructions SQL utilisent trop d'UC.

    Examinez les instructions liées à l'utilisation de l'UC pour voir si elles peuvent en utiliser moins. Exécutez une commande `EXPLAIN` et concentrez-vous sur les nœuds du plan qui ont le plus d'impact. Utilisez un visualiseur de plan d'exécution PostgreSQL. Pour essayer cet outil, consultez http://explain.dalibo.com/.

Réagissez aux changements de charge de travail

Si votre charge de travail a changé, recherchez les types de changements suivants :

Nouvelles requêtes

Déterminez si les nouvelles requêtes sont attendues. Si oui, assurez-vous que leur plan d'exécution et le nombre d'exécutions par seconde sont attendus.

Augmentation de la taille du jeu de données

Déterminez si un partitionnement, s'il n'est pas déjà implémenté, peut être utile. Cette stratégie peut réduire le nombre de pages qu'une requête doit récupérer.

Maintenance ou création d'index

Vérifiez si le calendrier de maintenance est attendu. Une bonne pratique consiste à planifier des activités de maintenance en dehors des périodes de pointe.

Nouvelles fonctions

Déterminez si ces fonctions s'exécutent comme prévu lors des tests. Plus précisément, déterminez si le nombre d'exécutions par seconde est attendu.

Nouveaux opérateurs

Déterminez s'ils fonctionnent comme prévu lors des tests.

Augmentation du nombre de requêtes exécutées en parallèle

Déterminez si l'une des situations suivantes s'est produite :

- Les relations ou index impliqués ont soudainement augmenté en termes de taille, de sorte qu'ils sont considérablement différents de `min_parallel_table_scan_size` ou `min_parallel_index_scan_size`.
- Des modifications récentes ont été apportées à `parallel_setup_cost` ou `parallel_tuple_cost`.
- Des modifications récentes ont été apportées à `max_parallel_workers` ou `max_parallel_workers_per_gather`.

IO : BufFileRead et IO : BufFileWrite

Les événements `IO:BufFileRead` et `IO:BufFileWrite` se produisent lorsqu'Aurora PostgreSQL crée des fichiers temporaires. Lorsque des opérations requièrent plus de mémoire que n'en confèrent les paramètres de mémoire de travail définis, elles écrivent des données temporaires sur un stockage permanent. Cette opération est parfois appelée déversement sur le disque. Pour plus d'informations sur les fichiers temporaires et leur utilisation, consultez Gestion des fichiers temporaires avec PostgreSQL.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL.

Contexte

`IO:BufFileRead` et `IO:BufFileWrite` se rapportent à la zone de mémoire de travail et à la zone de mémoire des travaux de maintenance. Pour en savoir plus sur ces zones de la mémoire locale, consultez Zone de mémoire de travail et Zone de mémoire des travaux de maintenance.

La valeur par défaut du paramètre `work_mem` est 4 Mo. Si une session effectue des opérations en parallèle, chaque application de travail gérant le parallélisme utilise 4 Mo de mémoire. Par conséquent, définissez `work_mem` prudemment. Si vous augmentez trop la valeur, une base de données exécutant plusieurs sessions peut utiliser trop de mémoire. Si vous définissez une valeur trop faible, Aurora PostgreSQL crée des fichiers temporaires sur le stockage local. Le disque I/O contenant ces fichiers temporaires peut réduire les performances.

Si vous observez la séquence d'événements suivante, votre base de données génère peut-être des fichiers temporaires :

1. Diminution soudaine et brutale de la disponibilité
2. Récupération rapide de l'espace libre

Vous pouvez également observer un schéma en dents de scie. Ce schéma peut indiquer que votre base de données crée constamment de petits fichiers.

Causes probables de l'augmentation du nombre d'événements d'attente

En général, ces événements d'attente sont provoqués par des opérations qui utilisent plus de mémoire que n'en allouent les paramètres `work_mem` ou `maintenance_work_mem`. Pour compenser, les opérations écrivent dans des fichiers temporaires. Les principales causes des événements `IO:BufFileRead` et `IO:BufFileWrite` sont les suivantes :

Requêtes nécessitant plus de mémoire qu'il n'en existe dans la zone de mémoire de travail

Les requêtes présentant les caractéristiques suivantes utilisent la zone de mémoire de travail :

- Jointures par hachage
- `ORDER BY` Clause

- `GROUP BY`Clause
- `DISTINCT`
- Fonctions de fenêtrage
- `CREATE TABLE AS SELECT`
- Actualisation de la vue matérialisée

Instructions nécessitant plus de mémoire qu'il n'en existe dans la zone de mémoire des travaux de maintenance

Les instructions suivantes utilisent la zone de mémoire des travaux de maintenance :

- `CREATE INDEX`
- `CLUSTER`

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d’attente.

Rubriques

- Identifiez le problème
- Examinez vos requêtes de jointure
- Examinez vos requêtes ORDER BY et GROUP BY
- Évitez d'utiliser l'opération DISTINCT
- Envisagez d'utiliser des fonctions de fenêtrage à la place des fonctions GROUP BY
- Examinez les vues matérialisées et les instructions CTAS
- Utilisez pg_repack lorsque vous créez des index
- Augmentez maintenance_work_mem lorsque vous mettez des tables en cluster
- Réglez la mémoire pour empêcher les E/S : BufFileRead et E/S : BufFileWrite

Identifiez le problème

Vous pouvez consulter l'utilisation des fichiers temporaires directement dans Performance Insights. Pour de plus amples informations, veuillez consulter Affichage de l’utilisation des fichiers temporaires avec Performance Insights. Lorsque Performance Insights est désactivé, vous remarquerez peut-

être une augmentation `IO:BufFileRead` des `IO:BufFileWrite` opérations. Pour résoudre le problème, procédez comme suit :

1. Examinez la `FreeLocalStorage` métrique sur Amazon CloudWatch.
2. Recherchez un schéma en dents de scie.

Un schéma en dents de scie indique une consommation et une libération rapides du stockage, souvent associées à des fichiers temporaires. Si vous observez ce schéma, activez Performance Insights. Lorsque vous utilisez Performance Insights, vous pouvez identifier quand les événements d'attente se produisent et quelles requêtes y sont associées. Votre solution dépend de la requête spécifique qui est à l'origine des événements.

Ou définissez le paramètre `log_temp_files`. Ce paramètre enregistre toutes les requêtes générant un nombre de Ko de fichiers temporaires supérieur au seuil. Si la valeur est `0`, Aurora PostgreSQL enregistre tous les fichiers temporaires. Si la valeur est `1024`, Aurora PostgreSQL enregistre toutes les requêtes qui produisent des fichiers temporaires de plus de 1 Mo. Pour en savoir plus sur `log_temp_files`, consultez Error Reporting and Logging dans la documentation PostgreSQL.

Examinez vos requêtes de jointure

Votre application utilise probablement des jointures. Par exemple, la requête suivante joint quatre tables.

```
SELECT *
       FROM order
 INNER JOIN order_item
       ON (order.id = order_item.order_id)
 INNER JOIN customer
       ON (customer.id = order.customer_id)
 INNER JOIN customer_address
       ON (customer_address.customer_id = customer.id AND
           order.customer_address_id = customer_address.id)
 WHERE customer.id = 1234567890;
```

Les pics d'utilisation des fichiers temporaires peuvent être dus à un problème dans la requête proprement dite. Par exemple, une clause rompue peut ne pas filtrer correctement les jointures. Prenons la deuxième jointure interne de l'exemple suivant.

```
SELECT *
       FROM order
```

```
INNER JOIN order_item
      ON (order.id = order_item.order_id)
INNER JOIN customer
      ON (customer.id = customer.id)
INNER JOIN customer_address
      ON (customer_address.customer_id = customer.id AND
          order.customer_address_id = customer_address.id)
WHERE customer.id = 1234567890;
```

La requête précédente joint par erreur `customer.id` à `customer.id`, générant un produit cartésien entre chaque client et chaque commande. Ce type de jointure accidentelle génère des fichiers temporaires volumineux. Selon la taille des tables, une requête cartésienne peut même saturer le stockage. Votre application peut présenter des jointures cartésiennes lorsque les conditions suivantes sont réunies :

- Vous observez des baisses importantes et brutales de la disponibilité du stockage, suivies d'une récupération rapide.
- Aucun index n'est créé.
- Aucune instruction `CREATE TABLE FROM SELECT` n'est émise.
- Aucune vue matérialisée n'est actualisée.

Pour savoir si les tables sont jointes à l'aide des clés appropriées, examinez votre requête et les directives de mappage objet-relationnel. Gardez à l'esprit que certaines requêtes de votre application ne sont pas appelées en permanence, et que certaines requêtes sont générées dynamiquement.

Examinez vos requêtes ORDER BY et GROUP BY

Dans certains cas, une clause `ORDER BY` peut entraîner un nombre excessif de fichiers temporaires. Considérez les directives suivantes :

- N'incluez des colonnes dans une clause `ORDER BY` que lorsqu'elles doivent être classées. Cette directive est particulièrement importante pour les requêtes qui renvoient des milliers de lignes et spécifient de nombreuses colonnes dans la clause `ORDER BY`.
- N'hésitez pas à créer des index pour accélérer les clauses `ORDER BY` lorsqu'elles correspondent à des colonnes qui présentent le même ordre croissant ou décroissant. Les index partiels sont préférables car ils sont plus petits. Les index de petite taille sont lus et parcourus plus rapidement.
- Si vous créez des index pour des colonnes qui peuvent accepter des valeurs nulles, déterminez si vous souhaitez que les valeurs nulles soient stockées à la fin ou au début des index.

Si possible, réduisez le nombre de lignes à classer en filtrant l'ensemble de résultats. Si vous utilisez des instructions ou des sous-requêtes liées à la clause `WITH`, n'oubliez pas qu'une requête interne génère un ensemble de résultats et le transmet à la requête externe. Plus le nombre de lignes qu'une requête peut filtrer est élevé, moins elle a de classement à effectuer.

- Si vous n'avez pas besoin de l'ensemble de résultats complet, utilisez la clause `LIMIT`. Par exemple, si vous avez uniquement besoin des cinq premières lignes, une requête utilisant la clause `LIMIT` ne continue pas à générer des résultats. La requête a ainsi besoin de moins de mémoire et de moins de fichiers temporaires.

Une requête qui utilise une clause `GROUP BY` peut également avoir besoin de fichiers temporaires. Les requêtes `GROUP BY` résument les valeurs à l'aide de fonctions telles que les suivantes :

- `COUNT`
- `AVG`
- `MIN`
- `MAX`
- `SUM`
- `STDDEV`

Pour régler les requêtes `GROUP BY`, suivez les recommandations relatives aux requêtes `ORDER BY`.

Évitez d'utiliser l'opération DISTINCT

Dans la mesure du possible, évitez d'utiliser l'opération `DISTINCT` pour supprimer les lignes en double. Plus votre requête renvoie de lignes inutiles et en double, plus l'opération `DISTINCT` devient coûteuse. Si possible, ajoutez des filtres dans la clause `WHERE`, même si vous utilisez les mêmes filtres pour différentes tables. Un filtrage de la requête et une jointure correctes vous permettent d'améliorer les performances et de réduire l'utilisation des ressources. Ils vous permettent également d'éviter les rapports et les résultats incorrects.

Si vous devez utiliser `DISTINCT` pour plusieurs lignes d'une même table, n'hésitez pas à créer un index composite. Le regroupement de plusieurs colonnes dans un index peut améliorer le temps nécessaire à l'évaluation des lignes distinctes. En outre, si vous utilisez Amazon Aurora PostgreSQL 10 ou version ultérieure, vous pouvez corréler les statistiques entre plusieurs colonnes à l'aide de la commande `CREATE STATISTICS`.

Envisagez d'utiliser des fonctions de fenêtrage à la place des fonctions GROUP BY

Avec GROUP BY, vous modifiez l'ensemble de résultats, puis récupérez le résultat agrégé. Avec les fonctions de fenêtrage, vous pouvez agréger les données sans modifier l'ensemble de résultats. Une fonction de fenêtrage utilise la clause OVER pour effectuer des calculs sur les ensembles définis par la requête, en corrélant une ligne avec une autre. Les fonctions de fenêtrage vous permettent d'utiliser toutes les fonctions GROUP BY ainsi que les fonctions suivantes :

- RANK
- ARRAY_AGG
- ROW_NUMBER
- LAG
- LEAD

Pour réduire le nombre de fichiers temporaires générés par une fonction de fenêtrage, supprimez les doublons d'un même ensemble de résultats lorsque vous avez besoin de deux agrégations distinctes. Considérons la requête suivante :

```
SELECT sum(salary) OVER (PARTITION BY dept ORDER BY salary DESC) as sum_salary
     , avg(salary) OVER (PARTITION BY dept ORDER BY salary ASC) as avg_salary
  FROM empsalary;
```

Vous pouvez réécrire la requête en utilisant la clause WINDOW comme suit.

```
SELECT sum(salary) OVER w as sum_salary
         , avg(salary) OVER w as_avg_salary
    FROM empsalary
  WINDOW w AS (PARTITION BY dept ORDER BY salary DESC);
```

Par défaut, le planificateur d'exécution Aurora PostgreSQL regroupe les nœuds similaires afin de ne pas dupliquer les opérations. Toutefois, en utilisant une déclaration explicite pour le bloc de fenêtres, vous pouvez gérer la requête plus facilement. Vous pouvez également améliorer les performances en empêchant la duplication.

Examinez les vues matérialisées et les instructions CTAS

Lorsqu'une vue matérialisée est actualisée, elle exécute une requête. Cette requête peut contenir une opération telle que GROUP BY, ORDER BY ou DISTINCT. Lors d'une actualisation, vous pouvez

observer un grand nombre de fichiers temporaires et les événements d'attente `IO:BufFileWrite` et `IO:BufFileRead`. De même, lorsque vous créez une table basée sur une instruction `SELECT`, l'instruction `CREATE TABLE` exécute une requête. Pour réduire le nombre de fichiers temporaires nécessaires, optimisez la requête.

Utilisez pg_repack lorsque vous créez des index

Lorsque vous créez un index, le moteur classe l'ensemble de résultats. À mesure que la taille des tables augmente et que les valeurs de la colonne indexée se diversifient, les fichiers temporaires ont besoin de plus d'espace. Dans la plupart des cas, vous ne pouvez pas empêcher la création de fichiers temporaires pour les tables volumineuses sans modifier la zone de mémoire des travaux de maintenance. Pour de plus amples informations, veuillez consulter Zone de mémoire des travaux de maintenance.

Une solution de contournement possible lors de la recréation d'un index volumineux consiste à utiliser l'outil pg_repack. Pour en savoir plus, consultez Reorganize tables in PostgreSQL databases with minimal locks dans la documentation pg_repack.

Augmentez maintenance_work_mem lorsque vous mettez des tables en cluster

La commande `CLUSTER` met en cluster la table spécifiée par table_name à partir d'un index existant spécifié par index_name. Aurora PostgreSQL recrée physiquement la table en suivant l'ordre d'un index donné.

Lorsque le stockage magnétique était prédominant, la mise en cluster était courante car le débit de stockage était limité. Maintenant que le stockage SSD est plus répandu, la mise en cluster est moins fréquente. Toutefois, en mettant des tables en cluster, vous pouvez encore bénéficier d'une légère amélioration des performances en fonction de la taille de la table, de l'index, de la requête, etc.

Si vous exécutez la commande `CLUSTER` et observez les événements d'attente `IO:BufFileWrite` et `IO:BufFileRead`, réglez `maintenance_work_mem`. Augmentez la taille de la mémoire en la définissant sur une valeur relativement élevée. Une valeur élevée permettra au moteur d'utiliser davantage de mémoire pour l'opération de mise en cluster.

Réglez la mémoire pour empêcher les E/S : BufFileRead et E/S : BufFileWrite

Dans certaines situations, vous devez régler la mémoire. Votre objectif est de trouver un équilibre par rapport aux exigences suivantes :

- Valeur de `work_mem` (voir Zone de mémoire de travail)

- Mémoire restante après déduction de la valeur `shared_buffers` (voir Groupe de mémoires tampons)
- Nombre maximal de connexions ouvertes et en cours d'utilisation, qui est limité par `max_connections`

Augmentez la taille de la zone de mémoire de travail

Dans certains cas, votre seule option consiste à augmenter la mémoire utilisée par votre session. Si vos requêtes sont correctement écrites et utilisent les bonnes clés pour les jointures, augmentez la valeur `work_mem`. Pour de plus amples informations, veuillez consulter Zone de mémoire de travail.

Pour savoir combien de fichiers temporaires une requête génère, définissez `log_temp_files` sur `0`. Si vous définissez la valeur `work_mem` sur la valeur maximale identifiée dans les journaux, vous empêchez la requête de générer des fichiers temporaires. Toutefois, `work_mem` définit le maximum par nœud du plan pour chaque connexion ou application de travail parallèle. Si la base de données compte 5 000 connexions, et si chacune d'entre elles utilise 256 Mio de mémoire, le moteur a besoin de 1,2 Tio de RAM. Votre instance risque donc de manquer de mémoire.

Réservez suffisamment de mémoire pour le groupe de mémoires tampons partagées

Votre base de données utilise des zones de mémoire telles que le groupe de mémoires tampons partagées, et pas seulement la zone de mémoire de travail. Prenez en compte les besoins de ces zones de mémoire supplémentaires avant d'augmenter `work_mem`. Pour en savoir plus sur le groupe de mémoires tampons, consultez Groupe de mémoires tampons.

Par exemple, supposons que votre classe d'instance Aurora PostgreSQL soit db.r5.2xlarge. Cette classe dispose de 64 Gio de mémoire. Par défaut, 75 % de la mémoire est réservée au groupe de mémoires tampons partagées. Après avoir soustrait la quantité allouée à la zone de mémoire partagée, il reste 16 384 Mo. Évitez d'allouer la mémoire restante exclusivement à la zone de mémoire de travail, car le système d'exploitation et le moteur ont également besoin de mémoire.

La mémoire que vous pouvez allouer à `work_mem` dépend de la classe d'instance. Si vous utilisez une classe d'instance plus importante, vous disposerez de plus de mémoire. Toutefois, dans l'exemple précédent, vous ne pouvez pas utiliser plus de 16 Gio. Sinon votre instance devient indisponible lorsqu'elle est à court de mémoire. Pour récupérer l'instance en cas d'indisponibilité, les services d'automatisation d'Aurora PostgreSQL redémarrent automatiquement.

Gérez le nombre de connexions

Supposons que votre instance de base de données compte 5 000 connexions simultanées. Chaque connexion utilise au moins 4 Mio de `work_mem`. La forte consommation de mémoire des connexions est susceptible de dégrader les performances. En réponse, vous disposez des options suivantes :

- Passez à une classe d'instance supérieure.
- Réduisez le nombre de connexions simultanées à la base de données à l'aide d'un proxy ou d'un regroupement de connexions.

Pour les proxies, utilisez Amazon RDS Proxy, PgBouncer ou un regroupement de connexions basé sur votre application. Cette solution réduit la charge de l'UC. Elle réduit également le risque lorsque toutes les connexions ont besoin de la zone de mémoire de travail. Lorsque les connexions à la base de données sont moins nombreuses, vous pouvez augmenter la valeur de `work_mem`. De cette façon, vous réduisez l'occurrence des événements d'attente `IO:BufFileRead` et `IO:BufFileWrite`. De plus, les requêtes en attente d'accès à la zone de mémoire de travail s'accélèrent de manière significative.

IO:DataFileRead

L'événement `IO:DataFileRead` se produit lorsqu'une connexion attend qu'un processus backend lise une page requise à partir du stockage parce que la page n'est pas disponible dans la mémoire partagée.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL.

Contexte

Toutes les requêtes et opérations en langage de manipulation de données (DML) accèdent aux pages du groupe de mémoires tampons. Les instructions qui peuvent induire des lectures sont : `SELECT`, `UPDATE` et `DELETE`. Par exemple, une instruction `UPDATE` peut lire des pages à partir de tables ou d'index. Si la page demandée ou mise à jour ne se trouve pas dans le groupe de mémoires tampons partagées, cette lecture peut provoquer l'événement `IO:DataFileRead`.

Comme le groupe de mémoires tampons partagées est limité, il peut être saturé. Dans ce cas, les requêtes de pages qui ne sont pas en mémoire forcent la base de données à lire des blocs sur le disque. Si l'événement `IO:DataFileRead` se produit fréquemment, votre groupe de mémoires tampons partagées est peut-être trop petit pour prendre en charge votre charge de travail. Ce problème se pose avec acuité pour les requêtes `SELECT` qui lisent un grand nombre de lignes qui ne rentrent pas dans le groupe de mémoires tampons. Pour en savoir plus sur le groupe de mémoires tampons, consultez Groupe de mémoires tampons.

Causes probables de l'augmentation du nombre d'événements d'attente

Les principales causes de l'événement `IO:DataFileRead` sont les suivantes :

Pics de connexion

Plusieurs connexions peuvent générer le même nombre d'événements d'attente IO:DataFileRead. Dans ce cas, un pic (augmentation soudaine et importante) d'événements `IO:DataFileRead` peut se produire.

Instructions SELECT et DML effectuant des analyses séquentielles

Votre application est peut-être en train d'effectuer une nouvelle opération. Ou une opération existante peut changer suite à un nouveau plan d'exécution. Dans ce cas, recherchez les tables (en particulier les tables volumineuses) qui présentent une valeur `seq_scan` plus élevée. Pour les trouver, interrogez `pg_stat_user_tables`. Pour suivre les requêtes qui génèrent plus d'opérations de lecture, utilisez l'extension `pg_stat_statements`.

CTAS et CREATE INDEX pour les jeux de données volumineux

Un CTAS est une instruction `CREATE TABLE AS SELECT`. Si vous exécutez un CTAS en utilisant un jeu de données volumineux comme source, ou si vous créez un index sur une table volumineuse, l'événement `IO:DataFileRead` peut se produire. Lorsque vous créez un index, la base de données peut avoir besoin de lire l'objet entier à l'aide d'une analyse séquentielle. Un CTAS génère des lectures `IO:DataFile` lorsque les pages ne sont pas en mémoire.

Exécution simultanée de plusieurs processus employés vacuum

Les processus employés vacuum peuvent être déclenchés manuellement ou automatiquement. Nous vous recommandons d'adopter une stratégie vacuum agressive. Toutefois, lorsqu'une table comporte de nombreuses lignes mises à jour ou supprimées, le nombre d'attentes `IO:DataFileRead` augmente. Une fois l'espace récupéré, le temps vacuum passé sur `IO:DataFileRead` diminue.

Ingestion de grandes quantités de données

Lorsque votre application ingère de grandes quantités de données, les opérations `ANALYZE` peuvent être plus fréquentes. Le processus `ANALYZE` peut être déclenché par un lanceur autovacuum, ou être appelé manuellement.

L'opération `ANALYZE` lit un sous-ensemble de la table. Le nombre de pages à analyser est calculé en multipliant 30 par la valeur `default_statistics_target`. Pour de plus amples informations, veuillez consulter la documentation sur PostgreSQL. Le paramètre `default_statistics_target` accepte des valeurs comprises entre 1 et 10 000, la valeur par défaut étant 100.

Pénurie de ressources

Si l'UC ou la bande passante réseau de l'instance sont entièrement utilisés, l'événement `IO:DataFileRead` peut se produire plus fréquemment.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Vérifiez les filtres de prédicat pour détecter les requêtes qui génèrent des attentes
- Minimisez l'effet des opérations de maintenance
- Réagissez à un nombre élevé de connexions

Vérifiez les filtres de prédicat pour détecter les requêtes qui génèrent des attentes

Supposons que vous identifiez des requêtes spécifiques qui génèrent des événements d'attente `IO:DataFileRead`. Vous pouvez les identifier à l'aide des techniques suivantes :

- Performance Insights
- Vues catalogue telles que celles fournies par l'extension `pg_stat_statements`
- Vue catalogue `pg_stat_all_tables`, si elle affiche périodiquement un nombre accru de lectures physiques
- Vue `pg_statio_all_tables`, si elle montre que les compteurs `_read` sont en augmentation

Nous vous recommandons de déterminer quels filtres sont utilisés dans le prédicat (clause `WHERE`) de ces requêtes. Suivez ces instructions :

- Exécutez la commande `EXPLAIN`. Dans la sortie, identifiez les types d'analyses utilisés. Une analyse séquentielle n'indique pas nécessairement un problème. Les requêtes qui utilisent des analyses séquentielles produisent naturellement plus d'événements `IO:DataFileRead` par rapport aux requêtes qui utilisent des filtres.

  Vérifiez que la colonne répertoriée dans la clause `WHERE` est indexée. Si ce n'est pas le cas, envisagez de créer un index pour cette colonne. Cette approche évite les analyses séquentielles et réduit le nombre d'événements `IO:DataFileRead`. Si une requête comporte des filtres restrictifs et continue à produire des analyses séquentielles, assurez-vous que les index appropriés sont utilisés.
- Déterminez si la requête accède à une table très volumineuse. Dans certains cas, le partitionnement d'une table peut améliorer les performances, en permettant à la requête de ne lire que les partitions nécessaires.
- Examinez la cardinalité (nombre total de lignes) de vos opérations de jointure. Notez le caractère restrictif des valeurs que vous transmettez dans les filtres de la clause `WHERE`. Si possible, ajustez votre requête pour réduire le nombre de lignes transmises à chaque étape du plan.

Minimisez l'effet des opérations de maintenance

Les opérations de maintenance telles que `VACUUM` et `ANALYZE` sont importantes. Nous vous recommandons de ne pas les désactiver parce que vous trouvez des événements d'attente `IO:DataFileRead` liés à ces opérations de maintenance. Les approches suivantes peuvent minimiser l'effet de ces opérations :

- Exécutez les opérations de maintenance manuellement pendant les heures creuses. Cette technique empêche la base de données d'atteindre le seuil des opérations automatiques.

- Pour les tables très volumineuses, partitionnez la table. Cette technique permet de réduire les frais liés aux opérations de maintenance. La base de données accède uniquement aux partitions qui nécessitent une maintenance.
- Lorsque vous intégrez de grandes quantités de données, pensez à désactiver la fonction d'analyse automatique.

La fonction autovacuum est automatiquement déclenchée pour une table lorsque la formule suivante est vraie.

```
pg_stat_user_tables.n_dead_tup > (pg_class.reltuples x autovacuum_vacuum_scale_factor)
 + autovacuum_vacuum_threshold
```

La vue `pg_stat_user_tables` et le catalogue `pg_class` comportent plusieurs lignes. Une ligne peut correspondre à une ligne de votre table. Cette formule suppose que les `reltuples` sont destinés à une table spécifique. Les paramètres `autovacuum_vacuum_scale_factor` (0,20 par défaut) et `autovacuum_vacuum_threshold` (50 tuples par défaut) sont généralement définis globalement pour l'ensemble de l'instance. Vous pouvez toutefois définir des valeurs différentes pour une table spécifique.

Rubriques

- Recherche des tables qui consomment de l'espace inutilement
- Recherchez les index qui consomment inutilement de l'espace
- Recherchez les tables éligibles au processus autovacuum

Recherche des tables qui consomment de l'espace inutilement

Pour trouver les tables consommant plus d'espace que nécessaire, exécutez la requête suivante. Lorsque cette requête est exécutée par un rôle d'utilisateur de base de données qui n'a pas le rôle `rds_superuser`, elle renvoie des informations sur les seules tables pour lesquelles le rôle de l'utilisateur détient les autorisations de lecture. Cette requête est prise en charge par PostgreSQL version 12 et versions ultérieures.

```
WITH report AS (
   SELECT   schemaname
           ,tblname
           ,n_dead_tup
           ,n_live_tup
```

```
           ,block_size*tblpages AS real_size
           ,(tblpages-est_tblpages)*block_size AS extra_size
           ,CASE WHEN tblpages - est_tblpages > 0
              THEN 100 * (tblpages - est_tblpages)/tblpages::float
              ELSE 0
            END AS extra_ratio, fillfactor, (tblpages-est_tblpages_ff)*block_size AS
 bloat_size
           ,CASE WHEN tblpages - est_tblpages_ff > 0
              THEN 100 * (tblpages - est_tblpages_ff)/tblpages::float
              ELSE 0
            END AS bloat_ratio
           ,is_na
    FROM (
           SELECT  ceil( reltuples / ( (block_size-page_hdr)/tpl_size ) ) +
 ceil( toasttuples / 4 ) AS est_tblpages
                  ,ceil( reltuples / ( (block_size-page_hdr)*fillfactor/
(tpl_size*100) ) ) + ceil( toasttuples / 4 ) AS est_tblpages_ff
                  ,tblpages
                  ,fillfactor
                  ,block_size
                  ,tblid
                  ,schemaname
                  ,tblname
                  ,n_dead_tup
                  ,n_live_tup
                  ,heappages
                  ,toastpages
                  ,is_na
             FROM (
                    SELECT ( 4 + tpl_hdr_size + tpl_data_size + (2*ma)
                               - CASE WHEN tpl_hdr_size%ma = 0 THEN ma ELSE
 tpl_hdr_size%ma END
                               - CASE WHEN ceil(tpl_data_size)::int%ma = 0 THEN ma ELSE
 ceil(tpl_data_size)::int%ma END
                           ) AS tpl_size
                           ,block_size - page_hdr AS size_per_block
                           ,(heappages + toastpages) AS tblpages
                           ,heappages
                           ,toastpages
                           ,reltuples
                           ,toasttuples
                           ,block_size
                           ,page_hdr
                           ,tblid
```

```
                            ,schemaname
                            ,tblname
                            ,fillfactor
                            ,is_na
                            ,n_dead_tup
                            ,n_live_tup
                           FROM (
                                SELECT  tbl.oid                       AS tblid
                                       ,ns.nspname                    AS schemaname
                                       ,tbl.relname                   AS tblname
                                       ,tbl.reltuples                 AS reltuples
                                       ,tbl.relpages                  AS heappages
                                       ,coalesce(toast.relpages, 0)   AS toastpages
                                       ,coalesce(toast.reltuples, 0)  AS toasttuples
                                       ,psat.n_dead_tup               AS n_dead_tup
                                       ,psat.n_live_tup               AS n_live_tup
                                       ,24                            AS page_hdr
                                       ,current_setting('block_size')::numeric AS
 block_size

 ,coalesce(substring( array_to_string(tbl.reloptions, ' ') FROM
 'fillfactor=([0-9]+)')::smallint, 100) AS fillfactor
                                       ,CASE WHEN version()~'mingw32' OR version()~'64-
bit|x86_64|ppc64|ia64|amd64' THEN 8 ELSE 4 END        AS ma
                                       ,23 + CASE WHEN MAX(coalesce(null_frac,0)) > 0
 THEN ( 7 + count(*) ) / 8 ELSE 0::int END              AS tpl_hdr_size
                                       ,sum( (1-coalesce(s.null_frac, 0)) *
 coalesce(s.avg_width, 1024) )                                  AS tpl_data_size
                                       ,bool_or(att.atttypid =
 'pg_catalog.name'::regtype) OR count(att.attname) <> count(s.attname)         AS is_na
                                  FROM  pg_attribute       AS att
                                  JOIN  pg_class           AS tbl    ON (att.attrelid =
 tbl.oid)
                                  JOIN  pg_stat_all_tables AS psat   ON (tbl.oid =
 psat.relid)
                                  JOIN  pg_namespace       AS ns     ON (ns.oid =
 tbl.relnamespace)
                               LEFT JOIN  pg_stats          AS s      ON
 (s.schemaname=ns.nspname AND s.tablename = tbl.relname AND s.inherited=false AND
 s.attname=att.attname)
                               LEFT JOIN  pg_class          AS toast  ON
 (tbl.reltoastrelid = toast.oid)
                                 WHERE  att.attnum > 0
                                   AND  NOT att.attisdropped
```

```
                                    AND  tbl.relkind = 'r'
                                GROUP BY  tbl.oid, ns.nspname, tbl.relname,
 tbl.reltuples, tbl.relpages, toastpages, toasttuples, fillfactor, block_size, ma,
 n_dead_tup, n_live_tup
                                ORDER BY  schemaname, tblname
                              ) AS s
                  ) AS s2
       ) AS s3
 ORDER BY bloat_size DESC
)
  SELECT *
    FROM report
   WHERE bloat_ratio != 0
 -- AND schemaname = 'public'
 -- AND tblname = 'pgbench_accounts'
;

-- WHERE NOT is_na
--   AND tblpages*((pst).free_percent + (pst).dead_tuple_percent)::float4/100 >= 1
```

Vous pouvez vérifier qu'il n'existe pas de gonflement de tables et d'index dans votre application. Pour de plus amples informations, consultez Diagnostic du gonflement de la table et de l'index.

Recherchez les index qui consomment inutilement de l'espace

Pour rechercher les index qui consomment inutilement de l'espace, exécutez la requête suivante.

```
-- WARNING: run with a nonsuperuser role, the query inspects
-- only indexes on tables you have permissions to read.
-- WARNING: rows with is_na = 't' are known to have bad statistics ("name" type is not
 supported).
-- This query is compatible with PostgreSQL 8.2 and later.

SELECT current_database(), nspname AS schemaname, tblname, idxname,
 bs*(relpages)::bigint AS real_size,
  bs*(relpages-est_pages)::bigint AS extra_size,
  100 * (relpages-est_pages)::float / relpages AS extra_ratio,
  fillfactor, bs*(relpages-est_pages_ff) AS bloat_size,
  100 * (relpages-est_pages_ff)::float / relpages AS bloat_ratio,
  is_na
  -- , 100-(sub.pst).avg_leaf_density, est_pages, index_tuple_hdr_bm,
  -- maxalign, pagehdr, nulldatawidth, nulldatahdrwidth, sub.reltuples, sub.relpages
  -- (DEBUG INFO)
FROM (
```

```
  SELECT coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)/(4+nulldatahdrwidth)::float)), 0
       -- ItemIdData size + computed avg size of a tuple (nulldatahdrwidth)
    ) AS est_pages,
    coalesce(1 +
       ceil(reltuples/floor((bs-pageopqdata-pagehdr)*fillfactor/
(100*(4+nulldatahdrwidth)::float))), 0
    ) AS est_pages_ff,
    bs, nspname, table_oid, tblname, idxname, relpages, fillfactor, is_na
    -- , stattuple.pgstatindex(quote_ident(nspname)||'.'||quote_ident(idxname)) AS
 pst,
    -- index_tuple_hdr_bm, maxalign, pagehdr, nulldatawidth, nulldatahdrwidth,
 reltuples
    -- (DEBUG INFO)
  FROM (
    SELECT maxalign, bs, nspname, tblname, idxname, reltuples, relpages, relam,
 table_oid, fillfactor,
      ( index_tuple_hdr_bm +
          maxalign - CASE -- Add padding to the index tuple header to align on MAXALIGN
            WHEN index_tuple_hdr_bm%maxalign = 0 THEN maxalign
            ELSE index_tuple_hdr_bm%maxalign
          END
        + nulldatawidth + maxalign - CASE -- Add padding to the data to align on
 MAXALIGN
            WHEN nulldatawidth = 0 THEN 0
            WHEN nulldatawidth::integer%maxalign = 0 THEN maxalign
            ELSE nulldatawidth::integer%maxalign
          END
      )::numeric AS nulldatahdrwidth, pagehdr, pageopqdata, is_na
      -- , index_tuple_hdr_bm, nulldatawidth -- (DEBUG INFO)
    FROM (
      SELECT
        i.nspname, i.tblname, i.idxname, i.reltuples, i.relpages, i.relam, a.attrelid
 AS table_oid,
        current_setting('block_size')::numeric AS bs, fillfactor,
        CASE -- MAXALIGN: 4 on 32bits, 8 on 64bits (and mingw32 ?)
          WHEN version() ~ 'mingw32' OR version() ~ '64-bit|x86_64|ppc64|ia64|amd64'
 THEN 8
          ELSE 4
        END AS maxalign,
        /* per page header, fixed size: 20 for 7.X, 24 for others */
        24 AS pagehdr,
        /* per page btree opaque data */
        16 AS pageopqdata,
```

```
       /* per tuple header: add IndexAttributeBitMapData if some cols are null-able */
       CASE WHEN max(coalesce(s.null_frac,0)) = 0
         THEN 2 -- IndexTupleData size
         ELSE 2 + (( 32 + 8 - 1 ) / 8)
         -- IndexTupleData size + IndexAttributeBitMapData size ( max num filed per
 index + 8 - 1 /8)
       END AS index_tuple_hdr_bm,
       /* data len: we remove null values save space using it fractionnal part from
 stats */
       sum( (1-coalesce(s.null_frac, 0)) * coalesce(s.avg_width, 1024)) AS
 nulldatawidth,
       max( CASE WHEN a.atttypid = 'pg_catalog.name'::regtype THEN 1 ELSE 0 END ) > 0
 AS is_na
     FROM pg_attribute AS a
       JOIN (
         SELECT nspname, tbl.relname AS tblname, idx.relname AS idxname,
           idx.reltuples, idx.relpages, idx.relam,
           indrelid, indexrelid, indkey::smallint[] AS attnum,
           coalesce(substring(
             array_to_string(idx.reloptions, ' ')
              from 'fillfactor=([0-9]+)')::smallint, 90) AS fillfactor
         FROM pg_index
           JOIN pg_class idx ON idx.oid=pg_index.indexrelid
           JOIN pg_class tbl ON tbl.oid=pg_index.indrelid
           JOIN pg_namespace ON pg_namespace.oid = idx.relnamespace
         WHERE pg_index.indisvalid AND tbl.relkind = 'r' AND idx.relpages > 0
       ) AS i ON a.attrelid = i.indexrelid
       JOIN pg_stats AS s ON s.schemaname = i.nspname
         AND ((s.tablename = i.tblname AND s.attname =
 pg_catalog.pg_get_indexdef(a.attrelid, a.attnum, TRUE))
         -- stats from tbl
         OR  (s.tablename = i.idxname AND s.attname = a.attname))
         -- stats from functionnal cols
       JOIN pg_type AS t ON a.atttypid = t.oid
     WHERE a.attnum > 0
     GROUP BY 1, 2, 3, 4, 5, 6, 7, 8, 9
   ) AS s1
  ) AS s2
    JOIN pg_am am ON s2.relam = am.oid WHERE am.amname = 'btree'
) AS sub
-- WHERE NOT is_na
ORDER BY 2,3,4;
```

Recherchez les tables éligibles au processus autovacuum

Pour rechercher les tables éligibles au processus autovacuum, exécutez la requête suivante.

```
--This query shows tables that need vacuuming and are eligible candidates.
--The following query lists all tables that are due to be processed by autovacuum.
-- During normal operation, this query should return very little.
WITH  vbt AS (SELECT setting AS autovacuum_vacuum_threshold
              FROM pg_settings WHERE name = 'autovacuum_vacuum_threshold')
    , vsf AS (SELECT setting AS autovacuum_vacuum_scale_factor
              FROM pg_settings WHERE name = 'autovacuum_vacuum_scale_factor')
    , fma AS (SELECT setting AS autovacuum_freeze_max_age
              FROM pg_settings WHERE name = 'autovacuum_freeze_max_age')
    , sto AS (SELECT opt_oid, split_part(setting, '=', 1) as param,
                split_part(setting, '=', 2) as value
              FROM (SELECT oid opt_oid, unnest(reloptions) setting FROM pg_class) opt)
SELECT
    '"'||ns.nspname||'"."'||c.relname||'"' as relation
    , pg_size_pretty(pg_table_size(c.oid)) as table_size
    , age(relfrozenxid) as xid_age
    , coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
 autovacuum_freeze_max_age
    , (coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
         coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) *
 c.reltuples)
         as autovacuum_vacuum_tuples
    , n_dead_tup as dead_tuples
FROM pg_class c
JOIN pg_namespace ns ON ns.oid = c.relnamespace
JOIN pg_stat_all_tables stat ON stat.relid = c.oid
JOIN vbt on (1=1)
JOIN vsf ON (1=1)
JOIN fma on (1=1)
LEFT JOIN sto cvbt ON cvbt.param = 'autovacuum_vacuum_threshold' AND c.oid =
 cvbt.opt_oid
LEFT JOIN sto cvsf ON cvsf.param = 'autovacuum_vacuum_scale_factor' AND c.oid =
 cvsf.opt_oid
LEFT JOIN sto cfma ON cfma.param = 'autovacuum_freeze_max_age' AND c.oid = cfma.opt_oid
WHERE c.relkind = 'r'
AND nspname <> 'pg_catalog'
AND (
    age(relfrozenxid) >= coalesce(cfma.value::float, autovacuum_freeze_max_age::float)
    or
    coalesce(cvbt.value::float, autovacuum_vacuum_threshold::float) +
```

```
      coalesce(cvsf.value::float,autovacuum_vacuum_scale_factor::float) * c.reltuples
 <= n_dead_tup
    -- or 1 = 1
)
ORDER BY age(relfrozenxid) DESC;
```

Réagissez à un nombre élevé de connexions

En surveillant Amazon CloudWatch, vous constaterez peut-être que la métrique `DatabaseConnections` affiche un pic. Cette augmentation indique un nombre accru de connexions à votre base de données. Nous vous recommandons l'approche suivante :

- Limitez le nombre de connexions que l'application peut ouvrir avec chaque instance. Si votre application dispose d'une fonction intégrée de regroupement des connexions, définissez-la sur un nombre raisonnable de connexions. Basez ce nombre sur ce que les vCPU de votre instance sont en mesure de paralléliser.

  Si votre application n'utilise pas de fonction de regroupement des connexions, envisagez d'utiliser un proxy Amazon RDS ou une autre solution. Cette approche permet à votre application d'ouvrir plusieurs connexions à l'aide de l'équilibreur de charge. L'équilibreur peut alors ouvrir un nombre restreint de connexions avec la base de données. Comme le nombre de connexions exécutées en parallèle est moindre, votre instance de base de données effectue moins de changements de contexte dans le noyau. Les requêtes progressent plus rapidement, ce qui entraîne une diminution des événements d'attente. Pour de plus amples informations, consultez Proxy Amazon RDS pour Aurora.
- Dans la mesure du possible, tirez parti des nœuds de lecture pour Aurora PostgreSQL et des réplicas en lecture pour RDS pour PostgreSQL. Lorsque votre application exécute une opération en lecture seule, envoyez ces requêtes au point de terminaison en lecture seule. Cette technique permet de répartir les requêtes de l'application sur tous les nœuds de lecture, réduisant ainsi la pression des I/O sur le nœud d'écriture.
- Envisagez une augmentation d'échelle de votre instance de base de données. Une classe d'instance de plus grande capacité offre davantage de mémoire, ce qui permet à Aurora PostgreSQL de disposer d'un groupe de mémoires tampons partagées plus important pour contenir les pages. Cette plus grande taille confère également à l'instance de base de données davantage de vCPU pour gérer les connexions. Ce plus grand nombre de vCPU est particulièrement utile lorsque les opérations qui génèrent des événements d'attente `IO:DataFileRead` sont des écritures.

IO:XactSync

L'événement `IO:XactSync` se produit lorsque la base de données attend que le sous-système de stockage Aurora confirme la validation d'une transaction standard, ou la validation ou restauration d'une transaction préparée. Une transaction préparée fait partie de la prise en charge d'une validation en deux phases par PostgreSQL. Cet événement peut également se produire lorsqu'une requête attend la validation d'une autre transaction, en particulier dans les cas où la validation automatique est désactivée. Dans de tels scénarios, les mises à jour peuvent sembler attendre XactSync même si elles n'ont pas encore été validées.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL.

Contexte

L'événement `IO:XactSync` indique que l'instance passe du temps à attendre que le sous-système de stockage Aurora confirme que les données de transaction ont été traitées.

Causes probables de l'augmentation du nombre d'événements d'attente

Un événement `IO:XactSync` trop fréquent peut révéler un problème de performances dont les causes sont généralement les suivantes :

Saturation du réseau

Le trafic entre les clients et l'instance de base de données ou le trafic vers le sous-système de stockage peut être trop important pour la bande passante réseau.

Pression exercée sur l'UC

Une charge de travail importante peut empêcher le démon de stockage Aurora d'obtenir suffisamment de temps UC.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Surveillez vos ressources
- Augmentation verticale de l'UC
- Augmentez la bande passante réseau
- Réduisez le nombre de validations

Surveillez vos ressources

Pour déterminer la cause de l'augmentation du nombre d'événements `IO:XactSync`, vérifiez les métriques suivantes :

- `WriteThroughput` et `CommitThroughput` – Les changements liés au débit d'écriture ou au débit de validation peuvent indiquer une augmentation de la charge de travail.
- `WriteLatency` et `CommitLatency` – Les changements liés à la latence d'écriture ou à la latence de validation peuvent indiquer une sollicitation accrue du sous-système de stockage.
- `CPUUtilization` – Si l'utilisation de l'UC de l'instance est supérieure à 90 %, le démon de stockage Aurora ne dispose peut-être pas de suffisamment de temps sur l'UC. Dans ce cas, les performances d'I/O se dégradent.

Pour en savoir plus sur ces métriques, consultez Métriques de niveau instance pour Amazon Aurora.

Augmentation verticale de l'UC

Pour résoudre les problèmes de pénurie liés à l'UC, passez à un type d'instance qui offre une plus grande capacité d'UC. Pour en savoir plus sur la capacité d'UC pour une classe d'instance de base de données, consultez Spécifications matérielles pour les classes d'instance de base de données pour Aurora.

Augmentez la bande passante réseau

Pour déterminer si l'instance atteint les limites de sa bande passante réseau, vérifiez les autres événements d'attente suivants :

- `IO:DataFileRead`, `IO:BufferRead`, `IO:BufferWrite` et `IO:XactWrite` – Les requêtes utilisant de grandes quantités d'I/O peuvent générer davantage d'événements d'attente de ce type.
- `Client:ClientRead` et `Client:ClientWrite` – Les requêtes associées à énormément de communication client peuvent générer davantage d'événements d'attente de ce type.

En cas de problème de bande passante réseau, passez à un type d'instance qui offre plus de bande passante réseau. Pour en savoir plus sur les performances réseau d'une classe d'instance de base de données, consultez Spécifications matérielles pour les classes d'instance de base de données pour Aurora.

Réduisez le nombre de validations

Pour réduire le nombre de validations, combinez les instructions en blocs de transactions.

IPC:DamRecordTxAck

L'événement `IPC:DamRecordTxAck` se produit lorsqu'Aurora PostgreSQL, dans une session utilisant des flux d'activité de base de données, génère un événement de flux d'activité, puis attend que cet événement devienne durable.

Rubriques

- Versions de moteur pertinentes
- Contexte
- Causes
- Actions

Versions de moteur pertinentes

Ces informations sur les événements d'attente s'appliquent à Aurora PostgreSQL 10.7 et aux versions 10 ultérieures, à Aurora PostgreSQL 11.4 et aux versions 11 ultérieures, ainsi qu'à toutes les versions 12 et 13.

Contexte

En mode synchrone, la durabilité des événements de flux d'activité est privilégiée par rapport aux performances de la base de données. En attendant une écriture durable de l'événement,

la session bloque d'autres activités de base de données, ce qui provoque l'événement d'attente `IPC:DamRecordTxAck`.

Causes

L'événement d'attente `IPC:DamRecordTxAck` est généralement dû au fait que la fonction DAS (Database Activity Streams) est un audit global. Une activité SQL élevée génère des événements de flux d'activité qui doivent être enregistrés.

Actions

Nous vous recommandons différentes actions en fonction de l'origine de votre événement d'attente :

- Réduisez le nombre d'instructions SQL ou désactivez les flux d'activité de la base de données. Cela permet de réduire le nombre d'événements nécessitant des écritures durables.
- Passez en mode asynchrone. Cela permet de réduire les conflits sur l'événement d'attente `IPC:DamRecordTxAck`.

  Cependant, la fonction DAS ne peut garantir la durabilité de chaque événement en mode asynchrone.

IPC:parallel wait events

Les événements `IPC:parallel wait events` suivants indiquent qu'une session est en attente d'une communication interprocessus liée aux opérations d'exécution de requêtes parallèles.

- `IPC:BgWorkerStartup` : un processus attend qu'un processus de travail parallèle termine sa séquence de démarrage. Cela se produit lors de l'initialisation des applications de travail pour l'exécution de requêtes parallèles.
- `IPC:BgWorkerShutdown` : un processus attend la fin de la séquence d'arrêt d'un processus de travail parallèle. Cela se produit pendant la phase de nettoyage de l'exécution de requêtes parallèles.
- `IPC:ExecuteGather` : un processus attend de recevoir des données de processus de travail en parallèle pendant l'exécution de la requête. Cela se produit lorsque le processus principal doit recueillir des résultats auprès de ses applications de travail.
- `IPC:ParallelFinish` : un processus attend que les applications de travail en parallèle terminent leur exécution et publient leurs résultats finaux. Cela se produit pendant la phase de finalisation de l'exécution des requêtes parallèles.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d’attente s’appliquent à toutes les versions d’Aurora PostgreSQL.

Contexte

L’exécution de requêtes parallèles dans PostgreSQL implique la collaboration de plusieurs processus pour le traitement d’une seule requête. Lorsqu’une requête est jugée appropriée pour la parallélisation, un processus principal est coordonné avec un ou plusieurs processus de travail en parallèle en fonction du réglage du paramètre `max_parallel_workers_per_gather`. Le processus principal répartit le travail entre les applications de travail, chaque application de travail traite sa partie des données de manière indépendante et les résultats sont renvoyés au processus principal.

> **Note**
>
> Chaque application de travail en parallèle fonctionne comme un processus distinct dont les besoins en ressources sont similaires à ceux d’une session utilisateur complète. Une requête parallèle avec quatre applications de travail peut ainsi consommer jusqu’à cinq fois plus de ressources (processeur, mémoire, bande passante d’E/S) par rapport à une requête non exécutée en parallèle, car le processus principal et chaque processus de travail conservent leurs propres allocations de ressources. Par exemple, des paramètres tels que `work_mem` sont appliqués individuellement à chaque application de travail, multipliant potentiellement l’utilisation totale de la mémoire entre tous les processus.

L’architecture des requêtes parallèles comprend trois composants principaux :

- Processus principal : processus clé qui initie l’opération en parallèle, répartit la charge de travail et assure la coordination avec les processus de travail.

- Processus de travail : processus d'arrière-plan qui exécutent des parties de la requête parallèle.
- Fusion Gather/Gather : opérations qui combinent les résultats de plusieurs processus de travail pour les renvoyer au processus principal.

Lors d'une exécution en parallèle, les processus doivent communiquer entre eux par le biais de mécanismes de communication interprocessus (IPC). Ces événements d'attente IPC se produisent au cours de différentes phases :

- Démarrage des applications de travail : lorsque des applications de travail en parallèle sont initialisées
- Échange de données : lorsque les applications de travail traitent les données et envoient les résultats au processus principal
- Arrêt des applications de travail : lorsque l'exécution en parallèle est terminée et que les applications de travail sont arrêtées
- Points de synchronisation : lorsque les processus doivent se coordonner ou attendre que d'autres processus terminent leurs tâches

Il est essentiel de comprendre ces événements d'attente pour pouvoir résoudre les problèmes de performances liés à l'exécution de requêtes parallèles, en particulier dans les environnements à forte concurrence dans lesquels plusieurs requêtes parallèles peuvent s'exécuter simultanément.

Causes probables de l'augmentation du nombre d'événements d'attente

Plusieurs facteurs peuvent contribuer à augmenter le nombre d'événements d'attente IPC liés à l'exécution de requêtes parallèles :

Forte concurrence des requêtes parallèles

Lorsque de nombreuses requêtes parallèles sont exécutées simultanément, cela peut entraîner une contention des ressources et une augmentation des temps d'attente pour les opérations IPC. Cela est particulièrement courant dans les systèmes impliquant des volumes de transactions ou des charges de travail analytiques élevés.

Plans de requêtes parallèles sous-optimaux

Si le planificateur de requêtes choisit des plans de requêtes parallèles inefficaces, cela peut entraîner une parallélisation inutile ou une mauvaise répartition du travail entre les applications de travail. Cela peut entraîner une augmentation des temps d'attente IPC, en particulier pour

les événements IPC:ExecuteGather et IPC:ParallelFinish. Ces problèmes de planification sont souvent dus à des statistiques obsolètes et au gonflement des tables et des index.

Démarrage et arrêt fréquents des applications de travail en parallèle

Les requêtes de courte durée qui déclenchent et interrompent fréquemment des applications de travail en parallèle peuvent entraîner une augmentation du nombre d'événements `IPC:BgWorkerStartup` et `IPC:BgWorkerShutdown`. Cela se produit souvent dans les charges de travail OLTP, qui impliquent beaucoup de petites requêtes pouvant être exécutées en parallèle.

Contraintes liées aux ressources

La capacité limitée du processeur, de la mémoire ou des E/S peut entraîner des goulets d'étranglement lors de l'exécution en parallèle, ce qui augmente les temps d'attente pour tous les événements IPC. Par exemple, si le processeur est saturé, les processus de travail peuvent mettre plus de temps à démarrer ou à traiter leur part du travail.

Structures de requête complexes

Les requêtes comportant plusieurs niveaux de parallélisme (jointures en parallèle suivies d'agrégations parallèles, par exemple) peuvent entraîner des modèles IPC plus complexes et des temps d'attente potentiellement accrus, en particulier pour les événements `IPC:ExecuteGather`.

Ensembles de résultats volumineux

Les requêtes qui génèrent des ensembles de résultats volumineux peuvent entraîner une augmentation des temps d'attente `IPC:ExecuteGather`, car le processus principal passe plus de temps à collecter et à traiter les résultats des processus de travail.

La compréhension de ces facteurs peut aider à diagnostiquer et à résoudre les problèmes de performances liés à l'exécution de requêtes parallèles dans Aurora PostgreSQL.

Actions

Lorsque vous voyez des temps d'attente liés à une requête parallèle, cela signifie généralement qu'un processus dorsal coordonne ou attend des processus de travail parallèles. Ces temps d'attente sont courants lors de l'exécution de plans en parallèle. Vous pouvez étudier et atténuer l'impact de ces temps d'attente en surveillant l'utilisation des applications de travail en parallèle, en vérifiant les paramètres et en ajustant l'exécution des requêtes et l'allocation des ressources.

Rubriques

Analyse des plans de requêtes pour détecter un parallélisme inefficace

L'exécution de requêtes parallèles peut souvent entraîner une instabilité du système, des pics de processeur et des variations imprévisibles des performances des requêtes. Il est essentiel d'analyser de manière approfondie si le parallélisme améliore réellement votre charge de travail spécifique. Utilisez EXPLAIN ANALYZE pour passer en revue les plans d'exécution de requêtes parallèles.

Désactivez temporairement le parallélisme au niveau de la session pour comparer l'efficacité du plan :

```
SET max_parallel_workers_per_gather = 0;
EXPLAIN ANALYZE <your_query>;
```

Réactivez le parallélisme et comparez :

```
RESET max_parallel_workers_per_gather;
EXPLAIN ANALYZE <your_query>;
```

Si la désactivation du parallélisme donne des résultats meilleurs ou plus cohérents, envisagez de le désactiver pour des requêtes spécifiques au niveau de la session à l'aide des commandes SET. Pour un impact plus large, vous pouvez désactiver le parallélisme au niveau de l'instance en ajustant les paramètres pertinents dans le groupe de paramètres de votre cluster ou de votre instance. Pour plus d'informations, consultez Paramètres Amazon Aurora PostgreSQL..

Surveillance de l’utilisation des requêtes parallèles

Utilisez les requêtes suivantes pour avoir une meilleure visibilité sur l’activité et la capacité des requêtes parallèles :

Vérifiez les processus de travail en parallèle actifs :

```
SELECT
    COUNT(*)
FROM
    pg_stat_activity
WHERE
    backend_type = 'parallel worker';
```

Cette requête indique le nombre de processus de travail en parallèle actifs. Une valeur élevée peut indiquer que le paramètre max_parallel_workers est configuré avec une valeur élevée, ce que vous pouvez choisir de réduire.

Vérifiez les requêtes parallèles simultanées :

```
SELECT
    COUNT(DISTINCT leader_pid)
FROM
    pg_stat_activity
WHERE
    leader_pid IS NOT NULL;
```

Cette requête renvoie le nombre de processus principaux distincts qui ont lancé des requêtes parallèles. Un nombre élevé indique que plusieurs sessions exécutent des requêtes parallèles simultanément, ce qui peut augmenter la demande en termes de processeur et de mémoire.

Vérification et ajustement des paramètres des requêtes parallèles

Passez en revue les paramètres suivants pour vous assurer qu’ils correspondent à votre charge de travail :

- `max_parallel_workers` : nombre total d’applications de travail en parallèle entre toutes les sessions.

- `max_parallel_workers_per_gather` : nombre maximum d'applications de travail par requête.

Pour les charges de travail OLAP, l'augmentation de ces valeurs peut améliorer les performances. Pour les charges de travail OLTP, des valeurs plus faibles sont généralement préférées.

```
SHOW max_parallel_workers;
SHOW max_parallel_workers_per_gather;
```

Optimisation de l'allocation des ressources

Surveillez l'utilisation du processeur et envisagez d'ajuster le nombre de processeurs virtuels s'il est constamment élevé et si votre application bénéficie des requêtes parallèles. Assurez-vous que la mémoire disponible est suffisante pour les opérations en parallèle.

- Utilisez les métriques Performance Insights pour déterminer si le système est lié au processeur.
- Chaque application de travail en parallèle utilise sa propre `work_mem`. Assurez-vous que l'utilisation totale de la mémoire respecte les limites de l'instance.

Les requêtes parallèles peuvent consommer beaucoup plus de ressources que les requêtes qui ne sont pas exécutées en parallèle, car chaque processus de travail est un processus complètement distinct qui a à peu près le même impact sur le système qu'une session utilisateur supplémentaire. Cela doit être pris en compte lors du choix d'une valeur pour ce paramètre, ainsi que lors de la configuration des autres paramètres contrôlant l'utilisation des ressources, tels que `work_mem`. Pour plus d'informations, consultez la documentation sur PostgreSQL. Les limites de ressources telles que `work_mem` sont appliquées individuellement à chaque application de travail, ce qui signifie que l'utilisation totale peut être beaucoup plus élevée entre tous les processus qu'elle ne le serait normalement pour un seul processus.

Envisagez d'augmenter le nombre de processeurs virtuels ou d'ajuster les paramètres de mémoire si votre charge de travail est fortement parallélisée.

Vérification de la gestion des connexions

En cas de saturation des connexions, passez en revue les stratégies de regroupement des connexions des applications. Envisagez d'implémenter le regroupement de connexions au niveau de l'application s'il n'est pas déjà utilisé.

Vérification et optimisation des opérations de maintenance

Coordonnez la création d'index et les autres tâches de maintenance pour éviter les conflits de ressources. Pensez à planifier ces opérations pendant les heures creuses. Évitez de planifier de lourdes opérations de maintenance (par exemple, des constructions d'index en parallèle) pendant les périodes de forte charge des requêtes utilisateur. Ces opérations peuvent consommer des applications de travail en parallèle et avoir un impact sur les performances des requêtes régulières.

Utilisation de la gestion de plans de requêtes (QPM)

Dans Aurora PostgreSQL, la fonctionnalité de gestion des plans de requêtes (QPM) est conçue pour garantir l'adaptabilité et la stabilité des plans, quelles que soient les modifications apportées à l'environnement de base de données qui peuvent entraîner une régression des plans de requêtes. Pour plus d'informations, consultez Présentation de la gestion des plans de requêtes d'Aurora PostgreSQL. QPM apporte un certain contrôle sur l'optimiseur. Passez en revue les plans approuvés dans QPM pour vous assurer qu'ils correspondent aux paramètres de parallélisme actuels. Mettez à jour ou supprimez les plans obsolètes susceptibles de provoquer une exécution en parallèle sous-optimale.

Vous pouvez également corriger les plans à l'aide de pg_hint_plan. Pour plus d'informations, consultez Correction de plans à l'aide de pg_hint_plan. Vous pouvez utiliser l'indicateur `Parallel` pour imposer l'exécution en parallèle. Pour plus d'informations, consultez Conseils pour les plans en parallèle.

IPC : ProcArrayGroupUpdate

L'`IPC:ProcArrayGroupUpdate`événement se produit lorsqu'une session attend que le chef de groupe mette à jour le statut de la transaction à la fin de cette opération. Alors que PostgreSQL associe généralement les événements d'attente de type IPC aux opérations de requêtes parallèles, cet événement d'attente particulier n'est pas spécifique aux requêtes parallèles.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations sur les événements d’attente sont prises en charge pour toutes les versions de Aurora PostgreSQL.

Contexte

Comprendre le tableau de processus — Le tableau de processus (proc) est une structure de mémoire partagée dans PostgreSQL. Il contient des informations sur tous les processus en cours, y compris les détails des transactions. Au cours de la fin de la transaction (`COMMITouROLLBACK`), il ProcArray doit être mis à jour pour refléter le changement et effacer le TransactionID du tableau. La session qui tente de terminer sa transaction doit acquérir un verrou exclusif sur le ProcArray. Cela empêche d'autres processus d'obtenir des verrous partagés ou exclusifs sur celui-ci.

Mécanisme de mise à jour de groupe — Lors de l'exécution d'un COMMIT ou d'un ROLLBACK, si un processus principal ne parvient pas à obtenir un ProcArrayLock code en mode exclusif, il met à jour un champ spécial appelé. ProcArrayGroupMember Cela ajoute la transaction à la liste des sessions qui ont l'intention de se terminer. Ce processus principal est ensuite mis en veille et le temps qu'il passe en veille est instrumenté comme événement d' ProcArrayGroupUpdate attente. Le premier processus du ProcArray with procArrayGroup Member, appelé processus leader, acquiert ProcArrayLock le mode exclusif. Il efface ensuite la liste des processus en attente d'effacement de l'identifiant de transaction du groupe. Une fois cette opération terminée, le leader libère ProcArrayLock puis réveille tous les processus de cette liste, les informant que leur transaction est terminée.

Causes probables de l’augmentation du nombre d’événements d’attente

Plus le nombre de processus en cours d'exécution est élevé, plus un leader conservera longtemps un procArrayLock mode exclusif. Par conséquent, plus le nombre de transactions d'écriture aboutit à un scénario de mise à jour de groupe, ce qui entraîne une accumulation potentielle de processus en attente de l'événement d'`ProcArrayGroupUpdate`attente. Dans la vue SQL principale de Database Insights, vous verrez que COMMIT est l'instruction présentant la majeure partie de cet événement d'attente. Cela est normal, mais cela nécessitera une étude plus approfondie du code SQL d'écriture spécifique en cours d'exécution afin de déterminer les mesures appropriées à prendre.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d’attente. Identifiez les `IPC:ProcArrayGroupUpdate` événements à l'aide d'Amazon RDS Performance Insights ou en interrogeant la vue système de PostgreSQL. `pg_stat_activity`

Rubriques

Surveillance des opérations de validation et d'annulation des transactions

Surveillez les validations et les annulations — L'augmentation du nombre de validations et de annulations peut entraîner une pression accrue sur le. ProcArray Par exemple, si une instruction SQL commence à échouer en raison d'une augmentation du nombre de violations de clés dupliquées, vous pouvez assister à une augmentation des annulations, ce qui peut accroître les ProcArray contentions et le surpeuplement des tables.

Amazon RDS Database Insights fournit les `xact_commit` métriques PostgreSQL et indique le nombre de validations `xact_rollback` et de rollbacks par seconde.

Réduire la simultanéité

Transactions par lots — Dans la mesure du possible, opérations groupées en transactions uniques afin de réduire les commit/rollback opérations.

Limiter la simultanéité — Réduisez le nombre de transactions actives simultanément afin d'atténuer les conflits liés au verrouillage sur le. ProcArray Bien que cela nécessite quelques tests, la réduction du nombre total de connexions simultanées peut réduire les conflits et maintenir le débit.

Mise en œuvre du regroupement de connexions

Solutions de regroupement de connexions : utilisez le regroupement de connexions pour gérer efficacement les connexions aux bases de données, en réduisant le nombre total de backends et donc la charge de travail sur le ProcArray. Bien que cela nécessite quelques tests, la réduction du nombre total de connexions simultanées peut réduire les conflits et maintenir le débit.

Pour plus d'informations, consultez la section Regroupement de connexions pour Aurora PostgreSQL.

Réduisez les tempêtes de connexion — De même, une tendance à créer et à terminer fréquemment des connexions entraîne une pression supplémentaire sur le ProcArray. En réduisant ce schéma, la contention globale est réduite.

Lock:advisory

L'événement `Lock:advisory` se produit lorsqu'une application PostgreSQL utilise un verrou pour coordonner l'activité sur plusieurs sessions.

Rubriques

Versions de moteur pertinentes

Ces informations sur les événements d'attente s'appliquent à Aurora PostgreSQL 9.6 et versions ultérieures.

Contexte

Les verrous consultatifs PostgreSQL sont des verrous coopératifs de niveau application explicitement verrouillés et déverrouillés par le code d'application de l'utilisateur. Une application peut utiliser des verrous consultatifs PostgreSQL pour coordonner l'activité sur plusieurs sessions. Contrairement aux verrous standard, de niveau objet ou ligne, l'application dispose d'un contrôle total sur la durée de vie du verrou. Pour en savoir plus, consultez Advisory Locks dans la documentation PostgreSQL.

Les verrous consultatifs peuvent être libérés avant la fin d'une transaction ou être maintenus par une session sur plusieurs transactions. Cela ne s'applique pas aux verrous implicites appliqués par le système, comme un verrou exclusif d'accès à une table acquis par une instruction `CREATE INDEX`.

Pour accéder à la description des fonctions utilisées pour acquérir (verrouiller) et libérer (déverrouiller) les verrous consultatifs, consultez Advisory Lock Functions dans la documentation PostgreSQL.

Les verrous consultatifs sont implémentés au-dessus du système de verrouillage PostgreSQL standard et sont visibles dans la vue système `pg_locks`.

Causes

Ce type de verrou est exclusivement contrôlé par une application qui l'utilise explicitement. Les verrous consultatifs qui sont acquis pour chaque ligne dans le cadre d'une requête peuvent provoquer un pic de verrous ou une accumulation à long terme.

Ces effets se produisent lorsque la requête est exécutée d'une manière qui acquiert des verrous sur plus de lignes que celles renvoyées par la requête. L'application doit finir par libérer chaque verrou, mais si des verrous sont acquis sur des lignes qui ne sont pas renvoyées, l'application ne peut pas tous les trouver.

L'exemple suivant est extrait de la section Advisory Locks de la documentation PostgreSQL.

```
SELECT pg_advisory_lock(id) FROM foo WHERE id > 12345 LIMIT 100;
```

Dans cet exemple, la clause `LIMIT` ne peut arrêter la sortie de la requête que lorsque les lignes ont déjà été sélectionnées en interne et que leurs valeurs d'ID ont été verrouillées. Cela peut se produire soudainement lorsqu'un volume de données croissant amène le planificateur à choisir un plan d'exécution différent qui n'a pas été testé lors de la phase de développement. Dans ce cas, l'accumulation se produit parce que l'application appelle explicitement `pg_advisory_unlock` pour chaque valeur d'ID verrouillée. Mais elle ne trouve pas l'ensemble de verrous acquis sur les lignes qui n'ont pas été renvoyées. Comme les verrous sont acquis au niveau de la session, ils ne sont pas libérés automatiquement à la fin de la transaction.

Les pics de tentatives de verrouillage bloquées peuvent également être liés à des conflits involontaires. Lors de ces conflits, des parties non liées de l'application partagent par erreur le même espace d'ID de verrou.

Actions

Examinez la façon dont les verrous consultatifs sont utilisés par l'application et indiquez où et quand, dans le flux d'application, chaque type de verrou consultatif est acquis et libéré.

Déterminez si une session acquiert trop de verrous ou si une session longue ne libère pas les verrous suffisamment tôt, ce qui entraîne une accumulation lente des verrous. Vous pouvez corriger une accumulation lente de verrous au niveau de la session en mettant fin à la session à l'aide de `pg_terminate_backend(pid)`.

Un client en attente d'un verrou consultatif apparaît dans `pg_stat_activity` avec `wait_event_type=Lock` et `wait_event=advisory`. Vous pouvez obtenir des valeurs de

verrouillage spécifiques en interrogeant la vue système `pg_locks` sur le même `pid`, à la recherche de `locktype=advisory` et `granted=f`.

Vous pouvez ensuite identifier la session de blocage en interrogeant `pg_locks` sur le même verrou consultatif doté de `granted=t`, comme illustré dans l'exemple suivant.

```
SELECT blocked_locks.pid AS blocked_pid,
         blocking_locks.pid AS blocking_pid,
         blocked_activity.usename AS blocked_user,
         blocking_activity.usename AS blocking_user,
         now() - blocked_activity.xact_start AS blocked_transaction_duration,
         now() - blocking_activity.xact_start AS blocking_transaction_duration,
         concat(blocked_activity.wait_event_type,':',blocked_activity.wait_event) AS
 blocked_wait_event,
         concat(blocking_activity.wait_event_type,':',blocking_activity.wait_event) AS
 blocking_wait_event,
         blocked_activity.state AS blocked_state,
         blocking_activity.state AS blocking_state,
         blocked_locks.locktype AS blocked_locktype,
         blocking_locks.locktype AS blocking_locktype,
         blocked_activity.query AS blocked_statement,
         blocking_activity.query AS blocking_statement
    FROM pg_catalog.pg_locks blocked_locks
    JOIN pg_catalog.pg_stat_activity blocked_activity ON blocked_activity.pid =
 blocked_locks.pid
    JOIN pg_catalog.pg_locks blocking_locks
        ON blocking_locks.locktype = blocked_locks.locktype
        AND blocking_locks.DATABASE IS NOT DISTINCT FROM blocked_locks.DATABASE
        AND blocking_locks.relation IS NOT DISTINCT FROM blocked_locks.relation
        AND blocking_locks.page IS NOT DISTINCT FROM blocked_locks.page
        AND blocking_locks.tuple IS NOT DISTINCT FROM blocked_locks.tuple
        AND blocking_locks.virtualxid IS NOT DISTINCT FROM blocked_locks.virtualxid
        AND blocking_locks.transactionid IS NOT DISTINCT FROM
 blocked_locks.transactionid
        AND blocking_locks.classid IS NOT DISTINCT FROM blocked_locks.classid
        AND blocking_locks.objid IS NOT DISTINCT FROM blocked_locks.objid
        AND blocking_locks.objsubid IS NOT DISTINCT FROM blocked_locks.objsubid
        AND blocking_locks.pid != blocked_locks.pid
    JOIN pg_catalog.pg_stat_activity blocking_activity ON blocking_activity.pid =
 blocking_locks.pid
    WHERE NOT blocked_locks.GRANTED;
```

Toutes les fonctions d'API des verrous consultatifs comportent deux ensembles d'arguments, soit un argument `bigint`, soit deux arguments `integer` :

- Pour les fonctions d'API comportant un argument `bigint`, les 32 bits supérieurs figurent dans `pg_locks.classid` et les 32 bits inférieurs se trouvent dans `pg_locks.objid`.
- Pour les fonctions d'API comportant deux arguments `integer`, le premier argument est `pg_locks.classid` et le deuxième est `pg_locks.objid`.

La valeur `pg_locks.objsubid` indique quelle forme d'API a été utilisée : 1 pour un argument `bigint` et 2 pour deux arguments `integer`.

Lock:extend

L'événement `Lock:extend` se produit lorsqu'un processus backend attend de verrouiller une relation pour l'étendre alors qu'un autre processus présente un verrou sur cette relation dans le même but.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL.

Contexte

L'événement `Lock:extend` indique qu'un processus backend attend d'étendre une relation sur laquelle un autre processus backend détient un verrou pendant qu'il étend cette relation. Comme un seul processus à la fois peut étendre une relation, le système génère un événement d'attente `Lock:extend`. Les opérations `INSERT`, `COPY` et `UPDATE` peuvent générer cet événement.

Causes probables de l'augmentation du nombre d'événements d'attente

Un événement `Lock:extend` trop fréquent peut révéler un problème de performances dont les causes sont généralement les suivantes :

Augmentation du nombre d'insertions ou de mises à jour simultanées dans la même table

Le nombre de sessions simultanées associées à des requêtes d'insertion ou de mise à jour peut augmenter.

Bande passante réseau insuffisante

La bande passante réseau de l'instance de base de données peut être insuffisante pour répondre aux besoins de communication de stockage de la charge de travail actuelle. Cela peut contribuer à une latence de stockage qui entraîne une augmentation des événements `Lock:extend`.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Réduisez les insertions et les mises à jour simultanées dans la même relation
- Augmentez la bande passante réseau

Réduisez les insertions et les mises à jour simultanées dans la même relation

Tout d'abord, déterminez s'il y a une augmentation des métriques `tup_inserted` et `tup_updated`, et une augmentation concomitante de cet événement d'attente. Si tel est le cas, déterminez quelles relations sont en conflit pour les opérations d'insertion et de mise à jour. Pour ce faire, interrogez la vue `pg_stat_all_tables` afin de connaître les valeurs des champs `n_tup_ins` et `n_tup_upd`. Pour en savoir plus sur la vue `pg_stat_all_tables`, consultez pg_stat_all_tables dans la documentation PostgreSQL.

Pour en savoir plus sur les requêtes de blocage et les requêtes bloquées, interrogez `pg_stat_activity` comme dans l'exemple suivant :

```
SELECT
    blocked.pid,
```

```
    blocked.usename,
    blocked.query,
    blocking.pid AS blocking_id,
    blocking.query AS blocking_query,
    blocking.wait_event AS blocking_wait_event,
    blocking.wait_event_type AS blocking_wait_event_type
FROM pg_stat_activity AS blocked
JOIN pg_stat_activity AS blocking ON blocking.pid = ANY(pg_blocking_pids(blocked.pid))
where
blocked.wait_event = 'extend'
and blocked.wait_event_type = 'Lock';

   pid  | usename  |            query             | blocking_id |
            blocking_query                          | blocking_wait_event |
 blocking_wait_event_type
  ------+----------+------------------------------+-------------
+-------------------------------------------------------------------
+--------------------+-------------------------
   7143 |  myuser  | insert into tab1 values (1); |        4600 | INSERT INTO tab1 (a)
 SELECT s FROM generate_series(1,1000000) s; | DataFileExtend      | IO
```

Après avoir identifié les relations qui contribuent à l'augmentation des événements `Lock:extend`, utilisez les techniques suivantes pour réduire les conflits :

- Déterminez si vous pouvez utiliser le partitionnement pour réduire les conflits relatifs à la même table. La séparation des tuples insérés ou mis à jour dans différentes partitions peut réduire les conflits. Pour plus d'informations sur le partitionnement, voir Gestion des partitions PostgreSQL avec l'extension pg_partman.
- Si l'événement d'attente est principalement dû à une activité de mise à jour, vous pouvez réduire la valeur du facteur de remplissage de la relation. Cela peut réduire les requêtes de nouveaux blocs lors de la mise à jour. Le facteur de remplissage est un paramètre de stockage relatif à une table qui détermine la quantité maximale d'espace requise pour remplir une page de la table. Il est exprimé en pourcentage de l'espace total d'une page. Pour en savoir plus sur le facteur de remplissage, consultez CREATE TABLE dans la documentation PostgreSQL.

> **Important**
>
> Si vous modifiez le facteur de remplissage, nous vous recommandons vivement de tester votre système, car en fonction de votre charge de travail, la modification de cette valeur peut nuire aux performances.

Augmentez la bande passante réseau

Pour déterminer si la latence d'écriture a augmenté, consultez la métrique `WriteLatency` dans CloudWatch. Si tel est le cas, utilisez les métriques Amazon CloudWatch `WriteThroughput` et `ReadThroughput` pour surveiller le trafic lié au stockage sur le cluster de bases de données. Ces métriques peuvent vous aider à déterminer si la bande passante réseau est suffisante pour l'activité de stockage de votre charge de travail.

Si la bande passante réseau est insuffisante, augmentez-la. Si votre instance de base de données atteint les limites de sa bande passante réseau, le seul moyen d'augmenter la bande passante est d'augmenter la taille de l'instance de base de données.

Pour plus d'informations sur les métriques CloudWatch, consultez CloudWatch Métriques Amazon pour Amazon Aurora. Pour en savoir plus sur les performances réseau de chaque classe d'instance de base de données, consultez Spécifications matérielles pour les classes d'instance de base de données pour Aurora.

Lock:Relation

L'événement `Lock:Relation` se produit lorsqu'une requête attend d'acquérir un verrou sur une table ou une vue (relation) actuellement verrouillée par une autre transaction.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL.

Contexte

La plupart des commandes PostgreSQL utilisent implicitement des verrous pour contrôler les accès simultanés aux données des tables. Vous pouvez également utiliser ces verrous explicitement dans

le code de votre application grâce à la commande `LOCK`. De nombreux modes de verrouillage ne sont pas compatibles entre eux et peuvent bloquer des transactions lorsqu'ils tentent d'accéder au même objet. Dans ce cas, Aurora PostgreSQL génère un événement `Lock:Relation`. Voici quelques exemples courants :

- Des verrous exclusifs tels que `ACCESS EXCLUSIVE` peuvent bloquer tous les accès simultanés. Les opérations en langage de définition de données (DDL) telles que `DROP TABLE`, `TRUNCATE`, `VACUUM FULL` et `CLUSTER` acquièrent implicitement des verrous `ACCESS EXCLUSIVE`. `ACCESS EXCLUSIVE` est également le mode de verrouillage par défaut pour les instructions `LOCK TABLE` qui ne spécifient pas explicitement de mode.
- L'utilisation de `CREATE INDEX (without CONCURRENT)` sur une table entre en conflit avec les instructions du langage de manipulation de données (DML) `UPDATE`, `DELETE` et `INSERT`, qui acquièrent des verrous `ROW EXCLUSIVE`.

Pour en savoir plus sur les verrous de niveau table et les modes de verrouillage conflictuels, consultez Explicit Locking dans la documentation PostgreSQL.

Le déblocage lié aux requêtes et transactions de blocage s'effectue généralement de l'une des manières suivantes :

- Requête de blocage – L'application peut annuler la requête ou l'utilisateur peut mettre fin au processus. Le moteur peut également forcer la requête à se terminer en raison de l'expiration du délai d'attente d'une instruction de la session ou d'un mécanisme de détection de blocage.
- Transaction de blocage – Une transaction met fin à son blocage lorsqu'elle exécute une instruction `ROLLBACK` ou `COMMIT`. Les restaurations sont également automatiques lorsque les sessions sont déconnectées par un client ou suite à des problèmes de réseau, ou lorsqu'elles sont interrompues. Les sessions peuvent être interrompues lorsque le moteur de base de données est arrêté, lorsque le système est à court de mémoire, etc.

Causes probables de l'augmentation du nombre d'événements d'attente

Lorsque l'événement `Lock:Relation` se produit plus fréquemment que la normale, il peut indiquer un problème de performances. Les causes sont généralement les suivantes :

Augmentation du nombre de sessions simultanées avec des verrous de table conflictuels

> Le nombre de sessions simultanées peut augmenter lorsque des requêtes verrouillent la même table avec des modes de verrouillage conflictuels.

Opérations de maintenance

Les opérations de maintenance liées à l'état comme `VACUUM` et `ANALYZE` peuvent considérablement augmenter le nombre de verrous en conflit. `VACUUM FULL` acquiert un verrou `ACCESS EXCLUSIVE`, et `ANALYZE` acquiert un verrou `SHARE UPDATE EXCLUSIVE`. Ces deux types de verrous peuvent provoquer un événement d'attente `Lock:Relation`. Les opérations de maintenance des données d'application telles que l'actualisation d'une vue matérialisée peuvent également augmenter le nombre de requêtes et de transactions bloquées.

Verrous sur les instances de lecture

Il peut y avoir un conflit entre les verrous relationnels des volumes en écriture et des volumes en lecture. Actuellement, seuls les verrous relationnels `ACCESS EXCLUSIVE` sont répliqués vers les instances de lecture. Cependant, le verrou relationnel `ACCESS EXCLUSIVE` entrera en conflit avec tout verrou relationnel `ACCESS SHARE` détenu par le volume en lecture. Cela peut entraîner une augmentation des événements d'attente de relation de verrouillage sur le volume en lecture.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Réduisez l'impact des instructions SQL de blocage
- Minimisez l'effet des opérations de maintenance
- Recherchez les verrous de lecture

Réduisez l'impact des instructions SQL de blocage

Pour réduire l'impact des instructions SQL de blocage, si possible, modifiez le code de votre application. Voici deux techniques courantes pour réduire les blocages :

- Utilisez l'option `NOWAIT` – Certaines commandes SQL, telles que les instructions `SELECT` et `LOCK` prennent en charge cette option. La directive `NOWAIT` annule la requête liée à la demande de verrou si le verrou ne peut pas être acquis immédiatement. Cette technique permet d'éviter qu'une session de blocage ne provoque un empilement de sessions bloquées derrière elle.

  Par exemple, supposons que la transaction A attende un verrou détenu par la transaction B. Si B demande un verrou sur une table qui est verrouillée par la transaction C, la transaction A peut être bloquée jusqu'à ce que la transaction C se termine. Mais si la transaction B utilise

`NOWAIT` lorsqu'elle demande le verrou sur C, elle peut échouer rapidement et faire en sorte que la transaction A n'ait pas à attendre indéfiniment.

- Utilisez `SET lock_timeout` – Définissez une valeur `lock_timeout` afin de limiter le délai d'attente d'une instruction SQL pour acquérir un verrou sur une relation. Si le verrou n'est pas acquis dans le délai spécifié, la transaction qui l'a demandé est annulée. Définissez cette valeur au niveau de la session.

Minimisez l'effet des opérations de maintenance

Les opérations de maintenance telles que `VACUUM` et `ANALYZE` sont importantes. Nous vous recommandons de ne pas les désactiver parce que vous trouvez des événements d'attente `Lock:Relation` liés à ces opérations de maintenance. Les approches suivantes peuvent minimiser l'effet de ces opérations :

- Exécutez les opérations de maintenance manuellement pendant les heures creuses.
- Pour réduire les attentes `Lock:Relation` causées par les tâches autovacuum, procédez aux réglages nécessaires de la fonction autovacuum. Pour en savoir plus sur le réglage de la fonction autovacuum, consultez Utilisation de la fonction autovacuum de PostgreSQL sur Amazon RDS dans le Guide de l'utilisateur Amazon RDS.

Recherchez les verrous de lecture

Vous pouvez déterminer comment des sessions ouvertes simultanément sur un enregistreur et un lecteur peuvent détenir des verrous qui se bloquent mutuellement. Une façon de le faire est d'exécuter des requêtes qui renvoient le type de verrou et la relation. Dans le tableau, vous trouverez une séquence de requêtes à deux sessions simultanées de ce type, une session d'enregistreur et une session de lecteur.

Le processus de relecture attend la durée de `max_standby_streaming_delay` avant d'annuler la requête du lecteur. Comme le montre l'exemple, le délai de verrouillage de 100 ms est bien inférieur au délai `max_standby_streaming_delay` par défaut de 30 secondes. Le verrouillage s'arrête avant que cela ne devienne un problème.

Événement de la séquence	Session	Commande ou sortie
Définit une variable d'environnement appelée READER	Session de lecteur	Commande de la CLI :

Événement de la séquence	Session	Commande ou sortie
avec la valeur spécifiée et essaie de se connecter à l'instance de base de données avec ce point de terminaison.		`export READER=au rorapg2.1234567891 0.us-west-1.rds.am azonaws.com` `psql -h $READER` Sortie : `psql (15devel, server 10.14)` `Type "help" for help.`
Définit une variable d'environ nement appelée WRITER et essaie de se connecter à l'instance de base de données avec ce point de terminaison.	Session d'enregistreur	Commande de la CLI : `export WRITER=au rorapg1.1234567891 0.us-west-1.rds.am azonaws.com` `psql -h $WRITER` Sortie : `psql (15devel, server 10.14)` `Type "help" for help.`
La session d'enregistreur crée une table t1 sur l'instance d'enregistreur.	Session d'enregistreur	Requête PostgreSQL : `postgres=> CREATE TABLE t1(b integer);` `CREATE TABLE`

Événement de la séquence	Session	Commande ou sortie
Si l’enregistreur n’a pas de requêtes conflictuelles, le verrou ACCESS EXCLUSIVE est acquis immédiatement au niveau de l’enregistreur.	Session d’enregistreur	Verrou `ACCESS EXCLUSIVE` activé
La session du lecture définit un intervalle d’expiration de 100 millisecondes pour le verrou.	Session de lecteur	Requête PostgreSQL : `postgres=> SET lock_timeout=100;` `SET`
La session de lecteur tente de lire les données de la table t1 sur l’instance de lecteur.	Session de lecteur	Requête PostgreSQL : `postgres=> SELECT * FROM t1;` Exemple de sortie : `b` `---` `(0 rows)`
La session d’enregistreur abandonne t1.	Session d’enregistreur	Requête PostgreSQL : `postgres=> BEGIN;` `BEGIN` `postgres=> DROP TABLE t1;` `DROP TABLE` `postgres=>`

Événement de la séquence	Session	Commande ou sortie
La requête expire et est annulée sur le lecteur.	Session de lecteur	Requête PostgreSQL : `postgres=>  SELECT * FROM t1;` Exemple de sortie : `ERROR:  canceling statement due to lock timeout` `LINE 1: SELECT * FROM t1;` `^`
Pour déterminer la cause de l'erreur, la session de lecteur interroge `pg_locks` et `pg_stat_activity` .	Session de lecteur	Requête PostgreSQL : `postgres=>  SELECT locktype, relation, mode, backend_type` `postgres=>  FROM pg_locks l, pg_stat_activity t1` `postgres=>  WHERE l.pid=t1.pid AND relation = 't1'::regclass::oid;`

Événement de la séquence	Session	Commande ou sortie
Le résultat indique que le processus `aurora wal replay` applique un verrou `ACCESS EXCLUSIVE` sur la table t1.	Session de lecteur	Résultat de la requête : `locktype | relation | mode | backend_type` `----------+----------+------------------+---------------` `relation | 68628525 | AccessExclusiveLock | aurora wal replay` `(1 row)`

Lock:transactionid

L'événement `Lock:transactionid` se produit lorsqu'une transaction attend un verrou au niveau ligne.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de l'augmentation du nombre d'événements d'attente
- Actions

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL.

Contexte

L'événement `Lock:transactionid` se produit lorsqu'une transaction tente d'acquérir un verrou de niveau ligne qui a déjà été accordé à une transaction exécutée en même temps. La session qui présente l'événement d'attente `Lock:transactionid` est bloquée à cause de ce verrou. Une

fois la transaction de blocage terminée dans une instruction `COMMIT` ou `ROLLBACK`, la transaction bloquée peut se poursuivre.

La sémantique de contrôle de simultanéité multiversion d'Aurora PostgreSQL garantit l'absence de blocage des lecteurs par les dispositifs d'écriture, et vice versa. Pour que des conflits se produisent au niveau ligne, les transactions de blocage et les transactions bloquées doivent émettre des instructions conflictuelles des types suivants :

- `UPDATE`
- `SELECT … FOR UPDATE`
- `SELECT … FOR KEY SHARE`

L'instruction `SELECT … FOR KEY SHARE` est un cas particulier. La base de données utilise la clause `FOR KEY SHARE` pour optimiser les performances de l'intégrité référentielle. La présence d'un verrou de niveau ligne sur une ligne peut bloquer les commandes `INSERT`, `UPDATE` et `DELETE` sur d'autres tables qui font référence à la ligne.

Causes probables de l'augmentation du nombre d'événements d'attente

Un événement trop fréquent est généralement dû à des instructions `UPDATE`, `SELECT … FOR UPDATE` ou `SELECT … FOR KEY SHARE` combinées aux conditions suivantes.

Rubriques

- Forte simultanéité
- État Idle in transaction (Transaction inactive)
- Transactions de longue durée

Forte simultanéité

Aurora PostgreSQL peut utiliser une sémantique de verrouillage détaillée de niveau ligne. La probabilité de conflits au niveau ligne augmente lorsque les conditions suivantes sont réunies :

- Une charge de travail à forte simultanéité se dispute les mêmes lignes.
- La simultanéité augmente.

État Idle in transaction (Transaction inactive)

Parfois, la colonne `pg_stat_activity.state` affiche la valeur `idle in transaction`. Cette valeur apparaît pour les sessions qui ont entamé une transaction, mais qui n'ont pas encore émis de commande `COMMIT` ou `ROLLBACK`. Si la valeur `pg_stat_activity.state` n'est pas `active`, la requête affichée dans `pg_stat_activity` est la plus récente à avoir été exécutée. La session de blocage ne traite pas activement une requête, car une transaction ouverte comporte un verrou.

Si une transaction inactive a acquis un verrou au niveau ligne, cela peut empêcher d'autres sessions de l'acquérir. Cette condition entraîne l'apparition fréquente de l'événement d'attente `Lock:transactionid`. Pour diagnostiquer le problème, examinez la sortie de `pg_stat_activity` et `pg_locks`.

Transactions de longue durée

Les transactions qui s'exécutent depuis longtemps comportent des verrous pendant longtemps. Ces verrous de longue durée peuvent bloquer l'exécution d'autres transactions.

Actions

Le verrouillage de ligne correspond à un conflit entre les instructions `UPDATE`, `SELECT … FOR UPDATE` ou `SELECT … FOR KEY SHARE`. Avant de rechercher une solution, déterminez quand ces instructions sont exécutées sur la même ligne. Utilisez ces informations pour choisir une des stratégies décrites dans les sections suivantes.

Rubriques

- Réagissez à une forte simultanéité
- Réagissez aux transactions inactives
- Réagissez aux transactions de longue durée

Réagissez à une forte simultanéité

En cas de problème lié à la simultanéité, essayez l'une des techniques suivantes :

- Réduisez la simultanéité dans l'application. Par exemple, réduisez le nombre de sessions actives.
- Implémentez un groupe de connexions. Pour savoir comment regrouper des connexions à l'aide de RDS Proxy, consultez Proxy Amazon RDS pour Aurora.

- Concevez l'application ou le modèle de données de manière à éviter les instructions `UPDATE` et `SELECT … FOR UPDATE` conflictuelles. Vous pouvez également réduire le nombre de clés étrangères accessibles par les instructions `SELECT … FOR KEY SHARE`.

Réagissez aux transactions inactives

Si `pg_stat_activity.state` indique `idle in transaction`, utilisez les stratégies suivantes :

- Si possible, activez la validation automatique. Cette approche empêche les transactions de bloquer d'autres transactions en attendant une instruction `COMMIT` ou `ROLLBACK`.
- Recherchez les chemins de code qui ne contiennent pas d'instruction `COMMIT`, `ROLLBACK` ou `END`.
- Assurez-vous que la logique de gestion des exceptions de votre application comporte toujours un chemin vers une `end of transaction` valide.
- Assurez-vous que votre application traite les résultats des requêtes après avoir mis fin à la transaction avec `COMMIT` ou `ROLLBACK`.

Réagissez aux transactions de longue durée

Si des transactions de longue durée sont à l'origine de l'apparition fréquente de `Lock:transactionid`, essayez les stratégies suivantes :

- N'utilisez pas de verrous de ligne dans les transactions de longue durée.
- Limitez la longueur des requêtes en implémentant la validation automatique chaque fois que possible.

Lock:tuple

L'événement `Lock:tuple` se produit lorsqu'un processus backend attend d'acquérir un verrou sur un tuple.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL.

Contexte

L'événement `Lock:tuple` indique qu'un backend attend d'acquérir un verrou sur un tuple alors qu'un autre moteur détient un verrou conflictuel sur le même tuple. Le tableau suivant illustre un scénario dans lequel les sessions génèrent l'événement `Lock:tuple`.

Heure	Session 1	Session 2	Session 3
t1	Démarre une transacti on.		
t2	Met à jour la ligne 1.		
t3		Met à jour la ligne 1. La session acquiert un verrou exclusif sur le tuple, puis attend que la session 1 libère le verrou par le biais d'une validation ou d'une restauration.	
t4			Met à jour la ligne 1. La session attend que la session 2 libère le verrou exclusif sur le tuple.

Vous pouvez également simuler cet événement d'attente à l'aide de l'outil de définition de points de référence `pgbench`. Configurez un nombre élevé de sessions simultanées pour mettre à jour la même ligne au sein d'une table avec un fichier SQL personnalisé.

Pour en savoir plus sur les modes de verrouillage conflictuels, consultez Explicit Locking dans la documentation PostgreSQL. Pour en savoir plus sur `pgbench`, consultez pgbench dans la documentation PostgreSQL.

Causes probables de l'augmentation du nombre d'événements d'attente

Un événement de ce type trop fréquent peut révéler un problème de performances dont les causes sont généralement les suivantes :

- Un grand nombre de sessions simultanées tentent d'acquérir un verrou conflictuel pour le même tuple en exécutant des instructions `UPDATE` ou `DELETE`.
- Les sessions à forte simultanéité exécutent une instruction `SELECT` en utilisant les modes de verrouillage `FOR UPDATE` ou `FOR NO KEY UPDATE`.
- Divers facteurs poussent les groupes d'applications ou de connexions à ouvrir davantage de sessions pour exécuter les mêmes opérations. Comme de nouvelles sessions tentent de modifier les mêmes lignes, la charge de base de données peut augmenter et un événement `Lock:tuple` peut apparaître.

Pour en savoir plus, consultez Row-Level Locks dans la documentation PostgreSQL.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Examinez la logique de votre application
- Recherchez la session de blocage
- Réduisez la simultanéité lorsqu'elle est forte
- Résolvez les problèmes liés aux goulots d'étranglement

Examinez la logique de votre application

Déterminez si une session de blocage est restée longtemps dans l'état `idle in transaction`. Si tel est le cas, la solution à court terme peut consister à mettre fin à la session de blocage. Vous pouvez utiliser la fonction `pg_terminate_backend`. Pour en savoir plus sur cette fonction, consultez Server Signaling Functions dans la documentation PostgreSQL.

Pour une solution à long terme, procédez comme suit :

- Modifiez la logique de l'application.

- Utilisez le paramètre `idle_in_transaction_session_timeout`. Ce paramètre met fin à toute session associée à une transaction ouverte qui est restée inactive plus longtemps que la durée spécifiée. Pour en savoir plus, consultez Client Connection Defaults dans la documentation PostgreSQL.
- Chaque fois que possible, utilisez la validation automatique. Pour en savoir plus, consultez SET AUTOCOMMIT dans la documentation PostgreSQL.

Recherchez la session de blocage

Pendant l'événement d'attente `Lock:tuple`, identifiez le blocage et la session bloquée en déterminant quels verrous dépendent les uns des autres. Pour en savoir plus, consultez Lock dependency information dans le wiki PostgreSQL. Pour analyser les événements `Lock:tuple` passés, utilisez la fonction Aurora `aurora_stat_backend_waits`.

L'exemple suivant interroge toutes les sessions, en y appliquant le filtre `tuple` et en les classant par `wait_time`.

```
--AURORA_STAT_BACKEND_WAITS
      SELECT a.pid,
             a.usename,
             a.app_name,
             a.current_query,
             a.current_wait_type,
             a.current_wait_event,
             a.current_state,
             wt.type_name AS wait_type,
             we.event_name AS wait_event,
             a.waits,
             a.wait_time
        FROM (SELECT pid,
                     usename,
                     left(application_name,16) AS app_name,
                     coalesce(wait_event_type,'CPU') AS current_wait_type,
                     coalesce(wait_event,'CPU') AS current_wait_event,
                     state AS current_state,
                     left(query,80) as current_query,
                     (aurora_stat_backend_waits(pid)).*
                FROM pg_stat_activity
               WHERE pid <> pg_backend_pid()
                 AND usename<>'rdsadmin') a
NATURAL JOIN aurora_stat_wait_type() wt
```

```
NATURAL JOIN aurora_stat_wait_event() we
WHERE we.event_name = 'tuple'
    ORDER BY a.wait_time;

  pid  | usename | app_name |                 current_query                  |
 current_wait_type | current_wait_event | current_state | wait_type | wait_event |
 waits | wait_time
-------+---------+----------+------------------------------------------------
+------------------+--------------------+--------------+-----------+------------
+-------+-----------
 32136 | sys     | psql     | /*session3*/ update tab set col=1 where col=1; | Lock
           | tuple              | active        | Lock      | tuple      |     1 |
 1000018
 11999 | sys     | psql     | /*session4*/ update tab set col=1 where col=1; | Lock
           | tuple              | active        | Lock      | tuple      |     1 |
 1000024
```

Réduisez la simultanéité lorsqu'elle est forte

L'événement `Lock:tuple` peut se produire constamment, en particulier lorsque la charge de travail est élevée. Dans ce cas, réduisez la simultanéité pour les lignes très occupées. Souvent, quelques lignes seulement contrôlent une file d'attente ou la logique booléenne, ce qui explique pourquoi ces lignes sont très occupées.

Vous pouvez réduire la simultanéité en utilisant différentes approches basées sur les besoins métier, la logique de l'application et le type de charge de travail. Par exemple, vous pouvez effectuer les opérations suivantes :

- Redéfinissez la logique de votre table et de vos données pour réduire la simultanéité.
- Modifiez la logique de l'application pour réduire la simultanéité au niveau ligne.
- Exploitez et redéfinissez les requêtes avec des verrous de niveau ligne.
- Utilisez la clause `NOWAIT` lors des nouvelles tentatives.
- Envisagez d'utiliser un contrôle de simultanéité logique optimiste et à verrouillage hybride.
- Envisagez de modifier le niveau d'isolement de la base de données.

Résolvez les problèmes liés aux goulots d'étranglement

L'événement `Lock:tuple` peut se produire avec des goulots d'étranglement tels qu'une pénurie d'UC ou une saturation de la bande passante d'Amazon EBS. Pour réduire les goulots d'étranglement, adoptez les approches suivantes :

- Procédez à une augmentation d'échelle de votre type de classe d'instance.
- Optimisez les requêtes gourmandes en ressources.
- Modifiez la logique de l'application.
- Archivez les données rarement consultées.

LWLock:buffer_content (BufferContent)

L'événement `LWLock:buffer_content` se produit lorsqu'une session attend de lire ou d'écrire une page de données en mémoire alors que celle-ci est verrouillée en écriture dans une autre session. Dans Aurora PostgreSQL 13 et versions ultérieures, cet événement d'attente est appelé `BufferContent`.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL.

Contexte

Pour lire ou manipuler des données, PostgreSQL y accède via des mémoires tampons partagées. Pour lire à partir de la mémoire tampon, un processus obtient un verrou léger (LWLock) sur le contenu de la mémoire tampon en mode partagé. Pour écrire dans la mémoire tampon, il obtient ce verrou en mode exclusif. Les verrous partagés permettent à d'autres processus d'acquérir simultanément des verrous partagés sur ce contenu. Les verrous exclusifs empêchent les autres processus d'obtenir tout type de verrou sur ce contenu.

L'événement `LWLock:buffer_content` (`BufferContent`) indique que plusieurs processus tentent de générer des verrous légers (LWLocks) sur le contenu d'une mémoire tampon spécifique.

Causes probables de l’augmentation du nombre d’événements d’attente

Un événement `LWLock:buffer_content` (`BufferContent`) trop fréquent peut révéler un problème de performances dont les causes sont généralement les suivantes :

Augmentation des mises à jour simultanées des mêmes données

Le nombre de sessions simultanées associées à des requêtes de mise à jour du même contenu de mémoire tampon peut augmenter. Ce conflit peut être plus marqué sur les tables contenant beaucoup d’index.

Les données de la charge de travail ne sont pas en mémoire

Lorsque les données traitées par la charge de travail active ne sont pas en mémoire, la fréquence de ces événements d’attente peut augmenter. Cet effet est dû au fait que les processus détenant des verrous peuvent les conserver plus longtemps pendant qu’ils effectuent des opérations d’I/O disque.

Utilisation excessive de contraintes de clé étrangère

Les contraintes de clé étrangère peuvent augmenter la durée pendant laquelle un processus conserve un verrou de contenu de mémoire tampon. Cet effet est dû au fait que les opérations de lecture ont besoin d’un verrou de contenu de mémoire tampon partagée sur la clé référencée pendant la mise à jour de cette clé.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d’attente. Vous pouvez identifier les événements `LWLock:buffer_content` (`BufferContent`) en utilisant Analyse des performances d’Amazon RDS ou en interrogeant la vue `pg_stat_activity`.

Rubriques

- Améliorez l’efficacité en mémoire
- Réduction de l’utilisation des contraintes de clé étrangère
- Supprimez les index inutilisés
- Suppression des index dupliqués
- Suppression ou RÉINDEXATION des index non valides
- Utilisation d’index partiels

- Suppression du gonflement des tables et des index

Améliorez l'efficacité en mémoire

Pour que les données de la charge de travail active aient plus de chances d'être mises en mémoire, partitionnez les tables ou procédez à une augmentation d'échelle de votre classe d'instance. Pour plus d'informations sur les classes d'instance de base de données, consultez Classes d'instance de base de données Amazon Aurora.

Surveillez la métrique `BufferCacheHitRatio`, qui mesure le pourcentage de demandes qui sont traitées par le cache des tampons d'une instance de base de données dans votre cluster de bases de données. Cette métrique fournit des informations sur la quantité de données traitées par la mémoire. Un taux de réussite élevé indique que votre instance de base de données dispose de suffisamment de mémoire pour votre ensemble de données de travail, tandis qu'un faible taux indique que vos requêtes accèdent fréquemment aux données à partir du stockage.

Dans la section Paramètres de la mémoire du rapport PG Collector, les résultats de lecture du cache par table et par index peuvent fournir des informations sur le taux d'accès au cache pour les tables et les index.

Réduction de l'utilisation des contraintes de clé étrangère

Examinez les charges de travail qui présentent un nombre élevé d'événements d'attente `LWLock:buffer_content` (`BufferContent`) pour déterminer si des contraintes de clé étrangère sont utilisées. Supprimez les contraintes de clé étrangère inutiles.

Supprimez les index inutilisés

Pour les charges de travail qui présentent un nombre élevé d'événements d'attente `LWLock:buffer_content` (`BufferContent`), identifiez les index inutilisés et supprimez-les.

La section des index inutilisés du rapport PG Collector peut fournir des informations sur les index inutilisés dans la base de données.

Suppression des index dupliqués

Identifiez les index dupliqués et supprimez-les.

La section des index dupliqués du rapport PG Collector peut fournir des informations sur les index dupliqués dans la base de données.

Suppression ou RÉINDEXATION des index non valides

Les index non valides surviennent généralement lors de l'utilisation de `CREATE INDEX CONCURRENTLY` ou `REINDEX CONCURRENTLY`. Dans ce cas, la commande échoue ou est abandonnée.

Les index non valides ne peuvent pas être utilisés pour les requêtes, mais ils sont tout de même mis à jour et occupent de l'espace disque.

La section Indexes non valides du rapport PG Collector peut fournir des informations sur les index non valides dans la base de données.

Utilisation d'index partiels

Des index partiels peuvent être utilisés pour améliorer les performances des requêtes et réduire la taille de l'index. Ils reposent sur un sous-ensemble d'une table, qui est défini par une expression conditionnelle. Comme indiqué dans la documentation sur les index partiels, ils peuvent réduire la charge de travail liée à la maintenance des index, car PostgreSQL n'a pas besoin de mettre à jour l'index dans tous les cas.

Suppression du gonflement des tables et des index

Un gonflement excessif des tables et des index peut avoir un impact négatif sur les performances de la base de données. Les tables et les index gonflés augmentent la taille de l'ensemble de travail actif, dégradant ainsi l'efficacité de la mémoire. En outre, le gonflement augmente les coûts de stockage et ralentit l'exécution des requêtes. Pour diagnostiquer les problèmes de gonflement, consultez Diagnostic du gonflement de la table et de l'index. La section Fragmentation (Bloat) du rapport PG Collector peut fournir des informations sur le gonflement des tables et des index.

Pour remédier au gonflement des tables et des index, plusieurs options s'offrent à vous :

VACUUM FULL

> `VACUUM FULL` crée une copie de la table, en ne gardant que les tuples dynamiques, puis remplace l'ancienne table par la nouvelle tout en appliquant un verrou `ACCESS EXCLUSIVE`. En empêchant toute lecture ou écriture dans la table, cela peut provoquer une panne. De plus, `VACUUM FULL` prend plus de temps si la table est volumineuse.

pg_repack

> `pg_repack` est utile dans les situations où `VACUUM FULL` pourrait ne pas convenir. Il crée une table qui contient les données de la table gonflée, suit les modifications par rapport à la

table d'origine, puis remplace la table d'origine par la nouvelle. Il ne verrouille pas la table d'origine pour les opérations de lecture ou d'écriture pendant la création de la nouvelle table. Pour découvrir comment utiliser `pg_repack`, consultez Suppression du gonflement avec pg_repack et pg_repack.

REINDEX

La commande `REINDEX` peut être utilisée pour remédier au gonflement de l'index. `REINDEX` écrit une nouvelle version de l'index sans les pages inactives ni les pages vides ou presque vides, réduisant ainsi sa consommation d'espace. Pour des informations détaillées sur la commande `REINDEX`, reportez-vous à la documentation de REINDEX.

Après avoir éliminé le gonflement des tables et des index, il peut être nécessaire d'augmenter la fréquence de la fonctionnalité d'autovacuum sur ces tables. La mise en œuvre de paramètres d'autovacuum agressifs au niveau des tables contribue à prévenir les futurs gonflements. Pour plus d'informations, consultez la documentation sur `Vacuuming and analyzing tables automatically`.

LWLock:buffer_mapping

Cet événement se produit lorsqu'une session attend d'associer un bloc de données à une mémoire tampon dans le groupe de mémoires tampons partagées.

> **Note**
>
> Cet événement apparaît sous la forme `LWLock:buffer_mapping` dans Aurora PostgreSQL 12 et versions antérieures, et sous la forme `LWLock:BufferMapping` dans Aurora PostgreSQL 13 et versions ultérieures.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes
- Actions

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à Aurora PostgreSQL 9.6 et versions ultérieures.

Contexte

Le groupe de mémoires tampons partagées est une zone de mémoire d'Aurora PostgreSQL qui contient toutes les pages actuellement ou précédemment utilisées par les processus. Lorsqu'un processus a besoin d'une page, il la lit dans le groupe de mémoires tampons partagées. Le paramètre `shared_buffers` définit la taille de la mémoire tampon partagée et réserve une zone de mémoire pour stocker la table et les pages d'index. Si vous modifiez ce paramètre, veillez à redémarrer la base de données. Pour de plus amples informations, consultez Mémoires tampons partagées.

L'événement d'attente `LWLock:buffer_mapping` se produit dans les scénarios suivants :

- Un processus recherche une page dans la table des mémoires tampons et acquiert un verrou de mappage de mémoire tampon partagée.
- Un processus charge une page dans le groupe de mémoires tampons et acquiert un verrou exclusif de mappage de mémoire tampon.
- Un processus supprime une page du groupe et acquiert un verrou exclusif de mappage de mémoire tampon.

Causes

Lorsque cet événement se produit plus souvent qu'à l'accoutumée, indiquant un possible problème de performances, la base de données effectue une pagination dans et hors du groupe de mémoires tampons partagées. Les causes sont généralement les suivantes :

- Requêtes volumineuses
- Index et tables gonflés
- Analyses de tables complètes
- Taille de groupe partagé inférieure à celle de l'ensemble de travail

Actions

Nous vous recommandons différentes actions en fonction de l'origine de votre événement d'attente.

Rubriques

- Surveillez les métriques liées à la mémoire tampon
- Évaluez votre stratégie d'indexation
- Réduisez le nombre de mémoires tampons qui doivent être allouées rapidement

Surveillez les métriques liées à la mémoire tampon

Lorsque les événements d'attente `LWLock:buffer_mapping` atteignent un pic, examinez le taux d'accès à la mémoire tampon. Vous pouvez utiliser ces métriques pour mieux comprendre ce qui se passe dans le cache de mémoire tampon. Examinez les métriques suivantes :

`BufferCacheHitRatio`

Cette métrique Amazon CloudWatch mesure le pourcentage de requêtes qui sont traitées par le cache de mémoire tampon d'une instance de base de données dans votre cluster de bases de données. Vous verrez peut-être cette métrique diminuer dans la période précédant l'événement d'attente `LWLock:buffer_mapping`.

`blks_hit`

Cette métrique de compteur Performance Insights indique le nombre de blocs qui ont été récupérés à partir du groupe de mémoires tampons partagées. Après l'apparition de l'événement d'attente `LWLock:buffer_mapping`, vous pouvez observer un pic dans `blks_hit`.

`blks_read`

Cette mesure de compteur Performance Insights indique le nombre de blocs pour lesquels une lecture des I/O a été nécessaire dans le groupe de mémoires tampons partagées. Vous observerez peut-être un pic de `blks_read` dans la période précédant l'événement d'attente `LWLock:buffer_mapping`.

Évaluez votre stratégie d'indexation

Pour vous assurer que votre stratégie d'indexation ne nuit pas aux performances, vérifiez les éléments suivants :

Gonflement des index

Assurez-vous que le gonflement des index et des tables n'entraîne pas la lecture de pages inutiles dans la mémoire tampon partagée. Si vos tables contiennent des lignes inutilisées, archivez les

données et supprimez les lignes des tables. Vous pouvez ensuite reconstruire les index des tables redimensionnées.

Index pour les requêtes fréquemment utilisées

Pour déterminer si vos index sont optimaux, surveillez les métriques du moteur de base de données dans Performance Insights. La métrique `tup_returned` indique le nombre de lignes lues. La métrique `tup_fetched` indique le nombre de lignes renvoyées au client. Si la métrique `tup_returned` est nettement supérieure à la métrique `tup_fetched`, les données risquent de ne pas être correctement indexées. De plus, les statistiques de votre table ne sont peut-être pas à jour.

Réduisez le nombre de mémoires tampons qui doivent être allouées rapidement

Pour réduire le nombre d'événements d'attente `LWLock:buffer_mapping`, essayez de réduire le nombre de mémoires tampons qui doivent être allouées rapidement. Une stratégie consiste à effectuer des opérations par lots de plus petite taille. Vous pouvez obtenir des lots plus petits en partitionnant vos tables.

LWLock:BufferIO (IPC:BufferIO)

L'événement `LWLock:BufferIO` se produit lorsqu'Aurora PostgreSQL ou RDS for PostgreSQL attend que d'autres processus terminent leurs opérations d'entrée/sortie (I/O) en cas de tentative simultanée d'accès à une page. Le but est que la même page soit lue dans la mémoire tampon partagée.

Rubriques

Versions de moteur pertinentes

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL. Pour Aurora PostgreSQL 12 et versions antérieures, cet événement d'attente est nommé lwlock:buffer_io, tandis qu'il est nommé lwlock:bufferio dans la version Aurora

PostgreSQL 13. Depuis la version Aurora PostgreSQL 14, l'événement d'attente BufferIO est passé du type d'événement d'attente (IPC:Bufferio) `LWLock` à `IPC`.

Contexte

Chaque mémoire tampon partagée possède un verrou I/O qui est associé à l'événement d'attente `LWLock:BufferIO`, chaque fois qu'un bloc (ou une page) doit être récupéré en dehors du groupe de mémoires tampons partagées.

Ce verrou est utilisé pour gérer plusieurs sessions qui ont toutes besoin d'accéder au même bloc. Ce bloc doit être lu en dehors du groupe de mémoires tampons partagées, défini par le paramètre `shared_buffers`.

Dès que la page est lue dans le groupe de mémoires tampons partagées, le verrou `LWLock:BufferIO` est libéré.

> **Note**
>
> L'événement d'attente `LWLock:BufferIO` précède l'événement d'attente IO:DataFileRead. L'événement d'attente `IO:DataFileRead` se produit lorsque les données sont lues à partir du stockage.

Pour en savoir plus sur les verrous légers, consultez Présentation du verrouillage.

Causes

Les principales causes de l'événement d'attente `LWLock:BufferIO` sont les suivantes :

- Plusieurs backends ou connexions tentant d'accéder à la même page qui est également en attente d'une opération I/O
- Rapport entre la taille du groupe de mémoires tampons partagées (défini par le paramètre `shared_buffers`) et le nombre de mémoires tampons nécessaires à la charge de travail actuelle
- La taille du groupe de mémoires tampons partagées n'est pas bien équilibrée par rapport au nombre de pages expulsées par d'autres opérations
- Index volumineux ou gonflés qui obligent le moteur à lire plus de pages que nécessaire dans le groupe de mémoires tampons partagées
- Absence d'index qui oblige le moteur de base de données à lire plus de pages que nécessaire dans les tables

- Pics soudains de connexions à la base de données tentant d'effectuer des opérations sur la même page

Actions

Nous vous recommandons différentes actions en fonction de l'origine de votre événement d'attente :

- Observez les métriques Amazon CloudWatch pour établir une corrélation entre les fortes diminutions de `BufferCacheHitRatio` et les événements d'attente `LWLock:BufferIO`. Cet effet peut indiquer que vous disposez d'un petit paramètre de mémoires tampons partagées. Il peut être nécessaire de l'augmenter ou de procéder à une augmentation d'échelle de votre classe d'instance de base de données. Vous pouvez décomposer votre charge de travail en plusieurs nœuds de lecture.
- Recherchez les index inutilisés et supprimez-les.
- Utilisez des tables partitionnées (qui comportent également des index partitionnés). Cela permet de limiter la réorganisation des index et de réduire son impact.
- Évitez d'indexer inutilement des colonnes.
- Empêchez les pics soudains de connexions à la base de données en utilisant un groupe de connexions.
- En guise de bonne pratique, limitez le nombre maximal de connexions à la base de données.

LWLock:lock_manager

Cet événement se produit lorsque le moteur Aurora PostgreSQL conserve la zone de mémoire du verrou partagé pour allouer, vérifier et annuler l'allocation d'un verrou parce qu'il est impossible d'utiliser un verrou à chemin d'accès rapide.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à Aurora PostgreSQL 9.6 et versions ultérieures.

Contexte

Lorsque vous émettez une instruction SQL, Aurora PostgreSQL enregistre des verrous pour protéger la structure, les données et l'intégrité de votre base de données pendant les opérations simultanées. Le moteur peut atteindre cet objectif en utilisant un verrou à chemin d'accès rapide ou non rapide. Un verrou à chemin d'accès non rapide est plus coûteux et génère plus de frais qu'un verrou à chemin d'accès rapide.

Verrouillage à chemin d'accès rapide

Pour réduire les frais liés aux verrous qui sont fréquemment acquis et libérés, mais qui entrent rarement en conflit, les processus backend peuvent utiliser le verrouillage à chemin d'accès rapide. La base de données utilise ce mécanisme pour les verrous qui répondent aux critères suivants :

- Ils utilisent la méthode de verrouillage DEFAULT.
- Ils représentent un verrou sur une relation de base de données plutôt qu'une relation partagée.
- Il s'agit de verrous faibles qui sont peu susceptibles d'entrer en conflit.
- Le moteur peut rapidement vérifier qu'aucun verrou conflictuel ne peut exister.

Le moteur ne peut pas utiliser de verrouillage à chemin d'accès rapide lorsque l'une des conditions suivantes est vraie :

- Le verrou ne répond pas aux critères précédents.
- Il n'y a plus d'emplacements disponibles pour le processus backend.

Pour en savoir plus sur le verrouillage à chemin d'accès rapide, consultez fast path dans le fichier README du gestionnaire de verrous PostgreSQL et pg-locks dans la documentation PostgreSQL.

Exemple de problème de mise à l'échelle pour le gestionnaire de verrous

Dans cet exemple, une table nommée `purchases` stocke cinq ans de données, partitionnées par jour. Chaque partition possède deux index. La séquence d'événements suivante se produit :

1. Vous interrogez des données réparties sur différents jours, ce qui oblige la base de données à lire de nombreuses partitions.
2. La base de données crée une entrée de verrou pour chaque partition. Si les index de partition font partie du chemin d'accès de l'optimiseur, la base de données crée également une entrée de verrou pour eux.
3. Lorsque le nombre d'entrées de verrou demandées pour le même processus backend est supérieur à 16, ce qui correspond à la valeur de `FP_LOCK_SLOTS_PER_BACKEND`, le gestionnaire de verrous utilise la méthode de verrouillage à chemin d'accès non rapide.

Les applications modernes peuvent comporter des centaines de sessions. Si des sessions simultanées interrogent le parent sans élaguer correctement les partitions, la base de données peut créer des centaines, voire des milliers, de verrous à chemin d'accès non rapide. En général, lorsque cette simultanéité est supérieure au nombre de vCPU, l'événement d'attente `LWLock:lock_manager` apparaît.

> **Note**
>
> L'événement d'attente `LWLock:lock_manager` n'est pas lié au nombre de partitions ou d'index contenus dans un schéma de base de données. Il est plutôt lié au nombre de verrous à chemin d'accès non rapide que la base de données doit contrôler.

Causes probables de l'augmentation du nombre d'événements d'attente

Lorsque l'événement d'attente `LWLock:lock_manager` se produit plus souvent qu'à l'accoutumée, indiquant un possible problème de performances, les causes les plus probables des pics soudains sont les suivantes :

- Les sessions actives simultanées exécutent des requêtes qui n'utilisent pas de verrous à chemin d'accès rapide. Ces sessions dépassent également le nombre maximum de vCPU.
- Un grand nombre de sessions actives simultanées accèdent à une table fortement partitionnée. Chaque partition possède plusieurs index.
- La base de données subit une tempête de connexions. Par défaut, certaines applications et certains logiciels de regroupement de connexions créent davantage de connexions lorsque la base de données est lente. Cette pratique aggrave le problème. Réglez le logiciel de regroupement de connexions de manière à éviter les tempêtes de connexions.

- Un grand nombre de sessions interrogent une table parente sans élaguer les partitions.
- Un langage de définition de données (DDL), un langage de manipulation de données (DML) ou une commande de maintenance verrouille exclusivement une relation occupée ou des tuples fréquemment consultés ou modifiés.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

Rubriques

- Élaguez les partitions
- Supprimez les index inutiles
- Réglez vos requêtes pour qu'elles utilisent le verrouillage à chemin d'accès rapide
- Procédez au réglage d'autres événements d'attente
- Réduisez les goulots d'étranglement matériels
- Utilisez une fonction de regroupement de connexions
- Mettez à niveau votre version d'Aurora PostgreSQL

Élaguez les partitions

L'élagage des partitions est une stratégie d'optimisation des requêtes qui exclut les partitions inutiles des analyses de tables, améliorant ainsi les performances. L'élagage des partitions est activé par défaut. S'il est désactivé, activez-le comme suit.

```
SET enable_partition_pruning = on;
```

Les requêtes peuvent tirer parti de l'élagage des partitions lorsque leur clause `WHERE` contient la colonne utilisée pour le partitionnement. Pour en savoir plus, consultez Partition Pruning dans la documentation PostgreSQL.

Supprimez les index inutiles

Votre base de données peut contenir des index inutilisés ou rarement utilisés. Si tel est le cas, pensez à les supprimer. Effectuez l’une des actions suivantes :

- Pour en savoir plus sur la recherche des index inutiles, consultez Unused Indexes dans le wiki PostgreSQL.

- Exécutez PG Collector. Ce script SQL rassemble les informations de la base de données et les présente sous forme de rapport HTML. Consultez la section « Unused indexes » (Index inutilisés). Pour en savoir plus, consultez pg-collector dans le référentiel GitHub AWS Labs.

Réglez vos requêtes pour qu'elles utilisent le verrouillage à chemin d'accès rapide

Pour savoir si vos requêtes utilisent le verrouillage à chemin d'accès rapide, interrogez la colonne `fastpath` de la table `pg_locks`. Si vos requêtes n'utilisent pas le verrouillage à chemin d'accès rapide, essayez de réduire le nombre de relations par requête à moins de 16.

Procédez au réglage d'autres événements d'attente

Si `LWLock:lock_manager` est premier ou deuxième dans la liste des attentes les plus fréquentes, vérifiez si les événements d'attente suivants apparaissent également dans la liste :

- `Lock:Relation`
- `Lock:transactionid`
- `Lock:tuple`

S'ils figurent parmi les premiers de la liste, commencez par régler ces événements d'attente. Ces événements peuvent être un moteur pour `LWLock:lock_manager`.

Réduisez les goulots d'étranglement matériels

Un goulot d'étranglement matériel peut se produire, comme une pénurie d'UC ou une saturation de votre bande passante Amazon EBS. Envisagez alors de réduire les goulots d'étranglement matériels. Procédez comme suit :

- Procédez à une augmentation d'échelle de votre classe d'instance.
- Optimisez les requêtes qui sollicitent énormément l'UC et la mémoire.
- Modifiez la logique de votre application.
- Archivez vos données.

Pour en savoir plus sur l'UC, la mémoire et la bande passante réseau EBS, consultez Types d'instances Amazon RDS.

Utilisez une fonction de regroupement de connexions

Si le nombre total de connexions actives dépasse le nombre maximal de vCPU, cela signifie que l'UC requise par les processus du système d'exploitation est supérieure à ce que votre type d'instance peut prendre en charge. Dans ce cas, vous pouvez utiliser ou régler un groupe de connexions. Pour en savoir plus sur les vCPU relatifs à votre type d'instance, consultez Types d'instances Amazon RDS.

Pour en savoir plus sur les groupes de connexions, consultez les ressources suivantes :

- Proxy Amazon RDS pour Aurora
- pgbouncer
- Connection Pools and Data Sources dans la documentation PostgreSQL

Mettez à niveau votre version d'Aurora PostgreSQL

Si votre version actuelle d'Aurora PostgreSQL est inférieure à la version 12, procédez à une mise à niveau vers la version 12 ou ultérieure. Les versions 12 et 13 de PostgreSQL disposent d'un mécanisme de partition amélioré. Pour en savoir plus la version 12, consultez le document PostgreSQL 12.0 Release Notes. Pour en savoir plus sur la mise à niveau d'Aurora PostgreSQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora PostgreSQL.

LWLock:MultiXact

Les événements d'attente `LWLock:MultiXactMemberBuffer`, `LWLock:MultiXactOffsetBuffer`, `LWLock:MultiXactMemberSLRU` et `LWLock:MultiXactOffsetSLRU` indiquent qu'une session attend de récupérer une liste de transactions qui modifient la même ligne dans une table donnée.

- `LWLock:MultiXactMemberBuffer` : un processus attend des E/S sur un tampon simple le moins récemment utilisé (SLRU) pour un membre multixact.
- `LWLock:MultiXactMemberSLRU` : un processus attend d'accéder au cache simple le moins récemment utilisé (SLRU) pour un membre multixact.
- `LWLock:MultiXactOffsetBuffer` : un processus attend des E/S sur un tampon simple le moins récemment utilisé (SLRU) pour un décalage multixact.
- `LWLock:MultiXactOffsetSLRU` : un processus attend d'accéder au cache simple le moins récemment utilisé (SLRU) pour un décalage multixact.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL.

Contexte

Un multixact est une structure de données qui stocke une liste d'identifiants de transaction (XID) qui modifient la même ligne de table. Lorsqu'une transaction unique fait référence à une ligne dans une table, l'ID de la transaction est stocké dans la ligne d'en-tête de la table. Lorsque plusieurs transactions font référence à la même ligne dans une table, la liste des ID de transaction est stockée dans la structure de données multixact. Les événements d'attente multixact indiquent qu'une session est en train de récupérer dans la structure de données la liste des transactions qui font référence à une ligne donnée d'une table.

Causes probables de l'augmentation du nombre d'événements d'attente

Les trois causes courantes de l'utilisation de multixact sont les suivantes :

- Sous-transactions à partir de points de sauvegarde explicites – La création explicite d'un point de sauvegarde dans vos transactions génère de nouvelles transactions pour la même ligne. Par exemple, en utilisant `SELECT FOR UPDATE`, puis `SAVEPOINT`, puis `UPDATE`.

  Certains pilotes, mappeurs objet-relationnel (ORM) et couches d'abstraction ont des options de configuration pour envelopper automatiquement toutes les opérations avec des points de sauvegarde. Cela peut générer de nombreux événements d'attente multixact dans certaines charges de travail. L'option `autosave` du pilote JDBC de PostgreSQL en est un exemple. Pour plus d'informations, consultez pgJDBC dans la documentation PostgreSQL. Un autre exemple est le pilote ODBC de PostgreSQL et son option `protocol`. Pour plus d'informations, consultez psqlODBC Configuration Options (Options de configuration de psqlODBC) dans la documentation du pilote ODBC PostgreSQL.

- Sous-transactions à partir de clauses EXCEPTION PL/pgSQL — Chaque clause `EXCEPTION` que vous écrivez dans vos fonctions ou procédures PL/pgSQL crée un `SAVEPOINT` en interne.
- Clés étrangères : plusieurs transactions acquièrent un verrou partagé sur l'enregistrement parent.

Lorsqu'une ligne donnée est incluse dans une opération à transactions multiples, le traitement de la ligne nécessite de récupérer les ID des transactions dans les listings `multixact`. Si les recherches ne peuvent pas obtenir le multixact à partir du cache mémoire, la structure de données doit être lue à partir de la couche de stockage Aurora. Ces E/S du stockage signifient que les requêtes SQL peuvent prendre plus de temps. Les pertes de mémoire cache peuvent commencer à se produire en cas d'utilisation intensive due à un grand nombre de transactions multiples. Tous ces facteurs contribuent à l'augmentation de cet événement d'attente.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente. Certaines de ces actions peuvent contribuer à réduire immédiatement les temps d'attente. Cependant, d'autres peuvent nécessiter des recherches supplémentaires et des corrections pour mettre à l'échelle votre charge de travail.

Rubriques

- Procéder au gel du processus vacuum sur les tables comportant cet événement d'attente
- Augmenter la fréquence d'autovacuum sur les tables comportant cet événement d'attente
- Augmenter les paramètres de mémoire
- Réduire les transactions de longue durée
- Actions à long terme

Procéder au gel du processus vacuum sur les tables comportant cet événement d'attente

Si cet événement d'attente connaît un pic soudain et affecte votre environnement de production, vous pouvez utiliser l'une des méthodes temporaires suivantes pour en réduire le nombre.

- Utilisez VACUUM FREEZE sur la table ou la partition de table concernée pour résoudre le problème immédiatement. Pour plus d'informations, consultez VACUUM.
- Utilisez la clause VACUUM (FREEZE, INDEX_CLEANUP FALSE) pour effectuer un vacuum rapide en omettant les index. Pour plus d'informations, consultez Vidage d'une table le plus rapidement possible.

Augmenter la fréquence d'autovacuum sur les tables comportant cet événement d'attente

Après avoir analysé toutes les tables de toutes les bases de données, VACUUM finira par supprimer les multixacts, et leurs valeurs multixact les plus anciennes seront avancées. Pour plus d'informations, consultez Multixacts et bouclage. Pour réduire au minimum le nombre d'événements d'attente LWLock:MultiXact, vous devez exécuter la fonction VACUUM aussi souvent que nécessaire. Pour ce faire, assurez-vous que la fonction VACUUM de votre cluster de bases de données Aurora PostgreSQL est configurée de manière optimale.

Si l'utilisation de VACUUM FREEZE sur la table ou la partition de table concernée résout le problème d'événement d'attente, nous vous recommandons d'utiliser un planificateur, tel que `pg_cron`, pour exécuter le VACUUM au lieu de régler l'autovacuum au niveau de l'instance.

Pour que l'autovacuum ait lieu plus fréquemment, vous pouvez réduire la valeur du paramètre de stockage `autovacuum_multixact_freeze_max_age` dans le tableau concerné. Pour plus d'informations, consultez autovacuum_multixact_freeze_max_age.

Augmenter les paramètres de mémoire

Vous pouvez optimiser l'utilisation de la mémoire pour les caches multixact en ajustant les paramètres suivants. Ces paramètres contrôlent la quantité de mémoire réservée à ces caches, ce qui peut contribuer à réduire les temps d'attente de multixact dans votre charge de travail. Nous vous recommandons de commencer par les valeurs suivantes :

Pour Aurora PostgreSQL 17 et versions ultérieures :

- `multixact_offset_buffers` = 128
- `multixact_member_buffers` = 256

Pour Aurora PostgreSQL 16 et versions antérieures :

- `multixact_offsets_cache_size` = 128
- `multixact_members_cache_size` = 256

> **Note**
>
> Dans Aurora PostgreSQL 17, les noms des paramètres sont passés de `multixact_offsets_cache_size` à `multixact_offset_buffers` et de `multixact_members_cache_size` à `multixact_member_buffers` pour s'aligner sur la communauté PostgreSQL 17.

Vous pouvez définir ces paramètres au niveau du cluster afin que toutes ses instances restent cohérentes. Nous vous recommandons de tester et d'ajuster les valeurs en fonction de vos exigences spécifiques en matière de charge de travail et de classe d'instance. Redémarrez l'instance d'enregistreur pour que la modification des paramètres prenne effet.

Les paramètres sont exprimés en termes d'entrées de cache multixact. Chaque entrée de cache utilise `8 KB` de mémoire. Pour calculer la mémoire totale réservée, multipliez la valeur de chaque paramètre par `8 KB`. Par exemple, si vous définissez un paramètre sur 128, la mémoire réservée totale sera de `128 * 8 KB = 1 MB`.

Réduire les transactions de longue durée

Les transactions de longue durée font que le vacuum conserve les informations de la transaction jusqu'à ce qu'elle soit validée ou jusqu'à ce que la transaction en lecture seule soit fermée. Nous vous recommandons de surveiller et de gérer de manière proactive les transactions de longue durée. Pour plus d'informations, consultez La base de données a une connexion de longue durée à l'état Transaction inactive. Essayez de modifier votre application pour éviter ou minimiser l'utilisation des transactions de longue durée.

Actions à long terme

Examinez votre charge de travail pour identifier la cause des débordements de multixact. Vous devez résoudre le problème afin de pouvoir mettre à l'échelle votre charge de travail et de réduire l'événement d'attente.

- Vous devez analyser le langage DDL utilisé pour créer vos tables. Assurez-vous que les structures des tables et les index sont bien conçus.
- Lorsque les tables concernées possèdent des clés étrangères, déterminez si elles sont nécessaires ou s'il existe un autre moyen d'appliquer l'intégrité référentielle.
- Lorsqu'une table contient de grands index inutilisés, la fonction autovacuum peut ne pas être adaptée à votre charge de travail et empêcher son exécution. Pour éviter cela, recherchez la présence d'index inutilisés et supprimez-les complètement. Pour plus d'informations, consultez Gestion de la fonction autovacuum avec de grands index.
- Réduisez l'utilisation des points de sauvegarde dans vos transactions.

LWLock:pg_stat_statements

L'événement d'attente LWLock:pg_stat_statements se produit lorsque l'extension `pg_stat_statements` verrouille de manière exclusive la table de hachage qui suit les instructions SQL. Cela se produit dans les scénarios suivants :

- Lorsque le nombre d'instructions suivies atteint la valeur de paramètre `pg_stat_statements.max` configurée et qu'il est nécessaire de faire de la place à d'autres entrées, l'extension effectue un tri sur le nombre d'appels, supprime les 5 % des instructions les moins exécutées et remplit à nouveau le hachage avec les entrées restantes.
- Lorsque `pg_stat_statements` effectue une opération `garbage collection` sur le fichier `pgss_query_texts.stat` sur le disque et réécrit le fichier.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente sont prises en charge pour toutes les versions de Aurora PostgreSQL.

Contexte

Comprendre l'extension pg_stat_statements : l'extension pg_stat_statements suit les statistiques d'exécution des instructions SQL dans une table de hachage. L'extension suit les instructions SQL jusqu'à la limite définie par le paramètre `pg_stat_statements.max`. Ce paramètre détermine le nombre maximum d'instructions pouvant être suivies, ce qui correspond au nombre maximum de lignes dans la vue pg_stat_statements.

Persistance des statistiques des instructions : l'extension conserve les statistiques des instructions lors des redémarrages de l'instance en :

- Écrivant des données dans un fichier nommé pg_stat_statements.stat

- Utilisant le paramètre pg_stat_statements.save pour contrôler le comportement de persistance

Lorsque pg_stat_statements.save est :

- activé (par défaut) : les statistiques sont enregistrées à l'arrêt et rechargées au démarrage du serveur
- désactivé : les statistiques ne sont ni enregistrées à l'arrêt, ni rechargées au démarrage du serveur

Stockage du texte des requêtes : l'extension stocke le texte des requêtes suivies dans un fichier nommé `pgss_query_texts.stat`. Ce fichier peut atteindre le double de la taille moyenne de toutes les instructions SQL suivies avant le récupérateur de mémoire. L'extension nécessite un verrouillage exclusif de la table de hachage pendant les opérations de nettoyage et de réécriture du fichier `pgss_query_texts.stat`.

Processus d'annulation de l'allocation des instructions : lorsque le nombre d'instructions suivies atteint la limite `pg_stat_statements.max` et que de nouvelles instructions doivent être suivies, l'extension :

- Utilise un verrou exclusif (LWLock:pg_stat_statements) sur la table de hachage.
- Charge les données existantes dans la mémoire locale.
- Effectue un tri rapide en fonction du nombre d'appels.
- Supprime les instructions les moins appelées (les 5 % inférieures).
- Reremplit la table de hachage avec les entrées restantes.

Surveillance de l'annulation de l'allocation des instructions : dans PostgreSQL 14 et versions ultérieures, vous pouvez surveiller l'annulation de l'allocation des instructions à l'aide de la vue pg_stat_statements_info. Cette vue inclut une colonne dealloc qui indique le nombre de fois où l'allocation des instructions a été annulée pour faire de la place à de nouvelles.

Si l'annulation de l'allocation des instructions se produit fréquemment, cela accroît la fréquence du récupérateur de mémoire au niveau du fichier `pgss_query_texts.stat` sur le disque.

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes typiques de l'allongement des temps d'attente de `LWLock:pg_stat_statements` sont les suivantes :

- Augmentation du nombre de requêtes uniques utilisées par l'application.
- La valeur du paramètre `pg_stat_statements.max` est faible par rapport au nombre de requêtes uniques utilisées.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente. Vous pouvez identifier les événements `LWLock:pg_stat_statements` en utilisant Analyse des performances d'Amazon RDS ou en interrogeant la vue `pg_stat_activity`.

Ajustez les paramètres `pg_stat_statements` suivants pour contrôler le comportement de suivi et réduire les événements d'attente liés aux instructions LWLock:pg_stat_.

Rubriques

- Désactivation du paramètre pg_stat_statements.track
- Augmentation de la valeur du paramètre pg_stat_statements.max
- Désactivation du paramètre pg_stat_statements.track_utility

Désactivation du paramètre pg_stat_statements.track

Si l'événement d'attente LWLock:pg_stat_statements a un impact négatif sur les performances de la base de données et qu'une solution rapide est nécessaire avant de poursuivre l'analyse de la vue `pg_stat_statements` afin d'identifier la cause première, le paramètre `pg_stat_statements.track` peut être désactivé en le définissant sur `none`. Cela désactive la collecte des statistiques des instructions.

Augmentation de la valeur du paramètre pg_stat_statements.max

Pour réduire les annulations d'allocation d'instructions et minimiser le récupérateur de mémoire du fichier `pgss_query_texts.stat` sur le disque, augmentez la valeur du paramètre `pg_stat_statements.max`. La valeur par défaut est `5,000`.

> **Note**
>
> Le paramètre `pg_stat_statements.max` est statique. Vous devez redémarrer votre instance de base de données pour appliquer les éventuelles modifications apportées à ce paramètre.

Désactivation du paramètre pg_stat_statements.track_utility

Vous pouvez analyser la vue pg_stat_statements pour déterminer les commandes utilitaires consommant le plus de ressources suivies par `pg_stat_statements`.

Le paramètre `pg_stat_statements.track_utility` contrôle si le module suit les commandes utilitaires, qui incluent toutes les commandes sauf SELECT, INSERT, UPDATE, DELETE et MERGE. Par défaut, ce paramètre est défini sur on.

Par exemple, lorsque votre application utilise de nombreuses requêtes de point de sauvegarde, qui sont intrinsèquement uniques, cela peut augmenter l'annulation de l'allocation des instructions. Pour résoudre ce problème, vous pouvez désactiver le paramètre `pg_stat_statements.track_utility` pour empêcher `pg_stat_statements` de suivre les requêtes de point de sauvegarde.

> **Note**
>
> Le paramètre `pg_stat_statements.track_utility` est un paramètre dynamique. Vous pouvez modifier sa valeur sans redémarrer votre instance de base de données.

Example Exemple de requêtes de point de sauvegarde uniques dans pg_stat_statements

```
                    query                         |       queryid
--------------------------------------------------+---------------------
 SAVEPOINT JDBC_SAVEPOINT_495701                  | -7249565344517699703
 SAVEPOINT JDBC_SAVEPOINT_1320                    | -1572997038849006629
 SAVEPOINT JDBC_SAVEPOINT_26739                   |  54791337410474486
 SAVEPOINT JDBC_SAVEPOINT_1294466                 |  8170064357463507593
 ROLLBACK TO SAVEPOINT JDBC_SAVEPOINT_65016       | -33608214779996400
 SAVEPOINT JDBC_SAVEPOINT_14185                   | -2175035613806809562
 SAVEPOINT JDBC_SAVEPOINT_45837                   | -6201592986750645383
 SAVEPOINT JDBC_SAVEPOINT_1324                    |  6388797791882029332
```

PostgreSQL 17 inclut plusieurs améliorations pour le suivi des commandes utilitaires :

- Les noms des points de sauvegarde sont désormais affichés sous forme de constantes.
- Les identifiants de transaction globaux (GID) des commandes de validation en deux phases sont désormais affichés sous forme de constantes.

- Les noms des instructions DEALLOCATE sont affichés sous forme de constantes.
- Les paramètres CALL sont désormais affichés sous forme de constantes.

Timeout:PgSleep

L'événement `Timeout:PgSleep` se produit lorsqu'un processus serveur a appelé la fonction `pg_sleep` et attend l'expiration du délai de mise en veille.

Rubriques

Versions de moteur prises en charge

Ces informations sur les événements d'attente s'appliquent à toutes les versions d'Aurora PostgreSQL.

Causes probables de l'augmentation du nombre d'événements d'attente

Cet événement d'attente se produit lorsqu'une application, une fonction stockée ou un utilisateur émet une instruction SQL qui appelle l'une des fonctions suivantes :

- `pg_sleep`
- `pg_sleep_for`
- `pg_sleep_until`

Les fonctions précédentes retardent l'exécution jusqu'à ce que le nombre de secondes spécifié se soit écoulé. Par exemple, `SELECT pg_sleep(1)` marque une pause d'une seconde. Pour en savoir plus, consultez Delaying Execution dans la documentation PostgreSQL.

Actions

Identifiez l'instruction qui exécutait la fonction `pg_sleep`. Déterminez si l'utilisation de la fonction est appropriée.

Optimisation de grâce aux informations proactives d'Amazon Guru DevOps

DevOpsGuru Proactive Insights détecte les conditions sur vos (clusters de bases de données Aurora PostgreSQL) susceptibles de provoquer des problèmes, et vous en informe avant qu'ils ne surviennent. Des insights proactifs peuvent vous alerter lorsqu'une connexion reste inactive pendant une transaction de longue durée. Pour plus d'informations sur la résolution des problèmes liés aux connexions restées inactives trop longtemps pendant une transaction, consultez La base de données a une connexion de longue durée à l'état Transaction inactive

DevOpsGuru peut effectuer les opérations suivantes :

- Éviter de nombreux problèmes courants liés aux bases de données en recoupant la configuration de votre base de données par rapport aux paramètres courants recommandés.
- Vous alerter face à des problèmes critiques dans votre flotte qui, s'ils ne sont pas vérifiés, peuvent entraîner des problèmes plus importants ultérieurement.
- Vous alerter face à des problèmes nouvellement découverts.

Chaque insight proactif contient une analyse de la cause du problème et des recommandations d'actions correctives.

Pour plus d'informations sur Amazon DevOps Guru pour Amazon RDS, consultezAnalyse des anomalies de performance d'Aurora avec Amazon DevOps Guru pour Amazon RDS.

La base de données a une connexion de longue durée à l'état Transaction inactive

Une connexion à la base de données est à l'état `idle in transaction` depuis plus de 1 800 secondes.

Rubriques

- Versions de moteur prises en charge
- Contexte
- Causes probables de ce problème
- Actions

- Métriques pertinentes

Versions de moteur prises en charge

Ces données d'insight sont prises en charge pour toutes les versions d'Aurora PostgreSQL.

Contexte

Une transaction à l'état `idle in transaction` peut contenir des verrous qui bloquent d'autres requêtes. Elle peut également empêcher `VACUUM` (y compris autovacuum) de nettoyer les lignes inactives, ce qui entraînerait le gonflement des index ou des tables ou le bouclage des identifiants de transactions.

Causes probables de ce problème

Une transaction initiée dans une session interactive avec BEGIN ou START TRANSACTION ne s'est pas terminée à l'aide d'une commande COMMIT, ROLLBACK ou END. Cela entraîne le passage de la transaction à l'état `idle in transaction`.

Actions

Vous pouvez trouver les transactions inactives en exécutant la requête `pg_stat_activity`.

Dans votre client SQL, exécutez la requête suivante pour répertorier toutes les connexions à l'état `idle in transaction` et les ordonner par durée :

```
SELECT now() - state_change as idle_in_transaction_duration, now() - xact_start as
 xact_duration,*
FROM  pg_stat_activity
WHERE state  = 'idle in transaction'
AND   xact_start is not null
ORDER BY 1 DESC;
```

Nous vous recommandons différentes actions en fonction des causes de votre insight.

Rubriques

- Arrêt de la transaction
- Interruption de la connexion
- Configuration du paramètre idle_in_transaction_session_timeout

- Vérification du statut AUTOCOMMIT
- Vérification de la logique de transaction dans le code de votre application

Arrêt de la transaction

Lorsque vous lancez une transaction dans une session interactive avec BEGIN ou START TRANSACTION, elle passe à l'état `idle in transaction`. Elle reste dans cet état jusqu'à ce que vous terminiez la transaction en émettant une commande COMMIT, ROLLBACK ou END, ou que vous déconnectiez complètement la connexion pour annuler la transaction.

Interruption de la connexion

Mettez fin à la connexion avec une transaction inactive à l'aide de la requête suivante :

```
SELECT pg_terminate_backend(pid);
```

pid est l'ID de processus de la connexion.

Configuration du paramètre idle_in_transaction_session_timeout

Définissez le paramètre `idle_in_transaction_session_timeout` dans le nouveau groupe de paramètres. La configuration de ce paramètre présente l'avantage de ne pas nécessiter d'intervention manuelle pour mettre fin à la longue période d'inactivité de la transaction. Pour plus d'informations sur ce paramètre, consultez la documentation PostgreSQL.

Le message suivant sera enregistré dans le fichier journal de PostgreSQL après l'interruption de la connexion, quand une transaction sera dans l'état idle_in_transaction pendant un temps supérieur à la durée spécifiée.

```
FATAL: terminating connection due to idle in transaction timeout
```

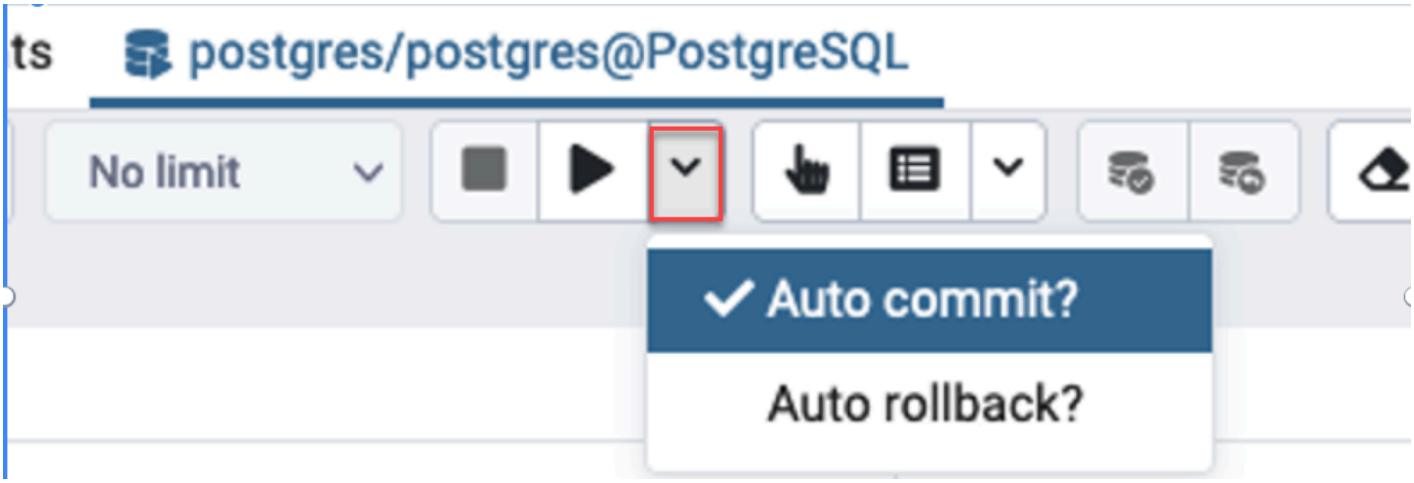
Vérification du statut AUTOCOMMIT

AUTOCOMMIT est activé par défaut. Mais s'il est désactivé accidentellement dans le client, veillez à le réactiver.

- Dans votre client psql, exécutez la commande suivante :

```
postgres=> \set AUTOCOMMIT on
```

- Dans pgadmin, activez-la en choisissant l'option AUTOCOMMIT à partir de la flèche déroulante.



Vérification de la logique de transaction dans le code de votre application

Examinez la logique de votre application pour détecter d'éventuels problèmes. Procédez comme suit :

- Vérifiez si la validation automatique JDBC est définie sur true dans votre application. Pensez également à utiliser des commandes `COMMIT` explicites dans votre code.
- Vérifiez votre logique de gestion des erreurs pour voir si elle clôture une transaction après des erreurs.
- Vérifiez si votre application met du temps à traiter les lignes renvoyées par une requête lorsque la transaction est ouverte. Si tel est le cas, pensez à coder l'application pour clôturer la transaction avant de traiter les lignes.
- Vérifiez si une transaction contient de nombreuses opérations de longue durée. Si tel est le cas, divisez une transaction individuelle en plusieurs transactions.

Métriques pertinentes

Les métriques PI suivantes sont liées à cet insight :

- idle_in_transaction_count : nombre de sessions à l'état `idle in transaction`.
- idle_in_transaction_max_time : durée de la transaction la plus longue à l'état `idle in transaction`.

Bonnes pratiques avec Amazon Aurora PostgreSQL

Vous trouverez ci-dessous plusieurs bonnes pratiques pour gérer votre cluster de bases de données Amazon Aurora PostgreSQL. Veillez également à passer en revue les tâches d'entretien de base. Pour plus d'informations, consultez Performance et mise à l'échelle d'Amazon Aurora PostgreSQL.

Rubriques

Contournement des performances lentes, du redémarrage automatique et du basculement pour les instances de base de données Aurora PostgreSQL

Si vous exécutez une charge de travail importante ou des charges de travail qui dépassent les ressources allouées à votre instance de base de données, vous pouvez épuiser les ressources sur lesquelles vous exécutez votre application et votre base de données Aurora. Pour obtenir des mesures sur votre instance de base de données, telles que l'utilisation du processeur, l'utilisation de la mémoire et le nombre de connexions de base de données utilisées, vous pouvez vous référer aux mesures fournies par Amazon CloudWatch, Performance Insights et Enhanced Monitoring. Pour plus d'informations sur la surveillance de votre instance de base de données, consultez Surveillance des métriques d'un cluster de bases de données Amazon Aurora.

Si votre charge de travail épuise les ressources que vous utilisez, votre instance de base de données peut ralentir, redémarrer ou même basculer vers une autre instance de base de données. Pour éviter cela, surveillez l'utilisation de vos ressources, examinez la charge de travail exécutée sur votre instance de base de données et effectuez des optimisations si nécessaire. Si les optimisations n'améliorent pas les métriques de l'instance et n'atténuent pas l'épuisement des ressources, envisagez d'augmenter votre instance de base de données avant d'atteindre ses limites. Pour plus d'informations sur les classes d'instance de base de données disponibles et leurs spécifications, consultez Classes d'instance de base de données Amazon Aurora.

Diagnostic du gonflement de la table et de l'index

Vous pouvez utiliser le contrôle de simultanéité multiversion (MVCC) PostgreSQL pour préserver l'intégrité des données. PostgreSQL MVCC fonctionne en enregistrant une copie interne des lignes mises à jour ou supprimées (également appelées tuples) jusqu'à ce qu'une transaction soit validée ou annulée. Cette copie interne enregistrée est invisible pour les utilisateurs. Toutefois, le gonflement de la table peut se produire lorsque ces copies invisibles ne sont pas nettoyées régulièrement par les utilitaires VACUUM ou AUTOVACUUM. S'il n'est pas vérifié, le gonflement de la table peut entraîner une augmentation des coûts de stockage et ralentir votre vitesse de traitement.

Dans de nombreux cas, les paramètres par défaut pour VACUUM ou AUTOVACUUM sur Aurora sont suffisants pour gérer les gonflements de table indésirables. Toutefois, vous pouvez vérifier qu'il n'y a pas de gonflement si votre application présente les conditions suivantes :

- Traite un grand nombre de transactions en un temps relativement court entre les processus VACUUM.
- Offre des performances médiocres et manque d'espace de stockage.

Pour commencer, collectez les informations les plus précises sur l'espace utilisé par les tuples morts et sur la quantité que vous pouvez récupérer en nettoyant le gonflement de la table et de l'index. Pour ce faire, utilisez l'extension `pgstattuple` pour recueillir des statistiques sur votre cluster Aurora. Pour plus d'informations, consultez pgstattuple. Les privilèges d'utilisation de l'extension `pgstattuple` sont limités au rôle `pg_stat_scan_tables` et aux super-utilisateurs de la base de données.

Pour créer l'extension `pgstattuple` sur Aurora, connectez une session client au cluster, par exemple psql ou pgAdmin, et utilisez la commande suivante :

```
CREATE EXTENSION pgstattuple;
```

Créez l'extension dans chaque base de données que vous souhaitez profiler. Après avoir créé l'extension, utilisez l'interface de ligne de commande (CLI) pour mesurer la quantité d'espace inutilisable que vous pouvez récupérer. Avant de recueillir des statistiques, modifiez le groupe de paramètres du cluster en définissant AUTOVACUUM sur 0. Un paramètre de 0 empêche Aurora de nettoyer automatiquement les tuples morts laissés par votre application, ce qui peut avoir une incidence sur la précision des résultats. Saisissez la commande suivante pour créer une table simple :

```
postgres=> CREATE TABLE lab AS SELECT generate_series (0,100000);
SELECT 100001
```

Dans l'exemple suivant, nous exécutons la requête avec AUTOVACUUM activé pour le cluster de bases de données. Le paramètre `dead_tuple_count` est 0, ce qui indique que AUTOVACUUM a supprimé les données ou les tuples obsolètes de la base de données PostgreSQL.

Pour utiliser `pgstattuple` afin de recueillir des informations sur la table, spécifiez le nom d'une table ou un identifiant d'objet (OID) dans la requête :

```
postgres=> SELECT * FROM pgstattuple('lab');
```

```
table_len  | tuple_count | tuple_len | tuple_percent | dead_tuple_count |
 dead_tuple_len | dead_tuple_percent | free_space | free_percent
-----------+-------------+-----------+---------------+------------------
+----------------+--------------------+------------+--------------
3629056    | 100001      | 2800028   | 77.16         | 0                | 0
  | 0                  | 16616      | 0.46
(1 row)
```

Dans la requête suivante, nous désactivons AUTOVACUUM et saisissons une commande qui supprime 25 000 lignes de la table. En conséquence, `dead_tuple_count` passe à 25 000.

```
postgres=> DELETE FROM lab WHERE generate_series < 25000;

DELETE 25000
```

```
SELECT * FROM pgstattuple('lab');
```

```
table_len | tuple_count | tuple_len | tuple_percent | dead_tuple_count | dead_tuple_len
 | dead_tuple_percent | free_space | free_percent
-----------+-------------+-----------+---------------+------------------
+----------------+--------------------+------------+--------------
3629056 | 75001 | 2100028 | 57.87 | 25000 | 700000 | 19.29 | 16616 | 0.46
(1 row)
```

Pour récupérer ces tuples morts, lancez un processus VACUUM.

Observation des gonflements sans interrompre votre application

Les paramètres d'un cluster Aurora sont optimisés pour fournir les bonnes pratiques pour la plupart des charges de travail. Toutefois, vous souhaiterez peut-être optimiser un cluster pour mieux l'adapter à vos applications et à vos modèles d'utilisation. Dans ce cas, vous pouvez utiliser l'extension `pgstattuple` sans perturber une application active. Pour ce faire, effectuez les étapes suivantes :

1. Clonez votre instance Aurora.
2. Modifiez le fichier de paramètres pour désactiver AUTOVACUUM dans le clone.
3. Exécutez une requête `pgstattuple` tout en testant le clone avec un exemple de charge de travail ou avec pgbench, un programme permettant d'exécuter des tests de référence sur PostgreSQL. Pour plus d'informations, consultez pgbench.

Après avoir lancé vos applications et consulté le résultat, utilisez pg_repack ou VACUUM FULL sur la copie restaurée et comparez les différences. Si vous constatez une baisse significative de dead_tuple_count, dead_tuple_len ou dead_tuple_percent, ajustez le calendrier de mise à vide de votre cluster de production afin de minimiser le gonflement.

Prévention du gonflement dans les tables temporaires

Si votre application crée des tables temporaires, assurez-vous qu'elle supprime ces tables temporaires lorsqu'elles ne sont plus nécessaires. Les processus de mise à vide automatique ne localisent pas les tables temporaires. Si elles ne sont pas vérifiées, les tables temporaires peuvent rapidement entraîner un gonflement de la base de données. De plus, le gonflement peut s'étendre aux tables système, qui sont les tables internes qui suivent les objets et les attributs de PostgreSQL, tels que pg_attribute et pg_depend.

Lorsqu'une table temporaire n'est plus nécessaire, vous pouvez utiliser une instruction TRUNCATE pour vider la table et libérer de l'espace. Ensuite, videz manuellement les tables pg_attribute et pg_depend. La mise à vide de ces tables garantit que la création et la troncation/la suppression continue de tables temporaires n'ajoute pas de tuples et ne contribue pas au gonflement du système.

Vous pouvez éviter ce problème lors de la création d'une table temporaire en incluant la syntaxe suivante qui supprime les nouvelles lignes lorsque le contenu est validé :

```
CREATE TEMP TABLE IF NOT EXISTS table_name(table_description) ON COMMIT DELETE ROWS;
```

La clause `ON COMMIT DELETE ROWS` tronque la table temporaire lorsque la transaction est validée.

Prévention du gonflement dans les index

Lorsque vous modifiez un champ indexé dans une table, la mise à jour de l'index entraîne la suppression d'un ou de plusieurs tuples dans cet index. Par défaut, le processus de mise à vide automatique élimine le gonflement des index, mais ce nettoyage nécessite beaucoup de temps et de ressources. Pour spécifier les préférences de nettoyage d'index lorsque vous créez une table, incluez la clause vacuum_index_cleanup. Par défaut, au moment de la création de la table, la clause est définie sur AUTO, ce qui signifie que le serveur décide si votre index doit être nettoyé lorsqu'il vide la table. Vous pouvez définir la clause sur ON pour activer le nettoyage de l'index pour une table spécifique, ou sur OFF pour désactiver le nettoyage de l'index pour cette table. N'oubliez pas que la désactivation du nettoyage des index peut vous faire gagner du temps, mais peut entraîner un gonflement de l'index.

Vous pouvez contrôler manuellement le nettoyage de l'index lorsque vous VACUUM une table dans la ligne de commande. Pour vider une table et supprimer les tuples morts des index, incluez la clause INDEX_CLEANUP avec la valeur ON et le nom de la table :

```
acctg=> VACUUM (INDEX_CLEANUP ON) receivables;
```

```
INFO: aggressively vacuuming "public.receivables"
VACUUM
```

Pour vider une table sans nettoyer les index, spécifiez la valeur OFF :

```
acctg=> VACUUM (INDEX_CLEANUP OFF) receivables;

INFO: aggressively vacuuming "public.receivables"
VACUUM
```

Gestion de mémoire améliorée dans Aurora PostgreSQL

Aurora PostgreSQL inclut désormais des fonctionnalités avancées de gestion de la mémoire afin d'optimiser les performances et la résilience des bases de données sous différentes charges de travail. Ces améliorations aident Aurora PostgreSQL à assurer une disponibilité et une réactivité constantes, même pendant les périodes de forte demande de mémoire.

Cette fonctionnalité est disponible et activée par défaut dans les versions Aurora PostgreSQL suivantes pour les instances provisionnées :

- 15.3 et versions mineures ultérieures
- 14.8 et versions mineures ultérieures
- 13.11 et versions mineures ultérieures
- 12.15 et versions mineures ultérieures
- 11.20 et versions mineures ultérieures

Cette fonctionnalité est disponible et activée par défaut dans les versions Aurora PostgreSQL suivantes pour les instances Aurora Serverless :

- 16.3 et versions mineures ultérieures
- 15.7 et versions mineures ultérieures
- 14.12 et versions mineures ultérieures
- 13.15 et versions mineures ultérieures

Lorsque les charges de travail des clients utilisent toute la mémoire disponible, le système d'exploitation peut redémarrer la base de données pour protéger les ressources, ce qui entraîne une indisponibilité temporaire. Les nouvelles améliorations apportées à la gestion de la mémoire

dans Aurora PostgreSQL annulent de manière proactive certaines transactions lorsque le système est soumis à une charge de mémoire élevée, ce qui contribue à maintenir la stabilité de la base de données.

Voici les principales caractéristiques liées à l'amélioration de la gestion de la mémoire :

- Annulation des transactions de base de données qui demandent plus de mémoire quand le système approche d'une sollicitation critique de la mémoire.
- Le système est considéré soumis à une sollicitation critique de la mémoire lorsqu'il épuise toute la mémoire physique et qu'il est sur le point d'épuiser l'échange. Dans ces circonstances, toute transaction demandant de la mémoire sera annulée afin de réduire immédiatement la sollicitation de la mémoire dans l'instance de base de données.
- Les lanceurs PostgreSQL essentiels et les exécutants en arrière-plan tels que les threads de travail Autovacuum sont toujours protégés.

Traitement des paramètres de gestion de la mémoire

Pour activer la gestion de la mémoire

Cette fonction est désactivée par défaut. Un message d'erreur s'affiche quand une transaction est annulée en raison d'une mémoire insuffisante, comme illustré dans l'exemple suivant :

```
ERROR: out of memory Detail: Failed on request of size 16777216.
```

Pour désactiver la gestion de la mémoire

Pour désactiver cette fonctionnalité, connectez-vous au cluster de bases de données Aurora PostgreSQL avec psql et utilisez l'instruction SET pour les valeurs des paramètres, comme indiqué ci-dessous.

> **Note**
>
> Nous vous recommandons de laisser la gestion de la mémoire activée. Cela permet d'éviter d'éventuelles erreurs de mémoire qui pourraient entraîner le redémarrage de la base de données en raison d'une saturation de la mémoire due à la charge de travail.

Le tableau suivant indique comment désactiver la fonctionnalité de gestion de la mémoire pour les différentes versions d'Aurora PostgreSQL :

Versions d'Aurora PostgreSQL	Paramètre	Par défaut	Commande pour désactiver la gestion de la mémoire au niveau de la session
11.20, 11.21, 12.15, 12.16, 13.11, 13.12, 14.8, 14.9, 15.3, 15.4	`rds.memory_allocation_guard`	`false`	`SET rds.memory_allocation_guard = true;`
12.17, 13.13, 14.10, 15.5 et versions ultérieures	`rds.enable_memory_management`	`true`	`SET rds.enable_memory_management = false;`

> **Note**
>
> Le paramètre `rds.memory_allocation_guard` est devenu obsolète dans Aurora PostgreSQL 12.17, 13.13, 14.10, 15.5 et versions ultérieures.

La définition des valeurs de ces paramètres dans le groupe de paramètres du cluster de bases de données empêche l'annulation des requêtes. Pour plus d'informations sur le groupe de paramètres du cluster de bases de données, consultez Groupes de paramètres pour Amazon Aurora.

Limitation

- Cette fonctionnalité n'est pas prise en charge dans la classe d'instance db.t3.medium.

Basculement rapide avec Amazon Aurora PostgreSQL

Vous apprendrez ensuite comment faire en sorte que le basculement se produise aussi rapidement que possible. Pour récupérer rapidement après un basculement, vous pouvez utiliser la gestion de cache en cluster pour votre cluster de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez Récupération rapide après basculement avec la gestion des caches de clusters pour Aurora PostgreSQL.

Voici quelques-unes des mesures que vous pouvez prendre pour que le basculement soit rapide :

- Définissez des keepalives TCP (Transmission Control Protocol) de courte durée, afin d'arrêter les requêtes plus longues avant l'expiration du délai de lecture en cas d'échec.
- Définissez les délais de mise en cache du système de nom de domaine (DNS) de Java de manière agressive. Cela permet de s'assurer que le point de terminaison en lecture seule d'Aurora peut correctement passer en revue les nœuds en lecture seule lors des tentatives de connexion ultérieures.
- Définition des variables d'expiration utilisées dans la chaîne de connexion JDBC à des valeurs aussi faibles que possible. Utilisation d'objets de connexion distincts pour les requêtes à exécution courte et longue.
- Utilisez les points de terminaison Aurora en lecture et en écriture qui sont fournis pour vous connecter au cluster.
- Utilisez les opérations de l'API RDS pour tester la réponse de l'application en cas de défaillance du serveur. Utilisez également un outil de suppression de paquets pour tester la réponse de l'application en cas de défaillance côté client.
- Utilisez le pilote JDBC AWS pour tirer pleinement parti des fonctionnalités de basculement d'Aurora PostgreSQL. Pour en savoir plus sur le pilote JDBC AWS et pour obtenir des instructions d'utilisation complètes, consultez le référentiel GitHub du pilote JDBC Amazon Web Services (AWS).

Ces opérations sont traitées plus en détail ci-après.

Rubriques

Définition des paramètres TCP keepalive

Lorsque vous établissez une connexion TCP, un ensemble de temporisateurs est employé avec la connexion. Lorsque le temporisateur keepalive atteint zéro, un paquet de test keepalive est envoyé au point de terminaison de la connexion. Si le test reçoit une réponse, vous pouvez supposer que la connexion est toujours en cours.

L'activation des paramètres TCP keepalive et leur réglage agressif garantissent que si votre client ne peut pas se connecter à la base de données, toute connexion active sera rapidement fermée. L'application peut alors se connecter à un nouveau point de terminaison.

Assurez-vous de définir les paramètres TCP keepalive suivants :

- `tcp_keepalives_idle` contrôle le délai en secondes au bout duquel un paquet keepalive est envoyé lorsqu'aucune donnée n'a été envoyée par le socket. Les ACK ne sont pas considérés comme des données. Nous vous recommandons de définir les paramètres suivants :

  ```
  tcp_keepalives_idle = 1
  ```

- `tcp_keepalives_interval` contrôle l'intervalle en secondes entre l'envoi des paquets keepalive suivants après l'envoi du paquet initial. Réglez cette heure à l'aide du paramètre `tcp_keepalives_idle`. Nous vous recommandons de définir les paramètres suivants :

  ```
  tcp_keepalives_interval = 1
  ```

- `tcp_keepalives_count` est le nombre de tests keepalive non reconnus qui ont lieu avant que l'application ne soit informée. Nous vous recommandons de définir les paramètres suivants :

  ```
  tcp_keepalives_count = 5
  ```

Ces paramètres doivent informer l'application dans les cinq secondes après que la base de données a cessé de répondre. Si les paquets keepalive sont souvent abandonnés dans le réseau de l'application, vous pouvez définir une valeur `tcp_keepalives_count` plus élevée. Cela permet d'augmenter la mémoire tampon dans les réseaux moins fiables, mais augmente le temps nécessaire à la détection d'une panne réelle.

Pour définir des paramètres TCP keepalive sous Linux

1. Testez comment configurer vos paramètres TCP keepalive.

   Nous vous recommandons de le faire en utilisant la ligne de commande avec les commandes suivantes. Cette configuration suggérée est valable pour l'ensemble du système. En d'autres termes, cela affecte également toutes les autres applications qui créent des sockets avec l'option `SO_KEEPALIVE` activée.

   ```
   sudo sysctl net.ipv4.tcp_keepalive_time=1
   sudo sysctl net.ipv4.tcp_keepalive_intvl=1
   ```

```
sudo sysctl net.ipv4.tcp_keepalive_probes=5
```

2. Lorsque vous avez trouvé une configuration qui fonctionne pour votre application, rendez persistants les paramètres suivants en ajoutant les lignes suivantes à `/etc/sysctl.conf`, avec les modifications que vous avez effectuées :

```
tcp_keepalive_time = 1
tcp_keepalive_intvl = 1
tcp_keepalive_probes = 5
```

Configuration de votre application pour le basculement rapide

Vous trouverez ci-dessous une discussion sur plusieurs changements de configuration d'Aurora PostgreSQL que vous pouvez mettre en œuvre pour un basculement rapide. Pour en savoir plus sur l'installation et la configuration du pilote JDBC PostgreSQL, consultez la documentation Pilote JDBC PostgreSQL.

Rubriques

- Réduction des délais d'expiration du cache du DNS
- Définition d'une chaîne de connexion Aurora PostgreSQL pour le basculement rapide
- Autres options pour obtenir la chaîne hôte

Réduction des délais d'expiration du cache du DNS

Lorsque votre application tente d'établir une connexion après un basculement, la nouvelle instance Aurora PostgreSQL en écriture sera une ancienne instance en lecture. Vous pouvez la trouver en utilisant le point de terminaison Aurora en lecture seule avant que les mises à jour DNS ne se soient entièrement propagées. En fixant la durée de vie (TTL) du DNS java à une valeur faible, inférieure à 30 secondes par exemple, on facilite le passage d'un nœud de lecteur à l'autre lors des tentatives de connexion ultérieures.

```
// Sets internal TTL to match the Aurora RO Endpoint TTL
java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
// If the lookup fails, default to something like small to retry
java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
```

Définition d'une chaîne de connexion Aurora PostgreSQL pour le basculement rapide

Pour utiliser le basculement rapide d'Aurora PostgreSQL, assurez-vous que la chaîne de connexion de votre application comporte une liste d'hôtes au lieu d'un seul. Voici un exemple de chaîne de connexion que vous pouvez utiliser pour vous connecter à un cluster Aurora PostgreSQL. Dans cet exemple, les hôtes sont en gras.

```
jdbc:postgresql://myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
/postgres?user=<primaryuser>&password=<primarypw>&loginTimeout=2
&connectTimeout=2&cancelSignalTimeout=2&socketTimeout=60
&tcpKeepAlive=true&targetServerType=primary
```

Pour une meilleure disponibilité et pour éviter une dépendance à l'API RDS, nous vous recommandons de garder un fichier pour vous connecter. Ce fichier contient une chaîne d'hôte que votre application lit lorsque vous établissez une connexion avec la base de données. Cette chaîne hôte possède tous les points de terminaison Aurora disponibles pour le cluster. Pour plus d'informations sur les points de terminaison Aurora, consultez Connexions de point de terminaison Amazon Aurora.

Par exemple, vous pouvez stocker vos points de terminaison dans un fichier local comme indiqué ci-dessous.

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

Votre application lit ce fichier pour renseigner la section hôte de la chaîne de connexion JDBC. Le changement de nom du cluster de bases de données entraîne la modification de ces points de terminaison. Assurez-vous que votre application gère cet événement s'il se produit.

Vous pouvez également utiliser une liste de nœuds d'instance de base de données, comme suit.

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node3.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node4.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

L'avantage de cette approche est que le pilote de connexion JDBC PostgreSQL boucle sur tous les nœuds de cette liste pour trouver une connexion valide. En revanche, lorsque vous utilisez les points

de terminaison Aurora, seuls deux nœuds sont essayés à chaque tentative de connexion. L'utilisation des nœuds d'instance de base de données présente toutefois un inconvénient. Si vous ajoutez ou supprimez des nœuds dans votre cluster et que la liste des points de terminaison devient obsolète, le pilote de connexion ne trouvera peut-être jamais l'hôte correct auquel il doit se connecter.

Pour faire en sorte que votre application n'attende pas trop longtemps pour se connecter à un hôte donné, définissez les paramètres suivants de manière agressive :

- `targetServerType` : contrôle si le pilote se connecte à un nœud en écriture ou en lecture. Pour vous assurer que vos applications se reconnectent uniquement à un nœud d'écriture, définissez la valeur de `targetServerType` sur `primary`.

  Les valeurs pour le paramètre `targetServerType` incluent `primary`, `secondary`, `any` et `preferSecondary`. La valeur `preferSecondary` tente d'abord d'établir une connexion avec une instance en lecture. Elle se connecte à l'instance en écriture si aucune connexion à l'instance en lecture ne peut être établie.
- `loginTimeout` : contrôle la durée d'attente de votre application pour se connecter à la base de données après qu'une connexion socket a été établie.
- `connectTimeout` – contrôle la durée d'attente du socket pour établir une connexion à la base de données.

Vous pouvez modifier d'autres paramètres d'application pour accélérer le processus de connexion, selon le degré d'énergie que vous voulez attribuer à votre application :

- `cancelSignalTimeout` : dans certaines applications, vous voudrez peut-être envoyer un signal d'annulation le meilleur possible pour une requête qui a expiré. Si ce signal d'annulation se trouve dans votre chemin de basculement, pensez à le définir de manière énergique pour éviter de l'envoyer à un hôte mort.
- `socketTimeout` – Ce paramètre contrôle la durée d'attente du socket pour les opérations de lecture. Ce paramètre peut servir de délai de requête global afin d'assurer que la durée d'attente des requêtes ne dépasse jamais sa valeur. Une bonne pratique consiste à avoir deux gestionnaires de connexion. Un gestionnaire de connexion exécute des requêtes à courte durée de vie et fixe cette valeur plus bas. Dans un autre gestionnaire de connexion, pour les requêtes de longue durée, cette valeur est beaucoup plus élevée. Avec cette approche, vous pouvez compter sur les paramètres TCP keepalive pour arrêter les requêtes de longue durée si le serveur tombe en panne.
- `tcpKeepAlive` : activez ce paramètre pour garantir que les paramètres TCP keepalive que vous définissez sont respectés.

- `loadBalanceHosts` – Lorsque ce paramètre est défini sur `true`, l'application se connecte à un hôte aléatoire choisi dans une liste d'hôtes candidats.

Autres options pour obtenir la chaîne hôte

Vous pouvez obtenir la chaîne hôte à partir de plusieurs sources, notamment de la fonction `aurora_replica_status` et en utilisant l'API Amazon RDS.

Dans de nombreux cas, vous devez déterminer quelle est l'instance en écriture du cluster ou trouver d'autres nœuds en lecture dans le cluster. Pour ce faire, votre application peut se connecter à n'importe quelle instance de base de données dans le cluster de bases de données et interroger la fonction `aurora_replica_status`. Vous pouvez utiliser cette fonction pour réduire le temps nécessaire à la recherche d'un hôte auquel se connecter. Toutefois, dans certains scénarios de réseaux, la fonction `aurora_replica_status` peut afficher des informations obsolètes ou incomplètes.

Une bonne manière de s'assurer que votre application peut trouver un nœud auquel se connecter est d'essayer de se connecter au point de terminaison de l'instance en écriture du cluster, puis au point de terminaison de l'instance en lecture du cluster. Vous faites cela jusqu'à ce que vous puissiez établir une connexion lisible. Ces points de terminaison ne changent pas, sauf si vous renommez votre cluster de bases de données. Ainsi, vous pouvez généralement les laisser en tant que membres statiques de votre application ou les stocker dans un fichier de ressources que votre application lit.

Une fois que vous avez établi une connexion à l'aide de l'un de ces points de terminaison, vous pouvez obtenir des informations sur le reste du cluster. Pour ce faire, appelez la fonction `aurora_replica_status`. Par exemple, la commande suivante récupère des informations avec `aurora_replica_status`.

```
postgres=> SELECT server_id, session_id, highest_lsn_rcvd, cur_replay_latency_in_usec,
 now(), last_update_timestamp
FROM aurora_replica_status();

server_id | session_id | highest_lsn_rcvd | cur_replay_latency_in_usec | now |
 last_update_timestamp
-----------+--------------------------------------+-----------------
+---------------------------+-------------------------------+-----------------------
mynode-1 | 3e3c5044-02e2-11e7-b70d-95172646d6ca | 594221001 | 201421 | 2017-03-07
 19:50:24.695322+00 | 2017-03-07 19:50:23+00
mynode-2 | 1efdd188-02e4-11e7-becd-f12d7c88a28a | 594221001 | 201350 | 2017-03-07
 19:50:24.695322+00 | 2017-03-07 19:50:23+00
```

```
mynode-3 | MASTER_SESSION_ID | | | 2017-03-07 19:50:24.695322+00 | 2017-03-07
 19:50:23+00
(3 rows)
```

Par exemple, la section hosts (hôtes) de votre chaîne de connexion peut commencer par les points de terminaison des clusters de l'instance en écriture et en lecture, comme indiqué ci-dessous.

```
myauroracluster.cluster-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432,
myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-beta.rds.amazonaws.com:5432
```

Dans ce scénario, votre application tente d'établir une connexion à un type de nœud, principal ou secondaire. Dès que votre application est connectée, nous vous conseillons de commencer par examiner le statut en lecture-écriture du nœud. Pour ce faire, recherchez le résultat de la commande `SHOW transaction_read_only`.

Si la valeur de retour de la requête est `OFF`, vous êtes bien connecté au nœud principal. Toutefois, supposons que la valeur de retour soit `ON` et que votre application nécessite une connexion en lecture/écriture. Dans ce cas, vous pouvez appeler la fonction `aurora_replica_status` pour déterminer le `server_id` qui possède `session_id='MASTER_SESSION_ID'`. Cette fonction vous donne le nom du nœud principal. Vous pouvez l'utiliser avec la fonction `endpointPostfix` décrite ci-après.

Assurez-vous de savoir ce que vous faites lorsque vous vous connectez à un réplica disposant de données obsolètes. Dans ce cas, la fonction `aurora_replica_status` peut présenter des informations qui ne sont pas à jour. Vous pouvez définir un seuil d'instabilité au niveau de l'application. Pour le vérifier, vous pouvez regarder la différence entre l'heure du serveur et la valeur `last_update_timestamp`. En général, votre application doit éviter de basculer entre deux hôtes en raison de conflits d'informations renvoyées par la fonction `aurora_replica_status`. Votre application devrait d'abord essayer tous les hôtes connus au lieu de suivre les données renvoyées par `aurora_replica_status`.

Liste des instances utilisant l'opération de l'API DescribeDBClusters, exemple en Java

Vous pouvez rechercher par programmation la liste des instances en utilisant le AWS SDK pour Java, et plus précisément l'opération d'API DescribeDBClusters.

Voici un petit exemple de la façon dont vous pouvez procéder en Java 8.

```
AmazonRDS client = AmazonRDSClientBuilder.defaultClient();
DescribeDBClustersRequest request = new DescribeDBClustersRequest()
```

```
   .withDBClusterIdentifier(clusterName);
DescribeDBClustersResult result =
rdsClient.describeDBClusters(request);

DBCluster singleClusterResult = result.getDBClusters().get(0);

String pgJDBCEndpointStr =
singleClusterResult.getDBClusterMembers().stream()
   .sorted(Comparator.comparing(DBClusterMember::getIsClusterWriter)
   .reversed()) // This puts the writer at the front of the list
   .map(m -> m.getDBInstanceIdentifier() + endpointPostfix + ":" +
 singleClusterResult.getPort()))
   .collect(Collectors.joining(","));
```

Ici, `pgJDBCEndpointStr` contient une liste formatée de points de terminaison, comme indiqué ci-dessous.

```
my-node1.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432,
my-node2.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com:5432
```

La variable `endpointPostfix` peut être une constante que votre application définit. Votre application peut aussi l'obtenir en interrogeant l'opération API `DescribeDBInstances` pour une seule instance de votre cluster. Cette valeur reste constante au sein d'une Région AWS et pour un client individuel. Cela permet donc d'économiser un appel d'API pour simplement conserver cette constante dans un fichier de ressources que votre application lit. Dans l'exemple précédent, elle est définie comme suit.

```
.cksc6xlmwcyw.us-east-1-beta.rds.amazonaws.com
```

Pour une meilleure disponibilité, nous vous conseillons d'utiliser par défaut les points de terminaison Aurora de votre cluster de bases de données si l'API ne répond pas ou est trop longue à répondre. Les points de terminaison sont toujours mis à jour dans le délai requis pour mettre à jour l'enregistrement DNS. La mise à jour de l'enregistrement DNS avec un point de terminaison prend généralement moins de 30 secondes. Vous pouvez stocker le point de terminaison dans un fichier de ressources que votre application consomme.

Test du basculement

Dans tous les cas, vous devez avoir un cluster de bases de données avec deux ou plusieurs instances de base de données.

Du côté serveur, certaines opérations d'API peuvent provoquer une panne qui peut servir à tester la manière dont votre application répond :

- FailoverDBCluster : cette opération tente de promouvoir une nouvelle instance de base de données dans votre cluster de bases de données en tant qu'instance en écriture.

  L'exemple de code suivant montre comment vous pouvez utiliser `failoverDBCluster` pour provoquer une panne. Pour plus d'informations sur la configuration d'un client Amazon RDS, consultez Utilisation du kit SDK AWS pour Java.

```
public void causeFailover() {

    final AmazonRDS rdsClient = AmazonRDSClientBuilder.defaultClient();

    FailoverDBClusterRequest request = new FailoverDBClusterRequest();
    request.setDBClusterIdentifier("cluster-identifier");

    rdsClient.failoverDBCluster(request);
}
```

- RebootDBInstance : le basculement n'est pas garanti avec cette opération d'API. Cependant, il ferme la base de données sur l'instance en écriture. Vous pouvez l'utiliser pour tester la façon dont votre application réagit à l'abandon des connexions. Ce paramètre `ForceFailover` ne s'applique pas aux moteurs Aurora. Utilisez plutôt l'opération d'API `FailoverDBCluster`.
- ModifyDBCluster : la modification du paramètre `Port` entraîne une panne lorsque les nœuds du cluster commencent à écouter sur un nouveau port. En général, votre application peut répondre à cette défaillance en s'assurant que seule votre application contrôle les changements de port. Assurez-vous également qu'elle peut mettre à jour de manière appropriée les points de terminaison dont elle dépend. Vous pouvez le faire en demandant à quelqu'un de mettre à jour manuellement le port lorsqu'il apporte des modifications au niveau de l'API. Vous pouvez également le faire en utilisant l'API RDS dans votre application pour déterminer si le port a changé.
- ModifyDBInstance : la modification du paramètre `DBInstanceClass` provoque une panne.
- DeleteDBInstance : la suppression de l'instance principale (écriture) entraîne la promotion d'une nouvelle instance de base de données en tant qu'instance en écriture dans votre cluster de bases de données.

Du côté de l'application ou du client, si vous utilisez Linux, vous pouvez tester la façon dont l'application réagit aux rejets soudains de paquets. Vous pouvez le faire en fonction du port, de l'hôte ou si des paquets TCP keepalive sont envoyés ou reçus en utilisant la commande iptables.

Exemple de basculement rapide en Java

L'exemple de code suivant montre comment une application peut configurer un gestionnaire de pilotes Aurora PostgreSQL.

L'application appelle la fonction `getConnection`lorsqu'elle a besoin d'une connexion. Un appel à `getConnection` peut ne pas parvenir à trouver un hôte valide. Par exemple, si aucune instance en écriture n'est trouvée mais que le paramètre `targetServerType` est défini sur `primary`. Dans ce cas, l'application appelante doit simplement réessayer d'appeler la fonction.

Pour éviter d'imposer le comportement de relance à l'application, vous pouvez envelopper cet appel de relance dans une fonction de regroupement de connexions. Avec la plupart des fonctions de regroupement de connexions, vous pouvez spécifier une chaîne de connexion JDBC. Votre application peut donc appeler `getJdbcConnectionString` et la transmettre à la fonction de regroupement de connexions. Cela signifie que vous pouvez utiliser un basculement plus rapide avec Aurora PostgreSQL.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

import org.joda.time.Duration;

public class FastFailoverDriverManager {
   private static Duration LOGIN_TIMEOUT = Duration.standardSeconds(2);
   private static Duration CONNECT_TIMEOUT = Duration.standardSeconds(2);
   private static Duration CANCEL_SIGNAL_TIMEOUT = Duration.standardSeconds(1);
   private static Duration DEFAULT_SOCKET_TIMEOUT = Duration.standardSeconds(5);

   public FastFailoverDriverManager() {
       try {
            Class.forName("org.postgresql.Driver");
       } catch (ClassNotFoundException e) {
```

```
            e.printStackTrace();
      }

      /*
        * RO endpoint has a TTL of 1s, we should honor that here. Setting this
aggressively makes sure that when
        * the PG JDBC driver creates a new connection, it will resolve a new different
RO endpoint on subsequent attempts
        * (assuming there is > 1 read node in your cluster)
        */
       java.security.Security.setProperty("networkaddress.cache.ttl" , "1");
      // If the lookup fails, default to something like small to retry
      java.security.Security.setProperty("networkaddress.cache.negative.ttl" , "3");
  }

  public Connection getConnection(String targetServerType) throws SQLException {
      return getConnection(targetServerType, DEFAULT_SOCKET_TIMEOUT);
  }

  public Connection getConnection(String targetServerType, Duration queryTimeout)
throws SQLException {
       Connection conn =
DriverManager.getConnection(getJdbcConnectionString(targetServerType, queryTimeout));

      /*
        * A good practice is to set socket and statement timeout to be the same thing
since both
        * the client AND server will stop the query at the same time, leaving no
running queries
        * on the backend
        */
       Statement st = conn.createStatement();
       st.execute("set statement_timeout to " + queryTimeout.getMillis());
       st.close();

      return conn;
  }

  private static String urlFormat = "jdbc:postgresql://%s"
          + "/postgres"
          + "?user=%s"
          + "&password=%s"
          + "&loginTimeout=%d"
          + "&connectTimeout=%d"
```

```
            + "&cancelSignalTimeout=%d"
            + "&socketTimeout=%d"
            + "&targetServerType=%s"
            + "&tcpKeepAlive=true"
            + "&ssl=true"
            + "&loadBalanceHosts=true";
    public String getJdbcConnectionString(String targetServerType, Duration
 queryTimeout) {
        return String.format(urlFormat,
                getFormattedEndpointList(getLocalEndpointList()),
                CredentialManager.getUsername(),
                CredentialManager.getPassword(),
                LOGIN_TIMEOUT.getStandardSeconds(),
                CONNECT_TIMEOUT.getStandardSeconds(),
                CANCEL_SIGNAL_TIMEOUT.getStandardSeconds(),
                queryTimeout.getStandardSeconds(),
                targetServerType
        );
    }

    private List<String> getLocalEndpointList() {
        /*
          * As mentioned in the best practices doc, a good idea is to read a local
 resource file and parse the cluster endpoints.
          * For illustration purposes, the endpoint list is hardcoded here
          */
         List<String> newEndpointList = new ArrayList<>();
         newEndpointList.add("myauroracluster.cluster-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");
         newEndpointList.add("myauroracluster.cluster-ro-c9bfei4hjlrd.us-east-1-
beta.rds.amazonaws.com:5432");

        return newEndpointList;
    }

    private static String getFormattedEndpointList(List<String> endpoints) {
        return IntStream.range(0, endpoints.size())
                .mapToObj(i -> endpoints.get(i).toString())
                .collect(Collectors.joining(","));
    }
}
```

Récupération rapide après basculement avec la gestion des caches de clusters pour Aurora PostgreSQL

Pour procéder à une récupération rapide de l'instance de base de données de l'enregistreur dans vos clusters Aurora PostgreSQL en cas de basculement, utilisez la gestion des caches de clusters pour Amazon Aurora PostgreSQL. La gestion des caches de clusters préserve les performances d'une application en cas de basculement.

Dans une situation de basculement classique, vous pouvez constater une dégradation temporaire, mais conséquente, des performances après un basculement. Cela est dû au fait que le cache des tampons est vide lorsque l'instance de base de données démarre. Un cache vide est également appelé cache passif. Un cache passif nuit aux performances, car l'instance de base de données doit lire sur le disque qui est ralenti, au lieu de tirer parti des valeurs stockées dans le cache de tampons.

La gestion des caches de clusters vous permet de définir une instance de base de données d'un lecteur spécifique en tant que cible de basculement. Elle garantit que les données présentes dans le cache du lecteur désigné restent synchronisées avec les données présentes dans le cache de l'instance de base de données de l'enregistreur. Le cache du lecteur désigné comportant des valeurs prérenseignées s'appelle un cache actif. En cas de basculement, le lecteur désigné utilise les valeurs présentes dans son cache actif dès qu'il est promu comme instance de base de données du nouvel enregistreur. Grâce à cette approche, votre application bénéficie de performances de récupération largement supérieures.

La gestion du cache de cluster nécessite que l'instance de lecteur désignée ait le même type et la même taille de classe d'instance (`db.r5.2xlarge` ou `db.r5.xlarge`, par exemple) que le scripteur. Gardez cela à l'esprit lors de la création de clusters de bases de données Aurora PostgreSQL afin que vos clusters puissent récupérer lors d'un basculement. Pour obtenir une liste des types et des tailles de classes d'instance, consultez Spécifications matérielles pour les classes d'instance de base de données pour Aurora.

> **Note**
>
> La gestion des caches de clusters n'est pas prise en charge pour les clusters de bases de données secondaires Aurora PostgreSQL qui font partie des bases de données globales Aurora. Pour les clusters principaux où cette fonctionnalité est prise en charge, il est recommandé de ne pas exécuter de charge de travail sur le lecteur de niveau 0 désigné.

Table des matières

Configuration de la gestion des caches de clusters

Pour configurer la gestion du cache de cluster, procédez comme suit dans l'ordre.

Rubriques

> **Note**
>
> Attendez au moins 1 minute après avoir effectué ces étapes pour que la gestion des caches de clusters soit pleinement opérationnelle.

Activation de la gestion des caches de clusters

Pour activer la gestion du cache de cluster, procédez comme suit.

Console

Pour activer la gestion des caches de clusters

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.

3. Dans la liste, choisissez le groupe de paramètres pour votre cluster de bases de données Aurora PostgreSQL.

   Le cluster de bases de données doit utiliser un groupe de paramètres autre que le groupe par défaut, car vous ne pouvez pas modifier les valeurs d'un groupe de paramètres par défaut.
4. Sous Parameter group actions (Actions de groupe de paramètres), choisissez Edit (Modifier).
5. Définissez la valeur du paramètre de cluster `apg_ccm_enabled` sur 1.
6. Sélectionnez Save Changes.

AWS CLI

Pour activer la gestion des caches de clusters pour un cluster de bases de données Aurora PostgreSQL, utilisez la commande `modify-db-cluster-parameter-group` de la AWS CLI avec les paramètres requis suivants :

- `--db-cluster-parameter-group-name`
- `--parameters`

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my-db-cluster-parameter-group \
    --parameters "ParameterName=apg_ccm_enabled,ParameterValue=1,ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my-db-cluster-parameter-group ^
    --parameters "ParameterName=apg_ccm_enabled,ParameterValue=1,ApplyMethod=immediate"
```

Définition de la priorité du niveau de promotion pour l'instance de base de données de l'enregistreur

Pour la gestion du cache de cluster, assurez-vous que la priorité de promotion est de niveau 0 pour l'instance de base de données de l'enregistreur du cluster de bases de données Aurora PostgreSQL. La priorité du niveau de promotion est une valeur qui spécifie l'ordre dans lequel un lecteur Aurora est promu comme instance de base de données de l'enregistreur après un échec. Les valeurs valides

sont comprises dans la plage 0–15, 0 étant la priorité la plus élevée et 15 la priorité la plus faible. Pour plus d'informations sur le niveau de promotion, consultez Tolérance aux pannes pour un cluster de bases de données Aurora.

Console

Pour définir la priorité de la promotion pour l'instance de base de données de l'enregistreur sur tier-0 (niveau 0)

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez l'instance de base de données Writer (Enregistreur) du cluster de bases de données Aurora PostgreSQL.
4. Sélectionnez Modify. La page Modifier l'instance de base de données s'affiche.
5. Dans le panneau Configuration supplémentaire, choisissez tier-0 (niveau 0) pour Priorité de basculement.
6. Choisissez Continuer et vérifiez le récapitulatif des modifications.
7. Pour appliquer les modifications immédiatement après les avoir enregistrées, choisissez Appliquer immédiatement.
8. Choisissez Modifier l'instance de base de données pour enregistrer vos modifications.

AWS CLI

Pour définir la priorité du niveau de promotion sur 0 pour l'instance de base de données de l'enregistreur à l'aide de l'AWS CLI, appelez la commande modify-db-instance avec les paramètres requis suivants :

- `--db-instance-identifier`
- `--promotion-tier`
- `--apply-immediately`

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-instance \
    --db-instance-identifier writer-db-instance \
    --promotion-tier 0 \
    --apply-immediately
```

Pour Windows :

```
aws rds modify-db-instance ^
    --db-instance-identifier writer-db-instance ^
    ---promotion-tier 0  ^
    --apply-immediately
```

Définition de la priorité du niveau de promotion pour une instance de base de données du lecteur

Vous devez définir une instance de base de données de lecteur pour la gestion des caches de clusters. Pour cela, choisissez un lecteur dans le cluster Aurora PostgreSQL ayant les mêmes taille et classe d'instance que l'instance de base de données du scripteur. Par exemple, si le scripteur utilise `db.r5.xlarge`, choisissez un lecteur utilisant le même type et la même taille de classe d'instance. Définissez ensuite la priorité du niveau de promotion sur 0.

La priorité du niveau de promotion est une valeur qui spécifie l'ordre dans lequel un lecteur Aurora est promu comme instance de base de données de l'enregistreur après un échec. Les valeurs valides sont comprises dans la plage 0–15, 0 étant la priorité la plus élevée et 15 la priorité la plus faible.

Console

Pour définir la priorité de la promotion de l'instance de base de données du lecteur sur tier-0 (niveau 0)

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Pour cela, choisissez une instance de base de données du lecteur du cluster de bases de données Aurora PostgreSQL ayant la même classe d'instance que l'instance de base de données de l'enregistreur.
4. Sélectionnez Modify. La page Modifier l'instance de base de données s'affiche.
5. Dans le panneau Configuration supplémentaire, choisissez tier-0 (niveau 0) pour Priorité de basculement.

6. Choisissez Continuer et vérifiez le récapitulatif des modifications.
7. Pour appliquer les modifications immédiatement après les avoir enregistrées, choisissez Appliquer immédiatement.
8. Choisissez Modifier l'instance de base de données pour enregistrer vos modifications.

AWS CLI

Pour définir la priorité du niveau de promotion sur 0 pour l'instance de base de données du lecteur à l'aide de l'AWS CLI, appelez la commande modify-db-instance avec les paramètres requis suivants :

- `--db-instance-identifier`
- `--promotion-tier`
- `--apply-immediately`

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-instance \
    --db-instance-identifier reader-db-instance \
    --promotion-tier 0 \
    --apply-immediately
```

Pour Windows :

```
aws rds modify-db-instance ^
    --db-instance-identifier reader-db-instance ^
    ---promotion-tier 0  ^
    --apply-immediately
```

Surveillance du cache de tampons

Une fois la gestion des caches de clusters définie, vous pouvez surveiller l'état de la synchronisation entre le cache de tampon des instances de base de données d'enregistreur et le cache de tampon actif du lecteur désigné. Pour examiner le contenu du cache de tampons sur l'instance de base de données de l'enregistreur et sur l'instance de base de données du lecteur désigné, utilisez le

module pg_buffercache PostgreSQL. Pour plus d'informations, consultez la documentation sur PostgreSQL pg_buffercache.

Utilisation de la fonction **aurora_ccm_status**

La gestion des caches de clusters fournit également la fonction aurora_ccm_status. Utilisez la fonction aurora_ccm_status sur l'instance de base de données de l'enregistreur pour obtenir les informations suivantes sur la progression de l'activation du cache sur le lecteur désigné :

- buffers_sent_last_minute – Nombre de tampons envoyés au lecteur désigné au cours de la dernière minute.
- buffers_found_last_minute : nombre de tampons fréquemment consultés, identifiés au cours de la dernière minute.
- buffers_sent_last_scan – Nombre de tampons envoyés au lecteur désigné au cours de la dernière analyse terminée du cache du tampon.
- buffers_found_last_scan – Nombre de tampons identifiés comme fréquemment consultés et ayant dû être envoyés au cours de la dernière analyse terminée du cache du tampon. Les tampons déjà mis en cache sur le lecteur désigné ne sont pas envoyés.
- buffers_sent_current_scan – Nombre de tampons envoyés jusqu'à maintenant au cours de l'analyse actuelle.
- buffers_found_current_scan – Nombre de tampons identifiés comme fréquemment consultés au cours de l'analyse actuelle.
- current_scan_progress – Nombre de tampons visités jusqu'à maintenant au cours de l'analyse actuelle.

L'exemple suivant illustre l'utilisation de la fonction aurora_ccm_status pour convertir une partie du résultat en débit d'activation et en pourcentage d'activation.

```
SELECT buffers_sent_last_minute*8/60 AS warm_rate_kbps,
   100*(1.0-buffers_sent_last_scan::float/buffers_found_last_scan) AS warm_percent
   FROM aurora_ccm_status();
```

Résolution des erreurs de configuration de la gestion du cache de cluster (CCM)

Lorsque vous activez le paramètre de cluster apg_ccm_enabled, la gestion du cache de cluster est automatiquement activée au niveau de l'instance de base de données de l'enregistreur et d'une instance de base de données du lecteur sur le cluster de bases de données Aurora PostgreSQL.

Les instances d'enregistreur et de lecteur doivent utiliser le même type et la même taille de classe d'instance. La priorité de leur niveau de promotion est définie sur `0`. Les autres instances de lecteur du cluster de bases de données doivent avoir un niveau de promotion différent de zéro, tandis que la gestion du cache du cluster est désactivée pour ces instances.

Voici plusieurs scénarios qui peuvent entraîner des problèmes de configuration et désactiver la gestion du cache du cluster :

- Lorsqu'aucune instance de base de données de lecteur unique n'est définie sur le niveau de promotion 0.
- Lorsque l'instance de base de données d'enregistreur n'est pas définie sur le niveau de promotion 0.
- Lorsque plusieurs instances de base de données de lecteur sont définies sur le niveau de promotion 0.
- Lorsque les instances de base de données de l'enregistreur et d'un lecteur avec le niveau de promotion 0 n'ont pas la même taille d'instance.

Gestion de l'abandon des connexions d'Aurora PostgreSQL avec regroupement des connexions

Lorsque les applications clientes se connectent et se déconnectent si souvent que le temps de réponse du cluster de bases de données Aurora PostgreSQL ralentit, on dit que le cluster connaît un abandon de connexion. Chaque nouvelle connexion au point de terminaison du cluster de bases de données Aurora PostgreSQL consomme des ressources, ce qui réduit les ressources pouvant être utilisées pour traiter la charge de travail réelle. L'abandon de la connexion est un problème que nous vous recommandons de gérer en suivant certaines des bonnes pratiques présentées ci-dessous.

Pour commencer, vous pouvez améliorer les temps de réponse sur les clusters de bases de données Aurora PostgreSQL qui présentent des taux élevés d'abandon de connexion. Pour ce faire, vous pouvez utiliser une fonction de regroupement de connexions, telle que RDS Proxy. Une fonction de regroupement de connexions fournit un cache de connexions prêtes à être utilisées pour les clients. Presque toutes les versions d'Aurora PostgreSQL prennent en charge RDS Proxy. Pour plus d'informations, consultez Amazon RDS Proxy avec Aurora PostgreSQL.

Si votre version spécifique d'Aurora PostgreSQL ne prend pas en charge RDS Proxy, vous pouvez utiliser une autre fonction de regroupement de connexions compatible avec PostgreSQL, telle que PgBouncer. Pour en savoir plus, consultez le site Web de PgBouncer.

Pour voir si votre cluster de bases de données Aurora PostgreSQL peut bénéficier du regroupement des connexions, vous pouvez vérifier le fichier `postgresql.log` des connexions et des déconnexions. Vous pouvez également utiliser Performance Insights pour connaître le taux d'abandon de connexion de votre cluster de bases de données Aurora PostgreSQL. Vous trouverez ci-dessous des informations sur ces deux sujets.

Consignation des connexions et des déconnexions

Les paramètres `log_connections` et `log_disconnections` de PostgreSQL peuvent capturer les connexions et déconnexions à l'instance en écriture du cluster de bases de données Aurora PostgreSQL. Par défaut, ces paramètres sont désactivés. Pour activer ces paramètres, utilisez un groupe de paramètres personnalisés et activez-les en remplaçant la valeur par 1. Pour obtenir plus d'informations sur les groupes de paramètres personnalisés, consultez Groupes de paramètres de cluster de bases de données pour les clusters de bases de données Amazon Aurora. Pour vérifier les paramètres, connectez-vous au point de terminaison de votre cluster de bases de données pour Aurora PostgreSQL en utilisant psql et effectuez la requête suivante.

```
labdb=> SELECT setting FROM pg_settings
  WHERE name = 'log_connections';
 setting
---------
on
(1 row)
labdb=> SELECT setting FROM pg_settings
  WHERE name = 'log_disconnections';
setting
---------
on
(1 row)
```

Lorsque ces deux paramètres sont activés, le journal capture toutes les nouvelles connexions et déconnexions. Vous voyez l'utilisateur et la base de données pour chaque nouvelle connexion autorisée. Au moment de la déconnexion, la durée de la session est également enregistrée, comme le montre l'exemple suivant.

```
2022-03-07 21:44:53.978 UTC [16641] LOG: connection authorized: user=labtek
 database=labdb application_name=psql
2022-03-07 21:44:55.718 UTC [16641] LOG: disconnection: session time: 0:00:01.740
 user=labtek database=labdb host=[local]
```

Pour vérifier l'abandon de connexion de votre application, activez ces paramètres s'ils ne le sont pas déjà. Rassemblez ensuite les données dans le journal PostgreSQL pour les analyser en exécutant votre application avec une charge de travail et une période de temps réalistes. Vous pouvez consulter le fichier journal dans la console RDS. Choisissez l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL, puis sélectionnez l'onglet Logs & events (Journaux et événements). Pour plus d'informations, consultez Liste et affichage des fichiers journaux de base de données.

Vous pouvez aussi télécharger le fichier journal à partir de la console et utiliser la séquence de commandes suivante. Cette séquence trouve le nombre total de connexions autorisées et abandonnées par minute.

```
grep "connection authorized\|disconnection: session time:"
 postgresql.log.2022-03-21-16|\
awk {'print $1,$2}' |\
sort |\
uniq -c |\
sort -n -k1
```

Dans l'exemple de sortie, vous pouvez voir un pic de connexions autorisées suivies de déconnexions à partir de 16:12:10.

```
.....
,.....
.........
5 2022-03-21 16:11:55 connection authorized:
9 2022-03-21 16:11:55 disconnection: session
5 2022-03-21 16:11:56 connection authorized:
5 2022-03-21 16:11:57 connection authorized:
5 2022-03-21 16:11:57 disconnection: session
32 2022-03-21 16:12:10 connection authorized:
30 2022-03-21 16:12:10 disconnection: session
31 2022-03-21 16:12:11 connection authorized:
27 2022-03-21 16:12:11 disconnection: session
27 2022-03-21 16:12:12 connection authorized:
27 2022-03-21 16:12:12 disconnection: session
41 2022-03-21 16:12:13 connection authorized:
47 2022-03-21 16:12:13 disconnection: session
46 2022-03-21 16:12:14 connection authorized:
41 2022-03-21 16:12:14 disconnection: session
24 2022-03-21 16:12:15 connection authorized:
29 2022-03-21 16:12:15 disconnection: session
```

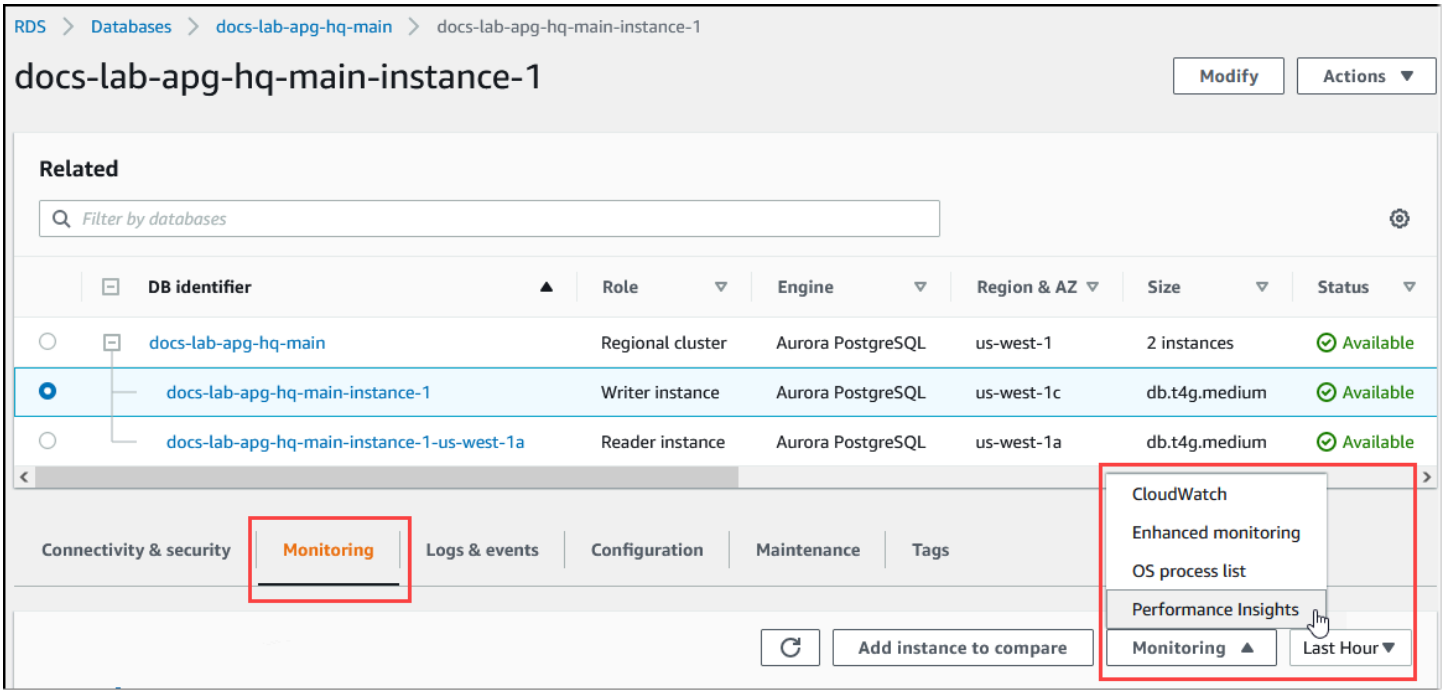
```
28 2022-03-21 16:12:16 connection authorized:
24 2022-03-21 16:12:16 disconnection: session
40 2022-03-21 16:12:17 connection authorized:
42 2022-03-21 16:12:17 disconnection: session
40 2022-03-21 16:12:18 connection authorized:
40 2022-03-21 16:12:18 disconnection: session
.....
,.......
.........
1 2022-03-21 16:14:10 connection authorized:
1 2022-03-21 16:14:10 disconnection: session
1 2022-03-21 16:15:00 connection authorized:
1 2022-03-21 16:16:00 connection authorized:
```

Grâce à ces informations, vous pouvez décider si votre charge de travail peut tirer parti d'une fonction de regroupement de connexions. Pour une analyse plus détaillée, vous pouvez utiliser Performance Insights.

Détection de l'abandon de connexions avec Performance Insights

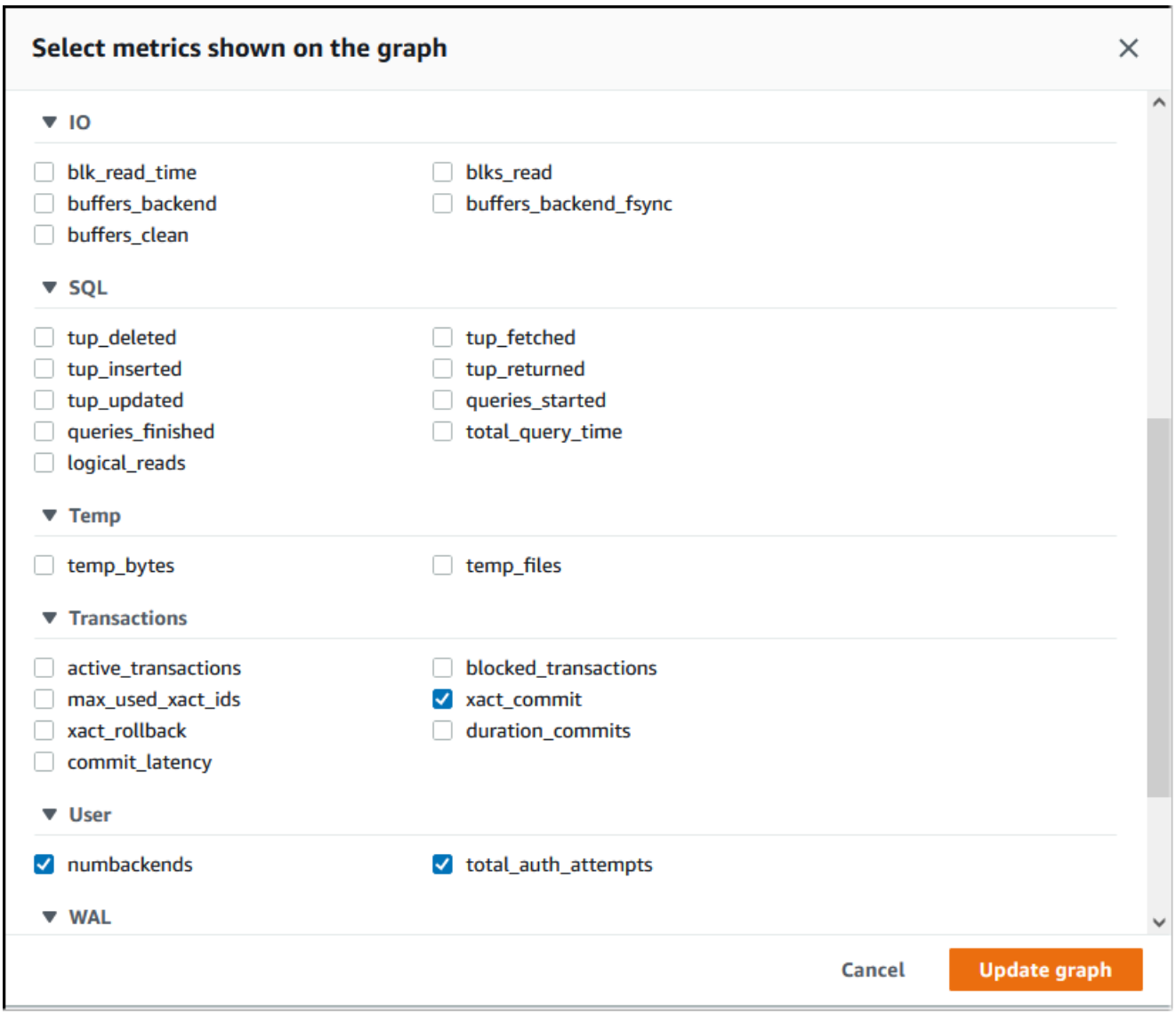
Vous pouvez utiliser Performance Insights pour évaluer le nombre d'abandons de connexion sur votre cluster de bases de données Aurora Édition compatible avec PostgreSQL. Lorsque vous créez un cluster de bases de données Aurora PostgreSQL, le paramètre pour Performance Insights est activé par défaut. Si vous avez désactivé ce choix lors de la création de votre cluster de bases de données, modifiez votre cluster pour activer cette fonctionnalité. Pour plus d'informations, consultez Modification d'un cluster de bases de données Amazon Aurora.

Lorsque Performance Insights fonctionne sur votre cluster de bases de données Aurora PostgreSQL, vous pouvez choisir les métriques que vous souhaitez surveiller. Vous pouvez accéder à Performance Insights à partir du panneau de navigation de la console. Vous pouvez également accéder à Performance Insights à partir de l'onglet Monitoring (Surveillance) de l'instance d'enregistreur pour votre cluster de bases de données Aurora PostgreSQL, comme le montre l'image suivante.



Dans la console Performance Insights, choisissez Manage metrics (Gérer les métriques). Pour analyser l'activité de connexion et de déconnexion de votre cluster de bases de données Aurora PostgreSQL, choisissez les métriques suivantes. Ce sont toutes des métriques de PostgreSQL.

- `xact_commit` : nombre de transactions dédiées.
- `total_auth_attempts` – Nombre de tentatives de connexions d'utilisateurs authentifiés par minute.
- `numbackends` : nombre de backends actuellement connectés à la base de données.

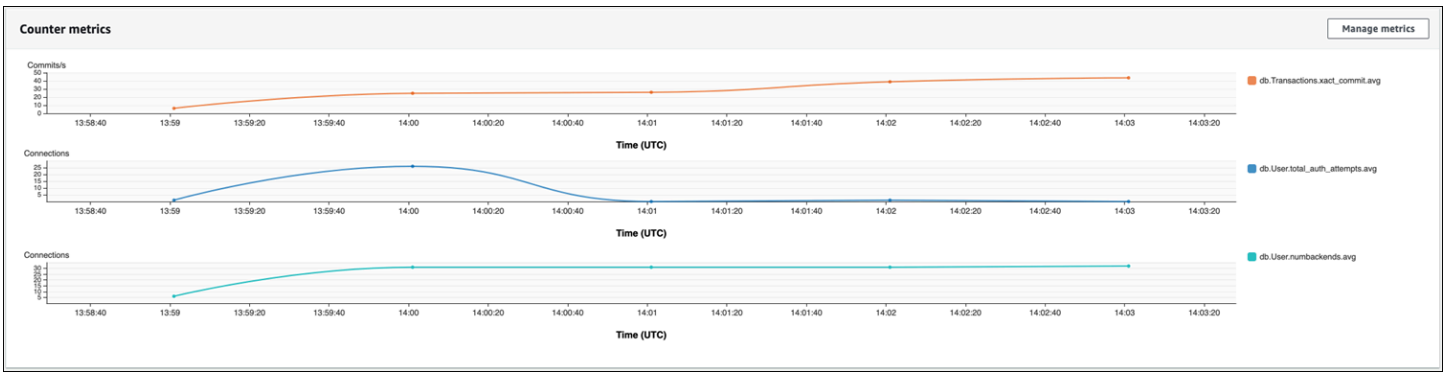


Pour enregistrer les paramètres et afficher l'activité de connexion, sélectionnez Update graph (Mettre à jour le graphique).

Dans l'image suivante, vous pouvez voir l'impact de l'exécution de pgbench avec 100 utilisateurs. La ligne indiquant les connexions est sur une pente ascendante constante. Pour en savoir plus sur pgbench et comment l'utiliser, consultez pgbench dans la documentation PostgreSQL.

L'image montre que l'exécution d'une charge de travail avec 100 utilisateurs sans fonction de regroupement de connexions peut entraîner une augmentation significative du nombre de `total_auth_attempts` pendant toute la durée du traitement de la charge de travail. Notez qu'il est préférable que `total_auth_attempts` reste le plus proche possible de zéro.

Avec le regroupement de connexions RDS Proxy, les tentatives de connexion augmentent au début de la charge de travail. Après la mise en place du regroupement de connexions, la moyenne diminue. Les ressources utilisées par les transactions et le backend restent cohérentes tout au long du traitement de la charge de travail.



Pour obtenir plus d'informations sur l'utilisation de Performance Insights avec votre cluster de bases de données Aurora PostgreSQL, consultez Surveillance de la charge de la base de données avec Performance Insights sur . Pour analyser les métriques, consultez Analyse des métriques à l'aide du tableau de bord de Performance Insights.

Démonstration des avantages du regroupement de connexions

Comme indiqué précédemment, si vous déterminez que votre cluster de bases de données Aurora PostgreSQL a un problème d'abandon de connexion, vous pouvez utiliser RDS Proxy pour améliorer les performances. Vous trouverez ci-dessous un exemple qui montre les différences dans le

traitement d'une charge de travail lorsque les connexions sont regroupées et lorsqu'elles ne le sont pas. L'exemple utilise pgbench pour modéliser une charge de travail de transaction.

Comme psql, pgbench est une application client PostgreSQL que vous pouvez installer et exécuter depuis votre machine cliente locale. Vous pouvez également l'installer et l'exécuter depuis l'instance Amazon EC2 que vous utilisez pour gérer votre cluster de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez pgbench dans la documentation de PostgreSQL.

Pour réaliser cet exemple, vous devez d'abord créer l'environnement pgbench dans votre base de données. La commande suivante désigne le modèle de base pour initialiser les tables pgbench dans la base de données spécifiée. Cet exemple utilise le compte utilisateur principal par défaut, `postgres`, pour la connexion. Modifiez-le selon vos besoins pour votre cluster de bases de données Aurora PostgreSQL. Vous créez l'environnement pgbench dans une base de données sur l'instance en écriture de votre cluster.

> **Note**
>
> Le processus d'initialisation de pgbench supprime et recrée les tables nommées `pgbench_accounts`, `pgbench_branches`, `pgbench_history` et `pgbench_tellers`. Assurez-vous que la base de données que vous choisissez pour *`dbname`* lorsque vous initialisez pgbench n'utilise pas ces noms.

```
pgbench -U postgres -h db-cluster-instance-1.111122223333.aws-region.rds.amazonaws.com
 -p 5432 -d -i -s 50 dbname
```

Pour pgbench, spécifiez les paramètres suivants.

-d

Produit un rapport de débogage pendant l'exécution de pgbench.

-h

Spécifie le point de terminaison de l'instance de base de données Aurora PostgreSQL en écriture.

-i

Initialise l'environnement pgbench dans la base de données pour les tests d'évaluation.

-p

Identifie le port utilisé pour les connexions à la base de données. La valeur par défaut pour Aurora PostgreSQL est généralement 5432 ou 5433.

-s

Spécifie le facteur d'échelle à utiliser pour remplir les tables avec des lignes. Le facteur d'échelle par défaut est 1, ce qui génère 1 ligne dans la table `pgbench_branches`, 10 lignes dans la table `pgbench_tellers` et 100 000 lignes dans la table `pgbench_accounts`.

-U

Spécifie le compte utilisateur pour l'instance d'enregistreur du cluster de bases de données Aurora PostgreSQL.

Une fois l'environnement pgbench configuré, vous pouvez exécuter des tests d'analyse comparative avec et sans regroupement de connexions. Le test par défaut consiste en une série de cinq commandes SELECT, UPDATE et INSERT par transaction qui s'exécutent de manière répétée pendant la durée spécifiée. Vous pouvez spécifier le facteur d'échelle, le nombre de clients et d'autres détails pour modéliser vos propres cas d'utilisation.

À titre d'exemple, la commande suivante exécute l'évaluation pendant 60 secondes (option -T, pour time) avec 20 connexions simultanées (option -c). L'option -C permet d'exécuter le test en utilisant une nouvelle connexion à chaque fois, plutôt qu'une fois par session client. Ce paramètre vous donne une indication de la surcharge de la connexion.

```
pgbench -h docs-lab-apg-133-test-instance-1.c3zr2auzukpa.us-west-1.rds.amazonaws.com -U
 postgres -p 5432 -T 60 -c 20 -C labdb
Password:**********
pgbench (14.3, server 13.3)
  starting vacuum...end.
  transaction type: <builtin: TPC-B (sort of)>
    scaling factor: 50
    query mode: simple
    number of clients: 20
    number of threads: 1
    duration: 60 s
    number of transactions actually processed: 495
    latency average = 2430.798 ms
    average connection time = 120.330 ms
    tps = 8.227750 (including reconnection times)
```

L'exécution de pgbench sur l'instance d'enregistreur d'un cluster de bases de données Aurora PostgreSQL sans réutilisation de connexions montre que seulement 8 transactions environ sont traitées chaque seconde. Cela donne un total de 495 transactions pendant le test d'une minute.

Si vous réutilisez les connexions, la réponse du cluster de bases de données Aurora PostgreSQL pour le nombre d'utilisateurs est presque 20 fois plus rapide. Avec la réutilisation, un total de 9 042 transactions est traité contre 495 dans le même laps de temps et pour le même nombre de connexions utilisateurs. La différence est que dans ce qui suit, chaque connexion est réutilisée.

```
pgbench -h docs-lab-apg-133-test-instance-1.c3zr2auzukpa.us-west-1.rds.amazonaws.com -U
 postgres -p 5432 -T 60 -c 20 labdb
Password:*********
pgbench (14.3, server 13.3)
      starting vacuum...end.
      transaction type: <builtin: TPC-B (sort of)>
      scaling factor: 50
      query mode: simple
      number of clients: 20
      number of threads: 1
      duration: 60 s
      number of transactions actually processed: 9042
      latency average = 127.880 ms
      initial connection time = 2311.188 ms
      tps = 156.396765 (without initial connection time)
```

Cet exemple vous montre que le regroupement des connexions peut améliorer considérablement les temps de réponse. Pour plus d'informations sur la configuration de RDS Proxy pour votre cluster de bases de données Aurora PostgreSQL, consultez Proxy Amazon RDS pour Aurora.

Gestion des connexions inactives dans PostgreSQL

Les connexions inactives se produisent lorsqu'une session de base de données reste active sur le serveur alors que l'application cliente a été abandonnée ou interrompue de manière anormale. Cette situation se produit généralement lorsque les processus clients se bloquent ou se terminent de manière inattendue sans fermer correctement leurs connexions à la base de données ou sans annuler les demandes en cours.

PostgreSQL identifie et nettoie efficacement les connexions inactives lorsque les processus du serveur sont inactifs ou tentent d'envoyer des données aux clients. Cependant, la détection est difficile pour les sessions inactives, pour les sessions qui attendent une entrée du client ou pour les sessions qui exécutent activement des requêtes. Pour gérer ces

scénarios, PostgreSQL fournit les paramètres `tcp_keepalives_*`, `tcp_user_timeout` et `client_connection_check_interval`.

Rubriques

Comprendre TCP Keepalive

TCP Keepalive est un mécanisme de niveau protocole qui permet de maintenir et de vérifier l'intégrité de la connexion. Chaque connexion TCP gère les paramètres au niveau du noyau, lesquels régissent le comportement de keepalive. Lorsque le minuteur Keepalive expire, le système procède comme suit :

- Il envoie un paquet de test sans aucune donnée, en définissant l'indicateur ACK.
- Il attend une réponse du point de terminaison distant conformément aux spécifications TCP/IP.
- Il gère l'état de la connexion en fonction de la réponse ou de l'absence de réponse.

Principaux paramètres TCP keepalive dans Aurora PostgreSQL

Paramètre	Description	Valeurs par défaut
`tcp_keepalives_idle`	Specifies number of seconds of inactivity before sending keepalive message.	300
`tcp_keepalives_interval`	Specifies number of seconds between retransmissions of unacknowledged keepalive messages.	30
`tcp_keepalives_count`	Maximum lost keepalive messages before declaring connection dead	2

Paramètre	Description	Valeurs par défaut
`tcp_user_timeout`	Specifies how long (in Milliseconds) unacknowledged data can remain before forcibly closing the connection.	0
`client_connection_check_interval`	Sets the interval (in Milliseconds) for checking client connection status during long-running queries. This ensures quicker detection of closed connections.	0

Cas d’utilisation des paramètres TCP Keepalive

Maintien en vie des sessions inactives

Pour éviter que des connexions inactives ne soient arrêtées par des pare-feux ou des routeurs pour cause d’inactivité :

- Configurez `tcp_keepalives_idle` pour envoyer des paquets keepalive à intervalles réguliers.

Détection des connexions inactives

Pour détecter rapidement les connexions inactives, procédez comme suit :

- Ajustez `tcp_keepalives_idle`, `tcp_keepalives_interval` et `tcp_keepalives_count`. Par exemple, avec les paramètres par défaut d’Aurora PostgreSQL, il faut environ une minute (2 tests de 30 secondes) pour détecter une connexion inactive. La réduction de ces valeurs peut accélérer la détection.
- Utilisez `tcp_user_timeout` pour spécifier le temps d’attente maximal pour un accusé de réception.

Les paramètres TCP keepalive aident le noyau à détecter les connexions inactives, mais PostgreSQL peut ne pas agir tant que le socket n’est pas utilisé. Si une session exécute une longue requête, il est possible que les connexions inactives ne soient détectées qu’une fois la requête terminée. Dans

PostgreSQL 14 et versions ultérieures, `client_connection_check_interval` peut accélérer la détection des connexions inactives en interrogeant régulièrement le socket lors de l'exécution de la requête.

Bonnes pratiques

- Définissez des intervalles keepalive raisonnables : réglez `tcp_user_timeout`, `tcp_keepalives_idle`, `tcp_keepalives_count` et `tcp_keepalives_interval` pour équilibrer la vitesse de détection et l'utilisation des ressources.
- Optimisez les conditions pour votre environnement : alignez les paramètres sur le comportement du réseau, les politiques de pare-feu et les besoins des sessions.
- Tirez parti des fonctionnalités de PostgreSQL : utilisez `client_connection_check_interval` dans PostgreSQL versions 14 et ultérieures pour des contrôles de connexion efficaces.

Réglage des paramètres de mémoire pour Aurora PostgreSQL

Dans Amazon Aurora PostgreSQL, vous pouvez utiliser plusieurs paramètres qui contrôlent la quantité de mémoire utilisée pour diverses tâches de traitement. Si une tâche prend plus de mémoire que la quantité définie pour un paramètre donné, Aurora PostgreSQL utilise d'autres ressources pour le traitement, par exemple en écrivant sur le disque. Cela peut entraîner un ralentissement ou un arrêt de votre cluster de bases de données Aurora PostgreSQL, avec une erreur de mémoire insuffisante.

Le réglage par défaut de chaque paramètre de mémoire permet généralement de traiter les tâches prévues. Cependant, vous pouvez également régler les paramètres liés à la mémoire de votre cluster de bases de données Aurora PostgreSQL . Vous effectuez ce réglage pour vous assurer qu'une quantité suffisante de mémoire est allouée pour traiter votre charge de travail spécifique.

Vous trouverez ci-dessous des informations sur les paramètres qui contrôlent la gestion de la mémoire. Vous pouvez également apprendre à évaluer l'utilisation de la mémoire.

Vérification et réglage des valeurs des paramètres

Les paramètres que vous pouvez définir pour gérer la mémoire et évaluer l'utilisation de la mémoire de votre cluster de bases de données Aurora PostgreSQL sont les suivants :

- `work_mem` : spécifie la quantité de mémoire que le cluster de bases de données Aurora PostgreSQL utilise pour les opérations de tri internes et les tables de hachage avant d'écrire dans des fichiers disques temporaires.
- `log_temp_files` : consigne la création de fichiers temporaires, leurs noms et leurs tailles. Lorsque ce paramètre est activé, une entrée de journal est enregistrée pour chaque fichier temporaire créé. Activez cette option pour voir à quelle fréquence votre cluster de bases de données Aurora PostgreSQL doit écrire sur le disque. Désactivez-la à nouveau après avoir recueilli des informations sur la génération de fichiers temporaires de votre cluster de bases de données Aurora PostgreSQL, pour éviter une journalisation excessive.
- `logical_decoding_work_mem` : spécifie la quantité de mémoire (en kilooctets) à utiliser par chaque tampon interne de réorganisation avant le déversement sur le disque. Cette mémoire est utilisée pour le décodage logique, qui désigne le processus de création d'un réplica. Il s'effectue en convertissant les données du fichier journal à écriture anticipée (WAL) en sortie de streaming logique nécessaire à la cible.

  La valeur de ce paramètre crée un seul tampon de la taille spécifiée pour chaque connexion de réplication. Par défaut, elle est de 65536 Ko. Une fois ce tampon rempli, l'excédent est écrit sur le disque sous forme de fichier. Pour minimiser l'activité du disque, vous pouvez définir la valeur de ce paramètre sur une valeur beaucoup plus élevée que celle de `work_mem`.

Ce sont tous des paramètres dynamiques, vous pouvez donc les modifier pour la session en cours. Pour ce faire, connectez-vous au cluster de bases de données Aurora PostgreSQL avec psql et en utilisant l'instruction `SET`, comme indiqué ci-dessous.

```
SET parameter_name TO parameter_value;
```

Les paramètres de la session ne sont valables que pour la durée de la session. Lorsque la session se termine, le paramètre revient à sa valeur dans le groupe de paramètres de la base de données. Avant de modifier un paramètre, vérifiez d'abord les valeurs actuelles en interrogeant la table `pg_settings`, comme suit.

```
SELECT unit, setting, max_val
   FROM pg_settings WHERE name='parameter_name';
```

Par exemple, pour trouver la valeur du paramètre `work_mem`, connectez-vous à l'instance d'enregistreur du cluster de bases de données Aurora PostgreSQL et exécutez la requête suivante.

```
SELECT unit, setting, max_val, pg_size_pretty(max_val::numeric)
  FROM pg_settings WHERE name='work_mem';
unit  | setting  | max_val   | pg_size_pretty
------+----------+-----------+---------------
 kB   | 1024     | 2147483647| 2048 MB
(1 row)
```

Pour modifier les paramètres afin qu'ils persistent, il faut utiliser un groupe de paramètres de cluster de bases de données personnalisé. Après avoir testé votre cluster de bases de données RDS pour PostgreSQL avec différentes valeurs pour ces paramètres à l'aide de l'instruction `SET`, vous pouvez créer un groupe de paramètres personnalisé et l'appliquer à votre cluster de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez Groupes de paramètres pour Amazon Aurora.

Comprendre le paramètre de la mémoire de travail

Le paramètre de mémoire de travail (`work_mem`) spécifie la quantité maximale de mémoire qu'Aurora PostgreSQL peut utiliser pour traiter des requêtes complexes. Les requêtes complexes comprennent celles qui impliquent des opérations de tri ou de regroupement ; en d'autres termes, les requêtes qui utilisent les clauses suivantes :

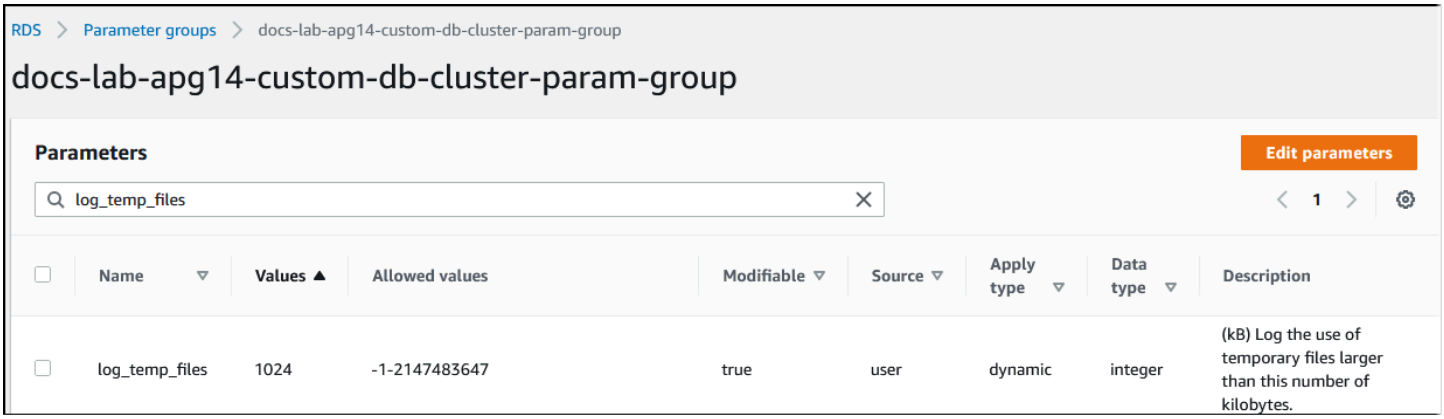
- ORDER BY
- DISTINCT
- GROUP BY
- JOIN (MERGE et HASH)

Le planificateur de requêtes affecte indirectement la façon dont votre cluster de bases de données Aurora PostgreSQL utilise la mémoire de travail. Le planificateur de requêtes génère des plans d'exécution pour le traitement des instructions SQL. Un plan donné peut décomposer une requête complexe en plusieurs unités de travail qui peuvent être exécutées en parallèle. Lorsque cela est possible, Aurora PostgreSQL utilise la quantité de mémoire spécifiée dans le paramètre `work_mem` pour chaque session avant d'écrire sur le disque pour chaque processus parallèle.

Plusieurs utilisateurs de bases de données exécutant plusieurs opérations simultanément et générant plusieurs unités de travail en parallèle peuvent épuiser la mémoire de travail allouée à votre cluster de bases de données Aurora PostgreSQL. Cela peut entraîner la création excessive de fichiers temporaires et d'entrées/sorties sur le disque, ou pire, cela peut entraîner une erreur de mémoire insuffisante.

Identifier l'utilisation des fichiers temporaires

Lorsque la mémoire nécessaire au traitement des requêtes dépasse la valeur spécifiée dans le paramètre `work_mem`, les données de travail sont déchargées sur le disque dans un fichier temporaire. Vous pouvez voir combien de fois cela se produit en activant le paramètre `log_temp_files`. Par défaut, ce paramètre est désactivé (il est défini sur -1). Pour capturer toutes les informations relatives aux fichiers temporaires, définissez ce paramètre sur 0. Définissez `log_temp_files` sur tout autre nombre entier positif pour capturer les informations de fichier temporaire pour les fichiers égaux ou supérieurs à cette quantité de données (en kilo-octets). Dans l'image suivante, vous pouvez voir un exemple de la AWS Management Console.



Après avoir configuré la journalisation des fichiers temporaires, vous pouvez tester votre propre charge de travail pour voir si votre paramètre de mémoire de travail est suffisant. Vous pouvez également simuler une charge de travail en utilisant pgbench, une application d'évaluation simple de la communauté PostgreSQL.

L'exemple suivant initialise `pgbench` (`-i`) en créant les tables et les lignes nécessaires à l'exécution des tests. Dans cet exemple, le facteur d'échelle (50 `-s`) crée 50 lignes dans la table `pgbench_branches`, 500 lignes dans `pgbench_tellers`, et 5 000 000 lignes dans la table `pgbench_accounts` de la base de données `labdb`.

```
pgbench -U postgres -h your-cluster-instance-1.111122223333.aws-regionrds.amazonaws.com
 -p 5432 -i -s 50 labdb
Password:
dropping old tables...
NOTICE:  table "pgbench_accounts" does not exist, skipping
NOTICE:  table "pgbench_branches" does not exist, skipping
NOTICE:  table "pgbench_history" does not exist, skipping
NOTICE:  table "pgbench_tellers" does not exist, skipping
creating tables...
```

```
generating data (client-side)...
5000000 of 5000000 tuples (100%) done (elapsed 15.46 s, remaining 0.00 s)
vacuuming...
creating primary keys...
done in 61.13 s (drop tables 0.08 s, create tables 0.39 s, client-side generate 54.85
 s, vacuum 2.30 s, primary keys 3.51 s)
```

Après avoir initialisé l'environnement, vous pouvez exécuter l'évaluation pour une durée spécifique (`-T`) et le nombre de clients (`-c`). Cet exemple utilise également l'option `-d` pour générer des informations de débogage à mesure que les transactions sont traitées par le cluster de bases de données Aurora PostgreSQL.

```
pgbench -h -U postgres your-cluster-instance-1.111122223333.aws-regionrds.amazonaws.com
 -p 5432 -d -T 60 -c 10 labdb
Password:*******
pgbench (14.3)
starting vacuum...end.
transaction type: <builtin: TPC-B (sort of)>
scaling factor: 50
query mode: simple
number of clients: 10
number of threads: 1
duration: 60 s
number of transactions actually processed: 1408
latency average = 398.467 ms
initial connection time = 4280.846 ms
tps = 25.096201 (without initial connection time)
```

Pour obtenir plus d'informations sur pgbench, consultez pgbench dans la documentation de PostgreSQL.

Vous pouvez utiliser la commande métacommande psql (\d) pour répertorier les relations telles que les tables, les vues et les index créés par pgbench.

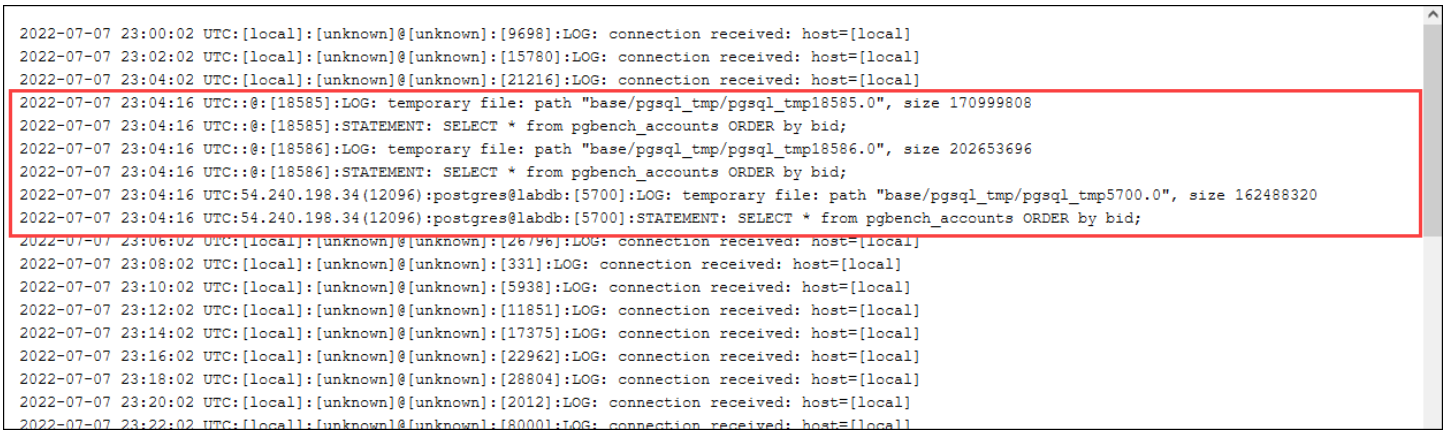
```
labdb=>  \d pgbench_accounts
 Table "public.pgbench_accounts"
  Column  |     Type      | Collation | Nullable | Default
----------+---------------+-----------+----------+---------
 aid      | integer       |           | not null |
 bid      | integer       |           |          |
 abalance | integer       |           |          |
 filler   | character(84) |           |          |
```

```
Indexes:
    "pgbench_accounts_pkey" PRIMARY KEY, btree (aid)
```

Comme le montre la sortie, la table `pgbench_accounts` est indexée sur la colonne `aid`. Pour s'assurer que la prochaine requête utilise la mémoire de travail, interrogez une colonne non indexée, comme celle présentée dans l'exemple suivant.

```
postgres=> SELECT * FROM pgbench_accounts ORDER BY bid;
```

Vérifiez la présence de fichiers temporaires dans le journal. Pour ce faire, ouvrez la AWS Management Console, choisissez l'instance du cluster de bases de données Aurora PostgreSQL, puis sélectionnez l'onglet Logs & Events (Journaux et événements). Visualisez les journaux dans la console ou téléchargez-les pour une analyse plus approfondie. Comme le montre l'image suivante, la taille des fichiers temporaires nécessaires au traitement de la requête indique que vous devriez envisager d'augmenter la quantité spécifiée pour le paramètre `work_mem`.



Vous pouvez configurer ce paramètre différemment pour les individus et les groupes, en fonction de vos besoins opérationnels. Par exemple, vous pouvez définir le paramètre `work_mem` sur 8 Go pour le rôle nommé `dev_team`.

```
postgres=> ALTER ROLE dev_team SET work_mem='8GB';
```

Avec ce paramètre pour `work_mem`, tout rôle qui est membre du rôle `dev_team` se voit attribuer jusqu'à 8 Go de mémoire de travail.

Utilisation des index pour un temps de réponse plus rapide

Si vos requêtes prennent trop de temps pour renvoyer les résultats, vous pouvez vérifier que vos index sont utilisés comme prévu. Tout d'abord, activez `\timing`, la métacommande psql, comme suit.

```
postgres=>  \timing on
```

Après avoir activé la synchronisation, utilisez une simple instruction SELECT.

```
postgres=> SELECT COUNT(*) FROM
  (SELECT * FROM pgbench_accounts
  ORDER BY bid)
  AS accounts;
count
-------
5000000
(1 row)
Time: 3119.049 ms (00:03.119)
```

Comme le montre la sortie, cette requête a pris un peu plus de 3 secondes pour se terminer. Pour améliorer le temps de réponse, créez un index sur `pgbench_accounts`, comme suit.

```
postgres=> CREATE INDEX ON pgbench_accounts(bid);
CREATE INDEX
```

Relancez la requête et remarquez que le temps de réponse est plus rapide. Dans cet exemple, la requête a été effectuée environ cinq fois plus vite, en une demi-seconde environ.

```
postgres=>  SELECT COUNT(*) FROM (SELECT * FROM pgbench_accounts ORDER BY bid) AS
 accounts;
 count
-------
 5000000
(1 row)
Time: 567.095 ms
```

Ajustement de la mémoire de travail pour le décodage logique

La réplication logique est disponible dans toutes les versions d'Aurora PostgreSQL depuis son introduction dans la version 10 de PostgreSQL. Lorsque vous configurez la réplication logique, vous

pouvez également définir le paramètre `logical_decoding_work_mem` pour spécifier la quantité de mémoire que le processus de décodage logique peut utiliser pour le processus de décodage et de streaming.

Pendant le décodage logique, les enregistrements WAL (write-ahead log) sont convertis en instructions SQL qui sont ensuite envoyées à une autre cible pour la réplication logique ou une autre tâche. Lorsqu'une transaction est écrite dans le WAL et ensuite convertie, la totalité de la transaction doit tenir dans la valeur spécifiée pour `logical_decoding_work_mem`. Par défaut, ce paramètre est défini sur 65536 Ko. Tout dépassement est écrit sur le disque. Cela signifie qu'il doit être relu à partir du disque avant de pouvoir être envoyé à sa destination, ce qui ralentit le processus global.

Vous pouvez évaluer la quantité de déversement de transactions dans votre charge de travail actuelle à un moment précis en utilisant la fonction `aurora_stat_file` comme indiqué dans l'exemple suivant.

```
SELECT split_part (filename, '/', 2)
   AS slot_name, count(1) AS num_spill_files,
   sum(used_bytes) AS slot_total_bytes,
   pg_size_pretty(sum(used_bytes)) AS slot_total_size
   FROM aurora_stat_file()
   WHERE filename like '%spill%'
   GROUP BY 1;
  slot_name | num_spill_files | slot_total_bytes | slot_total_size
------------+-----------------+------------------+-----------------
 slot_name  |       590       |      411600000   | 393 MB
(1 row)
```

Cette requête renvoie le nombre et la taille des fichiers de déversement sur votre cluster de bases de données Aurora PostgreSQL lorsque la requête est appelée. Les charges de travail qui s'exécutent depuis longtemps peuvent ne pas avoir encore de fichiers de déversement sur le disque. Pour établir le profil des charges de travail à long terme, nous vous recommandons de créer une table pour capturer les informations du fichier de déversement au fur et à mesure de l'exécution de la charge de travail. Vous pouvez créer la table comme suit.

```
CREATE TABLE spill_file_tracking AS
    SELECT now() AS spill_time,*
    FROM aurora_stat_file()
    WHERE filename LIKE '%spill%';
```

Pour voir comment les fichiers de déversement sont utilisés pendant la réplication logique, configurez un éditeur et un abonné, puis lancez une réplication simple. Pour plus d'informations, consultez Configuration de la réplication logique pour votre cluster de bases de données Aurora PostgreSQL. Une fois la réplication en cours, vous pouvez créer une tâche qui capture l'ensemble de résultats à partir de la fonction de fichier de déversement `aurora_stat_file()`, comme suit.

```
INSERT INTO spill_file_tracking
  SELECT now(),*
  FROM aurora_stat_file()
  WHERE filename LIKE '%spill%';
```

Utilisez la commande psql suivante pour exécuter la tâche une fois par seconde.

```
\watch 0.5
```

Pendant que la tâche est en cours d'exécution, connectez-vous à l'instance en écriture depuis une autre session psql. Utilisez la série d'instructions suivante pour exécuter une charge de travail qui dépasse la configuration de la mémoire et amène Aurora PostgreSQL à créer un fichier de déversement.

```
labdb=> CREATE TABLE my_table (a int PRIMARY KEY, b int);
CREATE TABLE
labdb=> INSERT INTO my_table SELECT x,x FROM generate_series(0,10000000) x;
INSERT 0 10000001
labdb=> UPDATE my_table SET b=b+1;
UPDATE 10000001
```

Ces déclarations prennent plusieurs minutes à s'effectuer. Lorsque vous avez terminé, appuyez simultanément sur les touches Ctrl et C pour arrêter la fonction de surveillance. Ensuite, utilisez la commande suivante pour créer une table qui contiendra les informations sur l'utilisation du fichier de déversement du cluster de bases de données Aurora PostgreSQL.

```
SELECT spill_time, split_part (filename, '/', 2)
    AS slot_name, count(1)
    AS spills, sum(used_bytes)
    AS slot_total_bytes, pg_size_pretty(sum(used_bytes))
    AS slot_total_size FROM spill_file_tracking
  GROUP BY 1,2 ORDER BY 1;
                  spill_time | slot_name             | spills | slot_total_bytes |
 slot_total_size
```

```
------------------------------+----------------------+--------+-----------------
+-----------------
2022-04-15 13:42:52.528272+00 | replication_slot_name | 1      | 142352280        | 136
 MB
2022-04-15 14:11:33.962216+00 | replication_slot_name | 4      | 467637996        | 446
 MB
2022-04-15 14:12:00.997636+00 | replication_slot_name | 4      | 569409176        | 543
 MB
2022-04-15 14:12:03.030245+00 | replication_slot_name | 4      | 569409176        | 543
 MB
2022-04-15 14:12:05.059761+00 | replication_slot_name | 5      | 618410996        | 590
 MB
2022-04-15 14:12:07.22905+00  | replication_slot_name | 5      | 640585316        | 611
 MB
(6 rows)
```

Le résultat montre que l'exécution de l'exemple a créé cinq fichiers de déversement qui ont utilisé 611 Mo de mémoire. Pour éviter d'écrire sur le disque, nous vous recommandons de définir le paramètre `logical_decoding_work_mem` sur la taille de mémoire la plus élevée suivante, à savoir 1024.

Utilisation des CloudWatch métriques Amazon pour analyser l'utilisation des ressources pour Aurora PostgreSQL

Aurora envoie automatiquement des données métriques par CloudWatch tranches d'une minute. Vous pouvez analyser l'utilisation des ressources pour Aurora CloudWatch PostgreSQL à l'aide de métriques. Vous pouvez évaluer le débit et l'utilisation du réseau à l'aide des métriques.

Évaluation du débit du réseau avec CloudWatch

Lorsque l'utilisation de votre système approche les limites de ressources pour votre type d'instance, le traitement peut ralentir. Vous pouvez utiliser CloudWatch Logs Insights pour surveiller l'utilisation de vos ressources de stockage et vous assurer que des ressources suffisantes sont disponibles. Si nécessaire, vous pouvez remplacer l'instance de base de données par une classe d'instance supérieure.

Le traitement du stockage Aurora peut être lent pour les raisons suivantes :

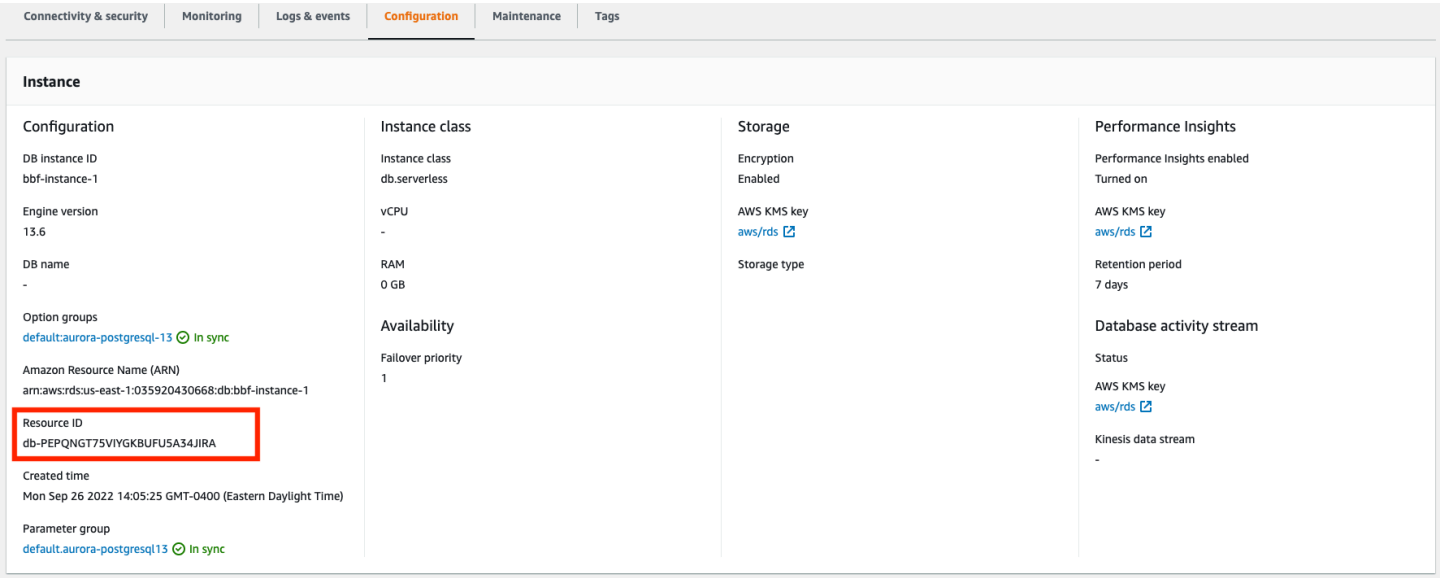
- Bande passante du réseau insuffisante entre le client et l'instance de base de données.
- Bande passante du réseau insuffisante pour le sous-système de stockage.

- Charge de travail importante pour votre type d’instance.

Vous pouvez interroger CloudWatch Logs Insights pour générer un graphique de l'utilisation des ressources de stockage Aurora afin de surveiller les ressources. Le graphique montre l’utilisation du processeur et les métriques qui vous aident à décider si vous souhaitez augmenter la taille de l’instance. Pour plus d'informations sur la syntaxe des requêtes pour CloudWatch Logs Insights, voir Syntaxe de requête CloudWatch Logs Insights

Pour les utiliser CloudWatch, vous devez exporter vos fichiers journaux Aurora PostgreSQL vers. CloudWatch Vous pouvez également modifier votre cluster existant pour exporter les journaux vers CloudWatch. Pour plus d'informations sur l'exportation des journaux vers CloudWatch, consultezActivation de l’option de publication des journaux vers Amazon CloudWatch.

Vous avez besoin de l'ID de ressource de votre instance de base de données pour interroger les CloudWatch Logs Insights. Le Resource ID (ID de ressource) est disponible dans l’onglet Configuration de votre console :



Pour interroger vos fichiers journaux pour obtenir des métriques de stockage des ressources :

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.

   La page CloudWatch d'accueil de la vue d'ensemble apparaît.

2. Si nécessaire, changez la Région AWS. Dans la barre de navigation, choisissez l'Région AWSemplacement de vos AWS ressources. Pour plus d’informations, consultez Régions et points de terminaison.

3. Dans le panneau de navigation, choisissez Logs (Journaux), puis Logs Insights.

   La page Logs Insights s’affiche.

4. Sélectionnez les fichiers journaux à analyser dans la liste déroulante.

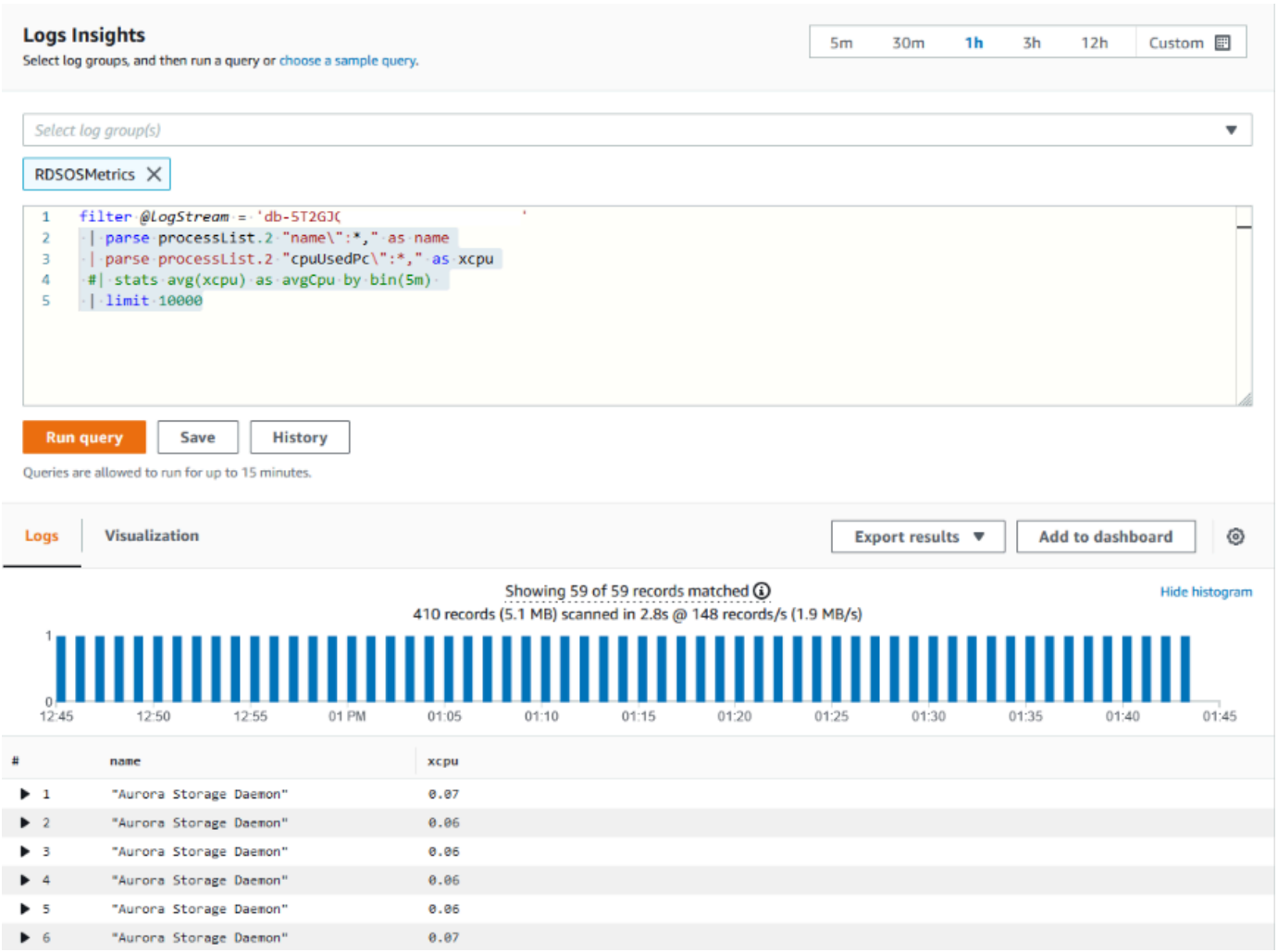
5. Entrez la requête suivante dans le champ, en remplaçant `<resource ID>` par l’ID de ressource de votre cluster de bases de données :

   ```
   filter @logStream = <resource ID> | parse @message "\"Aurora Storage Daemon\"*memoryUsedPc\":*,\"cpuUsedPc\":*," as a,memoryUsedPc,cpuUsedPc | display memoryUsedPc,cpuUsedPc #| stats avg(xcpu) as avgCpu by bin(5m) | limit 10000
   ```

6. Cliquez sur Run query (Exécuter la requête).

   Le graphique d’utilisation du stockage s’affiche.

   L’image suivante montre la page Logs Insights et l’affichage graphique.



Évaluation de l'utilisation des instances de base de données pour Aurora CloudWatch PostgreSQL à l'aide de métriques

Vous pouvez utiliser CloudWatch des métriques pour surveiller le débit de votre instance et découvrir si votre classe d'instance fournit des ressources suffisantes pour vos applications. Pour plus d'informations sur les limites de votre classe d'instance de base de données, consultez Spécifications matérielles pour les classes d'instance de base de données pour Aurora et recherchez les spécifications de votre classe d'instance de base de données afin de connaître les performances de votre réseau.

Si l'utilisation de votre instance de base de données est proche de la limite de classe d'instance, les performances peuvent commencer à ralentir. Les CloudWatch métriques peuvent confirmer cette situation afin que vous puissiez planifier une mise à l'échelle manuelle vers une classe d'instance plus importante.

Combinez les valeurs de CloudWatch métriques suivantes pour savoir si vous approchez de la limite de classe d'instance :

- NetworkThroughput— Le débit réseau reçu et transmis par les clients pour chaque instance du cluster de base de données Aurora. Cette valeur du débit n'inclut pas le trafic réseau entre les instances du cluster de bases de données et le volume de cluster.
- StorageNetworkThroughput— Le débit réseau reçu et envoyé au sous-système de stockage Aurora par chaque instance du cluster de base de données Aurora.

Ajoutez le NetworkThroughputà pour trouver le débit réseau reçu et envoyé au sous-système de stockage Aurora par chaque instance de votre cluster de base de données Aurora. StorageNetworkThroughput La limite de classe d'instance pour votre instance doit être supérieure à la somme de ces deux métriques combinées.

Vous pouvez utiliser les métriques suivantes pour consulter des informations supplémentaires sur le trafic réseau provenant de vos applications clientes lors de l'envoi et de la réception :

- NetworkReceiveThroughput— Le débit réseau reçu des clients par chaque instance du cluster de base de données Aurora PostgreSQL. Ce débit n'inclut pas le trafic réseau entre les instances du cluster de bases de données et le volume de cluster.
- NetworkTransmitThroughput : quantité de débit réseau envoyée aux clients par chaque instance du cluster de bases de données Aurora. Ce débit n'inclut pas le trafic réseau entre les instances du cluster de bases de données et le volume de cluster.
- StorageNetworkReceiveThroughput : quantité de débit réseau reçue du sous-système de stockage Aurora par chaque instance du cluster de bases de données.
- StorageNetworkTransmitThroughput— Le débit réseau envoyé au sous-système de stockage Aurora par chaque instance du cluster de base de données.

Ajoutez toutes ces métriques pour évaluer l'utilisation de votre réseau par rapport à la limite de classe d'instance. La limite de classe d'instance doit être supérieure à la somme de ces métriques combinées.

Les limites du réseau et l'utilisation du processeur pour le stockage sont mutuelles. Lorsque le débit réseau augmente, l'utilisation du processeur augmente également. La surveillance de l'utilisation du processeur et du réseau fournit des informations sur comment et pourquoi les ressources sont épuisées.

Pour minimiser l'utilisation du réseau, vous pouvez envisager ce qui suit :

- Utiliser une classe d'instance supérieure.
- Utiliser des stratégies de partitionnement `pg_partman`.
- Diviser les demandes d'écriture par lots afin de réduire le nombre total de transactions.
- Rediriger la charge de travail en lecture seule vers une instance en lecture seule.
- Supprimer tous les index inutilisés.
- Vérifier la présence d'objets gonflés et de VACUUM. En cas de gonflement important, utilisez l'extension PostgreSQL `pg_repack`. Pour plus d'informations sur `pg_repack`, consultez Reorganize tables in PostgreSQL databases with minimal locks (Réorganiser des tables dans des bases de données PostgreSQL avec des verrous minimaux).

Utilisation de la réplication logique pour effectuer une mise à niveau de version majeure pour Aurora PostgreSQL

Grâce à la réplication logique et au clonage rapide Aurora, vous pouvez effectuer une mise à niveau de version majeure qui utilise la version actuelle de la base de données Aurora PostgreSQL tout en migrant progressivement les données modifiées vers la nouvelle base de données de version majeure. Ce processus de mise à niveau à faible durée d'indisponibilité est appelé mise à niveau bleu/vert. La version actuelle de la base de données est appelée l'environnement « bleu » et la nouvelle version de la base de données est appelée l'environnement « vert ».

Le clonage rapide Aurora charge entièrement les données existantes en prenant un instantané de la base de données source. Le clonage rapide utilise un protocole de copie sur écriture construit au-dessus de la couche de stockage Aurora, qui vous permet de créer un clone de la base de données en peu de temps. Cette méthode est très efficace lors de la mise à niveau d'une grande base de données.

La réplication logique dans PostgreSQL suit et transfère les modifications de vos données de l'instance initiale à une nouvelle instance fonctionnant en parallèle jusqu'à ce que vous passiez à la version la plus récente de PostgreSQL. La réplication logique s'appuie sur un modèle publier et s'abonner. Pour plus d'informations sur la réplication logique Aurora PostgreSQL, consultez Réplication avec Amazon Aurora PostgreSQL

> **Tip**
>
> Vous pouvez minimiser la durée d'indisponibilité nécessaire à la mise à niveau d'une version majeure en utilisant la fonction gérée de déploiement bleu/vert d'Amazon RDS. Pour plus d'informations, consultez Utilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données.

Rubriques

Prérequis

Vous devez répondre aux exigences suivantes pour effectuer ce processus de mise à niveau à faible durée d'indisponibilité :

- Vous devez disposer des autorisations rds_superuser.
- Le cluster de bases de données Aurora PostgreSQL que vous souhaitez mettre à niveau doit exécuter une version prise en charge capable d'effectuer des mises à niveau de version majeure à l'aide de la réplication logique. Assurez-vous d'appliquer toutes les mises à jour et tous les correctifs de versions mineures à votre cluster de bases de données. La fonction `aurora_volume_logical_start_lsn` utilisée dans cette technique est prise en charge dans les versions suivantes d'Aurora PostgreSQL :
  - 15.2 et versions 15 ultérieures
  - 14.3 et versions 14 ultérieures
  - 13.6 et versions 13 ultérieures
  - 12.10 et versions 12 ultérieures
  - 11.15 et versions 11 ultérieures
  - 10.20 et versions 10 ultérieures

Pour plus d'informations sur la fonction `aurora_volume_logical_start_lsn`, consultez aurora_volume_logical_start_lsn.

- Toutes vos tables doivent comporter une clé primaire ou inclure une colonne d'identité PostgreSQL.
- Configurez le groupe de sécurité de votre VPC pour autoriser les accès entrants et sortants entre les deux clusters de bases de données Aurora PostgreSQL, anciens et nouveaux. Vous pouvez accorder l'accès à une plage spécifique de routage inter-domaines sans classe (CIDR) ou à un autre groupe de sécurité dans votre VPC ou dans un VPC homologue. (Peer VPC nécessite une connexion d'appairage de VPC).

> **Note**
>
> Pour obtenir des informations détaillées sur les autorisations requises pour configurer et gérer un scénario de réplication logique en cours d'exécution, consultez la documentation principale de PostgreSQL.

Limitations

Lorsque vous effectuez une mise à niveau à faible durée d'indisponibilité sur votre cluster de bases de données Aurora PostgreSQL vers une nouvelle version majeure, vous utilisez la fonctionnalité de réplication logique native de PostgreSQL. Elle possède les mêmes capacités et les mêmes limites que la réplication logique de PostgreSQL. Pour plus d'informations, consultez Réplication avec Amazon Aurora PostgreSQL.

- Les commandes du langage de définition des données (DDL) ne sont pas répliquées.
- La réplication ne prend pas en charge les modifications de schéma dans une base de données active. Le schéma est recréé dans sa forme originale au cours du processus de clonage. Si vous modifiez le schéma après le clonage, mais avant de terminer la mise à niveau, cette modification n'est pas prise en compte dans l'instance mise à niveau.
- Les objets volumineux ne sont pas répliqués, mais vous pouvez stocker des données dans des tables normales.
- La réplication n'est prise en charge que par les tables, y compris les tables partitionnées. La réplication vers d'autres types de relations, telles que les vues, les vues matérialisées ou les tables externes, n'est pas prise en charge.

- Les données des séquences ne sont pas répliquées et nécessitent une mise à jour manuelle après le basculement.

> **Note**
>
> Cette mise à jour ne prend pas en charge le script automatique. Vous devez effectuer toutes les étapes manuellement.

Réglage et vérification des valeurs des paramètres

Avant la mise à niveau, configurez l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL pour qu'elle agisse comme un serveur de publication. L'instance doit utiliser un groupe de paramètres de cluster base de données personnalisé avec les paramètres suivants :

- `rds.logical_replication` : définissez ce paramètre sur 1. Le paramètre `rds.logical_replication` a la même fonction que le paramètre `wal_level` d'un serveur PostgreSQL autonome et d'autres paramètres qui contrôlent la gestion du fichier journal en écriture.
- `max_replication_slots` : définissez ce paramètre comme le nombre total d'abonnements que vous prévoyez de créer. Si vous utilisez AWS DMS, définissez ce paramètre sur le nombre de tâches AWS DMS que vous prévoyez d'utiliser pour la capture des données modifiées à partir de ce cluster de bases de données.
- `max_wal_senders` : définissez le nombre de connexions simultanées, plus quelques connexions supplémentaires, à rendre disponible pour les tâches de gestion et les nouvelles sessions. Si vous utilisez AWS DMS, le nombre de max_wal_senders doit être égal au nombre de sessions simultanées plus le nombre de tâches AWS DMS qui peuvent être en cours à un moment donné.
- `max_logical_replication_workers` : définissez le nombre d'applications de travail de réplication logique et de synchronisation des tables que vous prévoyez. Il est généralement prudent de fixer le nombre d'applications de travail de réplication à la même valeur que celle utilisée pour max_wal_senders. Les application de travail sont extraites du pool de processus d'arrière-plan (max_worker_processes) alloué au serveur.
- `max_worker_processes` : définit le nombre de processus d'arrière-plan pour le serveur. Ce nombre doit être suffisant pour allouer des applications de travail pour la réplication, les processus auto-vacuum et les autres processus de maintenance qui peuvent avoir lieu simultanément.

Lorsque vous passez à une version plus récente d'Aurora PostgreSQL, vous devez dupliquer tous les paramètres que vous avez modifiés dans la version précédente du groupe de paramètres. Ces paramètres sont appliqués à la version mise à niveau. Vous pouvez interroger la table `pg_settings` pour obtenir une liste des paramètres afin de les recréer sur votre nouveau cluster de bases de données Aurora PostgreSQL.

Par exemple, pour obtenir les paramètres de réplication, exécutez la requête suivante :

```
SELECT name, setting FROM pg_settings WHERE name in
('rds.logical_replication', 'max_replication_slots',
'max_wal_senders', 'max_logical_replication_workers',
'max_worker_processes');
```

Mise à niveau d'Aurora PostgreSQL vers une nouvelle version majeure

Pour préparer le serveur de publication (bleu)

1. Dans l'exemple qui suit, l'instance d'enregistreur source (bleue) est un cluster de bases de données Aurora PostgreSQL exécutant PostgreSQL version 11.15. C'est le nœud éditeur dans notre scénario de réplication. Pour cette démonstration, notre instance d'enregistreur source héberge un exemple de table qui contient une série de valeurs :

   ```
   CREATE TABLE my_table (a int PRIMARY KEY);
   INSERT INTO my_table VALUES (generate_series(1,100));
   ```

2. Pour créer une publication sur l'instance source, connectez-vous au nœud en écriture de l'instance avec psql (la CLI pour PostgreSQL) ou avec le client de votre choix). Entrez la commande suivante dans chaque base de données :

   ```
   CREATE PUBLICATION publication_name FOR ALL TABLES;
   ```

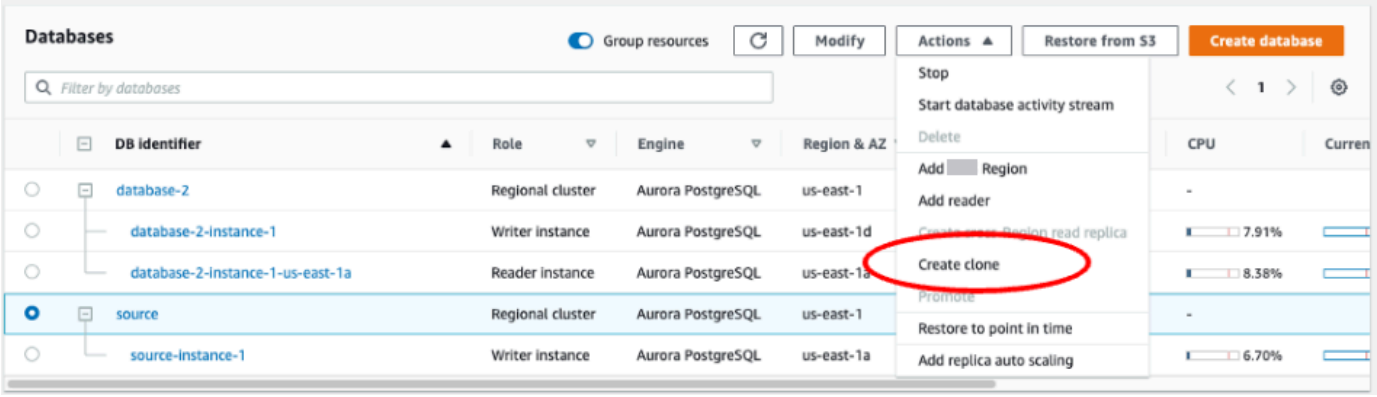
   L'élément publication_name spécifie le nom de la publication.

3. Vous devez également créer un emplacement de réplication sur l'instance. La commande suivante crée un emplacement de réplication et charge le plug-in `pgoutput` de décodage logique. Le plug-in modifie le contenu lu depuis le protocole WAL (Write-Ahead Logging) vers le protocole de réplication logique, et filtre les données selon la spécification de publication.

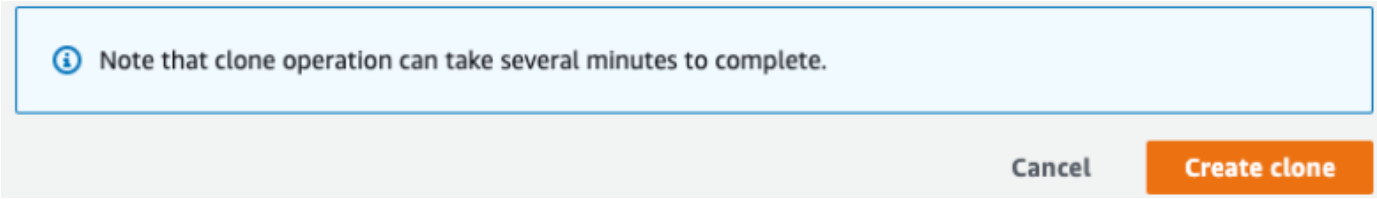
   ```
   SELECT pg_create_logical_replication_slot('replication_slot_name', 'pgoutput');
   ```

Pour cloner le serveur de publication

1. Utilisez la console Amazon RDS pour créer un clone de l'instance source. Mettez en surbrillance le nom de l'instance dans la console Amazon RDS, puis choisissez Create clone (Créer un clone) dans le menu Actions.



2. Entrez un nom unique pour l'instance. La plupart des paramètres sont des valeurs par défaut de l'instance source. Lorsque vous avez apporté les modifications requises pour la nouvelle instance, choisissez Create clone (Créer un clone).



3. Pendant que l'instance cible est en cours d'initialisation, la colonne Status (État) du nœud en écriture affiche Creating (Création) dans la colonne Status (État). Lorsque l'instance est prête, le statut passe à Available (Disponible).

Pour préparer le clone en vue d'une mise à niveau

1. Le clone est l'instance « verte » du modèle de déploiement. Il s'agit de l'hôte du nœud d'abonnement de réplication. Lorsque le nœud devient disponible, connectez-vous avec psql et interrogez le nouveau nœud en écriture pour obtenir le numéro de séquence du journal (LSN). Le LSN identifie le début d'un enregistrement dans le flux WAL.

```
SELECT aurora_volume_logical_start_lsn();
```

2. Dans la réponse à la requête, vous trouvez le numéro LSN. Vous aurez besoin de ce numéro plus tard dans le processus, alors notez-le quelque part.

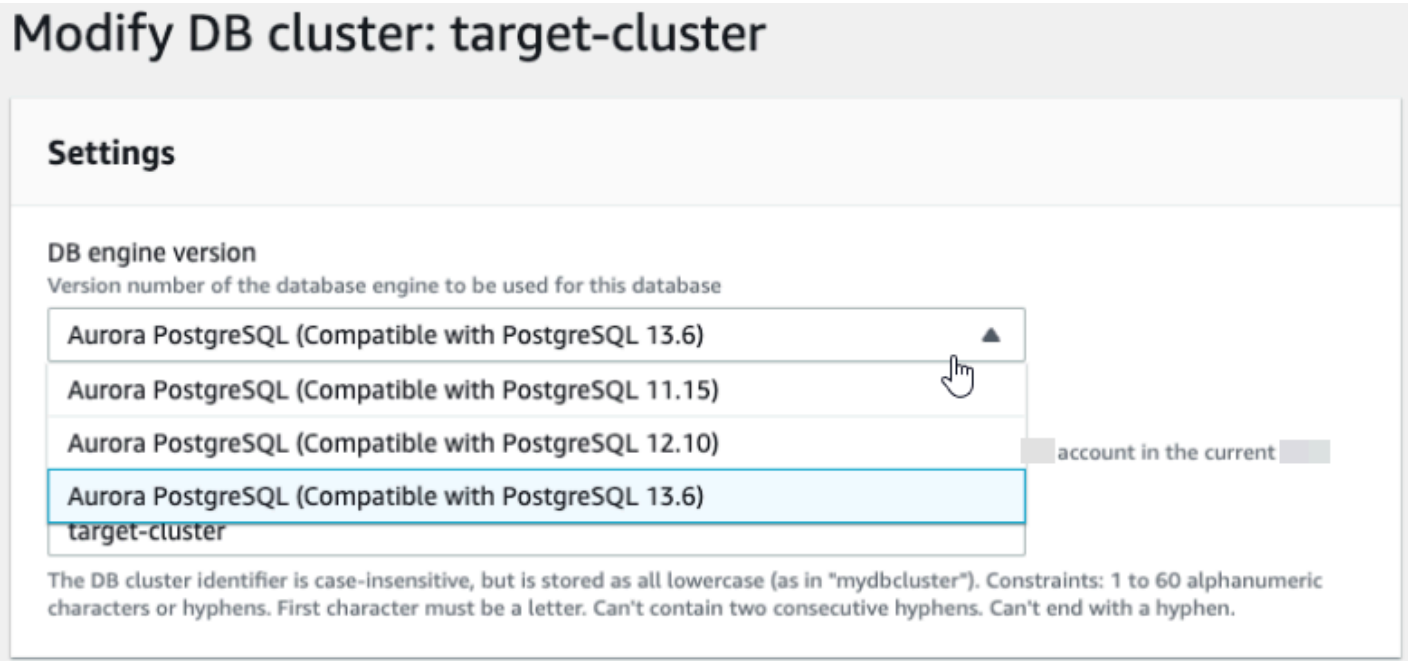
```
postgres=> SELECT aurora_volume_logical_start_lsn();
aurora_volume_logical_start_lsn
---------------
0/402E2F0
(1 row)
```

3. Avant de mettre à niveau le clone, supprimez l'emplacement de réplication du clone.

```
SELECT pg_drop_replication_slot('replication_slot_name');
```

Pour mettre à niveau le cluster vers une nouvelle version majeure

- Après avoir cloné le nœud fournisseur, utilisez la console Amazon RDS pour lancer une mise à niveau de version majeure sur le nœud d'abonnement. Mettez en surbrillance le nom de l'instance dans la console RDS, puis cliquez sur le bouton Modify (Modifier). Sélectionnez la version mise à jour et vos groupes de paramètres mis à jour, et appliquez les paramètres immédiatement pour mettre à niveau l'instance cible.



- Vous pouvez également utiliser la CLI pour effectuer une mise à niveau :

```
aws rds modify-db-cluster –db-cluster-identifier $TARGET_Aurora_ID –engine-version
 13.6 –allow-major-version-upgrade –apply-immediately
```

Pour préparer l'abonné (vert)

1. Lorsque le clone est disponible après la mise à niveau, connectez-vous à psql et définissez l'abonnement. Pour ce faire, vous devez spécifier les options suivantes dans la commande `CREATE SUBSCRIPTION` :

   - `subscription_name` : nom de l'abonnement.
   - `admin_user_name` : nom d'un utilisateur administratif avec les autorisations rds_superuser.
   - `admin_user_password` : mot de passe associé à l'utilisateur administratif.
   - `source_instance_URL` : URL de l'instance du serveur de publication.
   - `database` : base de données à laquelle le serveur d'abonnement se connectera.
   - `publication_name` : nom du serveur de publication.
   - `replication_slot_name` : nom de l'emplacement de réplication.

   ```
   CREATE SUBSCRIPTION subscription_name
   CONNECTION 'postgres://admin_user_name:admin_user_password@source_instance_URL/
   database' PUBLICATION publication_name
   WITH (copy_data = false, create_slot = false, enabled = false, connect = true,
    slot_name = 'replication_slot_name');
   ```

2. Après avoir créé l'abonnement, interrogez la vue pg_replication_origin pour récupérer la valeur roname, qui est l'identifiant de l'origine de réplication. Chaque instance possède une valeur `roname` :

   ```
   SELECT * FROM pg_replication_origin;
   ```

   Exemples :

   ```
   postgres=>
   SELECT * FROM pg_replication_origin;

   roident | roname
   ---------+----------
   1 | pg_24586
   ```

3. Fournissez le LSN que vous avez enregistré à partir de la requête précédente du nœud éditeur et la valeur `roname` renvoyée par le nœud abonné [INSTANCE] dans la commande. Cette

commande utilise la fonction pg_replication_origin_advance pour spécifier le point de départ de la séquence de journaux pour la réplication.

```
SELECT pg_replication_origin_advance('roname', 'log_sequence_number');
```

`roname` est l'identifiant renvoyé par la vue pg_replication_origin.

`log_sequence_number` est la valeur renvoyée par la requête précédente de la fonction `aurora_volume_logical_start_lsn`.

4. Utilisez ensuite la clause `ALTER SUBSCRIPTION... ENABLE` pour activer la réplication logique.

```
ALTER SUBSCRIPTION subscription_name ENABLE;
```

5. À ce stade, vous pouvez confirmer que la réplication fonctionne. Ajoutez une valeur à l'instance de publication, puis confirmez que la valeur est répliquée vers le nœud d'abonnement.

   Ensuite, utilisez la commande suivante pour surveiller le retard de réplication sur le nœud éditeur :

```
SELECT now() AS CURRENT_TIME, slot_name, active, active_pid,
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS diff_size, pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn) AS diff_bytes FROM pg_replication_slots WHERE slot_type =
 'logical';
```

   Exemples :

```
postgres=> SELECT now() AS CURRENT_TIME, slot_name, active, active_pid,
 pg_size_pretty(pg_wal_lsn_diff(pg_current_wal_lsn(),
confirmed_flush_lsn)) AS diff_size, pg_wal_lsn_diff(pg_current_wal_lsn(),
 confirmed_flush_lsn) AS diff_bytes FROM pg_replication_slots WHERE slot_type =
 'logical';

current_time                   | slot_name             | active | active_pid |
 diff_size | diff_bytes
-------------------------------+-----------------------+--------+------------
+-----------+------------
2022-04-13 15:11:00.243401+00  | replication_slot_name | t      | 21854      | 136
 bytes | 136
```

```
(1 row)
```

Vous pouvez contrôler le délai de réplication à l'aide des valeurs `diff_size` et `diff_bytes`. Lorsque ces valeurs atteignent 0, le réplica a rattrapé l'instance de base de données source.

Exécution des tâches après la mise à niveau

Lorsque la mise à niveau est terminée, l'état de l'instance s'affiche comme Available (Disponible) dans la colonne Status (État) du tableau de bord de la console. Sur la nouvelle instance, nous vous recommandons de procéder comme suit :

- Redirigez vos applications pour qu'elles pointent vers le nœud en écriture.
- Ajoutez des nœuds en lecture pour gérer la charge de travail et assurer une haute disponibilité en cas de problème avec le nœud en écriture.
- Les clusters de bases de données Aurora PostgreSQL nécessitent parfois des mises à jour du système d'exploitation. Ces mises à jour peuvent inclure une version plus récente de la bibliothèque glibc. Lors de ces mises à jour, nous vous recommandons de suivre les directives décrites dans Les classements pris en charge dans Aurora PostgreSQL.
- Mettez à jour les autorisations des utilisateurs sur la nouvelle instance pour garantir l'accès.

Après avoir testé votre application et vos données sur la nouvelle instance, nous vous recommandons de faire une dernière sauvegarde de votre instance initiale avant de la supprimer. Pour plus d'informations sur l'utilisation de la réplication logique sur un hôte Aurora, consultez Configuration de la réplication logique pour votre cluster de bases de données Aurora PostgreSQL.

Résolution des problèmes de stockage dans Aurora PostgreSQL

Si la quantité de mémoire de travail nécessaire pour les opérations de tri ou de création d'index dépasse la quantité allouée par le paramètre `work_mem`, Aurora PostgreSQL écrit les données excédentaires dans des fichiers de disque temporaires. Lorsqu'il écrit les données, Aurora PostgreSQL utilise le même espace de stockage que celui qu'il utilise pour stocker les journaux d'erreurs et de messages, c'est-à-dire le stockage local. Chaque instance de votre cluster de bases de données Aurora PostgreSQL dispose d'une certaine quantité de stockage local. La quantité de stockage est basée sur sa classe d'instance de base de données. Pour augmenter la quantité de stockage local, vous devez modifier l'instance pour utiliser une classe d'instance de base de données

plus grande. Pour connaître les spécifications de classes d'instance de base de données, consultez Spécifications matérielles pour les classes d'instance de base de données pour Aurora.

Vous pouvez surveiller l'espace de stockage local de votre cluster de bases de données Aurora PostgreSQL en observant les métriques Amazon CloudWatch pour `FreeLocalStorage`. Cette métrique indique la quantité de stockage disponible pour chaque instance de base de données dans le cluster de bases de données Aurora pour les tables et les journaux temporaires. Pour plus d'informations, consultez Surveillance des métriques Amazon Aurora avec Amazon CloudWatch.

Les opérations de tri, d'indexation et de regroupement commencent dans la mémoire de travail mais doivent souvent être déchargées vers le stockage local. Si votre cluster de bases de données Aurora PostgreSQL manque de stockage local à cause de ces types d'opérations, vous pouvez résoudre le problème en prenant l'une des mesures suivantes.

- Augmenter la quantité de mémoire de travail. Cela réduit la nécessité d'utiliser le stockage local. Par défaut, PostgreSQL alloue 4 Mo pour chaque opération de tri, de groupe et d'indexation. Pour vérifier la valeur actuelle de la mémoire de travail pour l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL, connectez-vous à l'instance en utilisant `psql` et en exécutant la commande suivante.

  ```
  postgres=> SHOW work_mem;
  work_mem
  ----------
   4MB
  (1 row)
  ```

  Vous pouvez augmenter la mémoire de travail au niveau de la session avant les opérations de tri, de regroupement et autres, comme suit.

  ```
  SET work_mem TO '1 GB';
  ```

  Pour plus d'informations sur la mémoire de travail, consultez Consommation des ressources dans la documentation PostgreSQL.

- Modifier la période de conservation des journaux afin que ceux-ci soient stockés pendant des périodes plus courtes. Pour savoir comment procéder, consultez Fichiers journaux de base de données Aurora PostgreSQL.

Pour les clusters de bases de données Aurora PostgreSQL de plus de 40 To, n'utilisez pas les classes d'instance db.t2, db.t3, ou db.t4g. Nous recommandons d'utiliser les classes d'instance de base de données T uniquement pour les serveurs de développement et de test, ou pour d'autres serveurs non dédiés à la production. Pour plus d'informations, consultez Types de classes d'instance de base de données.

Réplication avec Amazon Aurora PostgreSQL

Vous trouverez ci-dessous des informations sur la réplication avec Amazon Aurora PostgreSQL, notamment sur la manière de surveiller et d'utiliser la réplication.

Rubriques

- Utilisation de réplicas Aurora
- Amélioration de la disponibilité en lecture des réplicas Aurora
- Surveillance de la réplication Aurora PostgreSQL
- Présentation de la réplication logique PostgreSQL avec Aurora
- Configuration de la réplication logique pour votre cluster de bases de données Aurora PostgreSQL
- Désactivation de la réplication logique
- Surveillance du cache à écriture simultanée et des emplacements logiques pour la réplication logique Aurora PostgreSQL
- Exemple : utilisation de la réplication logique avec les clusters de bases de données Aurora PostgreSQL
- Exemple : réplication logique à l'aide d'Aurora PostgreSQL et AWS Database Migration Service
- Configuration de l'authentification IAM pour les connexions de réplication logiques

Utilisation de réplicas Aurora

Les réplicas Aurora sont les points de terminaison indépendants d'un cluster de bases de données Aurora, utilisés de préférence pour la mise à l'échelle des opérations de lecture et l'augmentation de la disponibilité. Un cluster de base de données Aurora peut inclure jusqu'à 15 répliques Aurora situées dans les zones de disponibilité de la AWS région du cluster de base de données Aurora.

Le volume de cluster de bases de données est composé de plusieurs copies des données du cluster de bases de données. Cependant, les données du volume de cluster sont représentées comme un

seul volume logique à l'instance principale en écriture et aux réplicas Aurora du cluster de bases de données. Pour plus d'informations sur les réplicas Aurora, consultez Réplicas Aurora.

Les réplicas Aurora fonctionnement parfaitement pour le dimensionnement en lecture, car ils sont intégralement dédiés aux opérations de lecture du volume de votre cluster. L'instance de base de données en écriture gère les opérations d'écriture. Le volume du cluster est partagé entre toutes les instances de votre cluster de bases de données Aurora PostgreSQL. Ainsi, aucun travail supplémentaire n'est nécessaire pour répliquer une copie des données pour chaque réplica Aurora.

Avec Aurora PostgreSQL, quand un réplica Aurora est supprimé, le point de terminaison de son instance est supprimé immédiatement et le réplica Aurora est supprimé du point de terminaison du lecteur. S'il y a des instructions qui s'exécutent sur le réplica Aurora en cours de suppression, une période de grâce de trois minutes est accordée. Les instructions existantes peuvent se terminer élégamment pendant la période de grâce. Lorsque la période de grâce se termine, le réplica Aurora est arrêté et supprimé.

Les clusters de base de données Aurora PostgreSQL prennent en charge les répliques Aurora dans AWS différentes régions, en utilisant la base de données globale Aurora. Pour de plus amples informations, veuillez consulter Utilisation d'Amazon Aurora Global Database.

> **Note**
>
> Avec la fonctionnalité de disponibilité en lecture, si vous voulez redémarrer les réplicas Aurora dans le cluster de bases de données, vous devez le faire manuellement. Pour les clusters de bases de données créés avant cette fonction, le redémarrage de l'instance de base de données d'écriture redémarre automatiquement les réplicas Aurora. Le redémarrage automatique rétablit un point d'entrée qui garantit la read/write cohérence au sein du cluster de base de données.

Amélioration de la disponibilité en lecture des réplicas Aurora

Aurora PostgreSQL améliore la disponibilité en lecture dans le cluster de bases de données en répondant en continu aux demandes de lecture lorsque l'instance de base de données d'écriture redémarre ou lorsque le réplica Aurora n'est pas en mesure de suivre le trafic d'écriture.

La fonction de disponibilité en lecture est disponible par défaut sur les versions suivantes d'Aurora PostgreSQL :

- 16.1 et toutes les versions ultérieures
- 15.2 et versions 15 ultérieures
- 14.7 et versions 14 ultérieures
- 13.10 et versions 13 ultérieures
- 12.14 et versions 12 ultérieures

La fonction de disponibilité en lecture est prise en charge par la base de données globale Aurora dans les versions suivantes :

- 16.1 et toutes les versions ultérieures
- 15.4 et versions 15 ultérieures
- 14.9 et versions 14 ultérieures
- 13.12 et versions 13 ultérieures
- 12.16 et versions 12 ultérieures

Pour utiliser la fonction de disponibilité en lecture pour un cluster de bases de données créé sur l'une de ces versions avant ce lancement, redémarrez l'instance d'enregistreur du cluster de bases de données.

Lorsque vous modifiez les paramètres statiques de votre cluster de bases de données Aurora PostgreSQL, vous devez redémarrer l'instance d'enregistreur pour que les modifications des paramètres soient prises en compte. Par exemple, vous devez redémarrer l'instance d'enregistreur lorsque vous définissez la valeur de `shared_buffers`. Grâce à la fonctionnalité de disponibilité en lecture des réplicas Aurora, le cluster de bases de données assure une disponibilité améliorée, ce qui réduit l'impact dont il fait l'objet lorsque l'instance d'enregistreur redémarre. Les instances de lecture ne redémarrent pas et continuent de répondre aux demandes de lecture. Pour appliquer les modifications de paramètres statiques, redémarrez chaque instance de lecteur individuelle.

Le réplica Aurora d'un cluster de bases de données Aurora PostgreSQL peut remédier à des erreurs de réplication telles que le redémarrage de l'enregistreur, le basculement, la lenteur de la réplication et les problèmes de réseau en rétablissant rapidement l'état de la base de données en mémoire après la reconnexion avec l'enregistreur. Cette approche permet aux instances des réplicas Aurora d'être cohérentes avec les dernières mises à jour de stockage alors que la base de données client est toujours disponible.

Les transactions en cours qui entrent en conflit avec la restauration de la réplication peuvent recevoir une erreur, mais le client peut réessayer ces transactions une fois que les lecteurs s'alignent sur l'enregistreur.

Surveillance des réplicas Aurora

Vous pouvez surveiller les réplicas Aurora lors d'une récupération suite à une déconnexion de l'enregistreur. Utilisez les métriques ci-dessous pour vérifier les informations les plus récentes concernant l'instance de lecteur et pour suivre les transactions en lecture seule en cours de traitement.

- La `aurora_replica_status` fonction est mise à jour pour renvoyer le maximum up-to-date d'informations pour l'instance de lecteur lorsqu'elle est toujours connectée. L'horodatage de la dernière mise à jour dans `aurora_replica_status` est toujours vide pour la ligne correspondant à l'instance de base de données sur laquelle la requête est exécutée. Cela indique que l'instance de lecteur possède les données les plus récentes.
- Lorsque le réplica Aurora se déconnecte de l'instance d'enregistreur, puis se reconnecte, l'événement de base de données suivant est émis :

  ```
  Read replica has been disconnected from the writer instance and
  reconnected.
  ```

- Lorsqu'une requête en lecture seule est annulée en raison d'un conflit de récupération, vous pouvez voir un ou plusieurs des messages d'erreur suivants dans le journal des erreurs de la base de données :

  ```
  Canceling statement due to conflict with recovery.

  User query may not have access to page data to replica disconnect.

  User query might have tried to access a file that no longer exists.

  When the replica reconnects, you will be able to repeat your command.
  ```

Limitations

Les réplicas Aurora avec la fonctionnalité de disponibilité en lecture sont soumis aux limitations suivantes :

- Les réplicas Aurora du cluster de bases de données secondaire peuvent redémarrer si les données ne peuvent pas être diffusées depuis l'instance d'enregistreur pendant la restauration de la réplication.
- Les réplicas Aurora ne prennent pas en charge la récupération de la réplication en ligne si celle-ci est déjà en cours et doit redémarrer.
- Les réplicas Aurora redémarrent quand votre instance de base de données approche du bouclage de l'ID de transaction. Pour plus d'informations sur le bouclage de l'ID de transaction, consultez Prévention des échecs de bouclage de l'ID de transaction.
- Les réplicas Aurora peuvent redémarrer lorsque le processus de réplication est bloqué dans certaines circonstances.

Surveillance de la réplication Aurora PostgreSQL

Le dimensionnement en lecture et la haute disponibilité dépendent d'un temps de retard minimal. Vous pouvez surveiller le retard d'une réplique Aurora par rapport à l'instance de base de données Writer de votre cluster de bases de données Aurora PostgreSQL en surveillant la métrique Amazon. CloudWatch `ReplicaLag` Comme les réplicas Aurora lisent à partir du même volume de cluster que l'instance de base de données en écriture, la métrique `ReplicaLag` a une signification différente pour un cluster de bases de données Aurora PostgreSQL. La métrique `ReplicaLag` d'un réplica Aurora indique le retard du cache de page du réplica Aurora par rapport à celui de l'instance de base de données en écriture.

Pour plus d'informations sur la surveillance des instances et des CloudWatch métriques RDS, consultezSurveillance des métriques d'un cluster de bases de données Amazon Aurora.

Présentation de la réplication logique PostgreSQL avec Aurora

En utilisant la fonction de réplication logique de PostgreSQL avec votre cluster de bases de données Aurora PostgreSQL, vous pouvez répliquer et synchroniser des tables individuelles plutôt que l'ensemble de l'instance de base de données. La réplication logique s'appuie sur un modèle publier et s'abonner pour répliquer les modifications depuis la source vers un ou plusieurs destinataires. Elle s'appuie sur les enregistrements de modification depuis le journal d'écriture anticipée (WAL) de PostgreSQL. La source, ou l'éditeur, envoie les données WAL pour les tables spécifiées à un ou plusieurs destinataires (abonné), répliquant ainsi les modifications et maintenant la table d'un abonné synchronisée avec la table de l'éditeur. L'ensemble des modifications apportées par l'éditeur est identifié à l'aide d'une publication. Les abonnés obtiennent les modifications en créant

un abonnement qui définit la connexion à la base de données de l'éditeur et à ses publications. Un emplacement de réplication est le mécanisme utilisé dans ce schéma pour suivre la progression d'un abonnement.

Pour les clusters de bases de données Aurora PostgreSQL, les enregistrements WAL sont enregistrés sur le stockage Aurora. Le cluster de bases de données Aurora PostgreSQL qui joue le rôle d'éditeur dans un scénario de réplication logique lit les données WAL du stockage Aurora, les décode et les envoie à l'abonné afin que les modifications puissent être appliquées à la table de cette instance. L'éditeur utilise un décodeur logique pour décoder les données destinées aux abonnés. Par défaut, les clusters de bases de données Aurora PostgreSQL utilisent le plug-in `pgoutput` PostgreSQL natif lors de l'envoi de données. D'autres décodeurs logiques sont disponibles. Par exemple, Aurora PostgreSQL prend également en charge le plug-in `wal2json` qui convertit les données WAL en JSON.

Depuis Aurora PostgreSQL version 14.5, 13.8, 12.12, et 11.17, Aurora PostgreSQL augmente le processus de réplication logique de PostgreSQL avec un cache à écriture simultanée pour améliorer les performances. Les journaux de transactions WAL sont mis en cache localement, dans une mémoire tampon, afin de réduire la quantité d'entrées/sorties sur disque, c'est-à-dire la lecture du stockage Aurora pendant le décodage logique. Le cache à écriture simultanée est utilisé par défaut lorsque vous utilisez la réplication logique pour votre cluster de bases de données Aurora PostgreSQL. Aurora fournit plusieurs fonctions que vous pouvez utiliser pour gérer le cache. Pour plus d'informations, consultez Surveillance du cache à écriture simultanée de la réplication logique Aurora PostgreSQL.

La réplication logique est prise en charge par toutes les versions actuellement disponibles d'Aurora PostgreSQL. Pour plus d'informations, consultez Mises à jour d'Amazon Aurora PostgreSQL dans Notes de mise à jour d'Aurora PostgreSQL.

La réplication logique est prise en charge par Babelfish pour Aurora PostgreSQL à partir des versions suivantes :

- 15.7 et versions ultérieures
- 16.3 et versions ultérieures

> **Note**
>
> Outre la fonctionnalité de réplication logique native de PostgreSQL introduite dans PostgreSQL 10, Aurora PostgreSQL prend également en charge l'extension `pglogical`.

Pour plus d'informations, consultez Utilisation de pglogical pour synchroniser les données entre les instances.

Pour plus d'informations sur la réplication logique PostgreSQL, consultez Réplication logique et Concepts de décodage logique dans la documentation PostgreSQL.

> **Note**
>
> PostgreSQL 16 a ajouté la prise en charge du décodage logique à partir de répliques de lecture. Cette fonctionnalité n'est pas prise en charge sur Aurora PostgreSQL.

Configuration de la réplication logique pour votre cluster de bases de données Aurora PostgreSQL

La configuration de la réplication logique nécessite des privilèges `rds_superuser`. Votre cluster de bases de données Aurora PostgreSQL doit être configuré pour utiliser un groupe de paramètres de cluster de bases de données personnalisé afin que vous puissiez définir les paramètres nécessaires comme indiqué dans la procédure suivante. Pour plus d'informations, consultez Groupes de paramètres de cluster de bases de données pour les clusters de bases de données Amazon Aurora.

Configurer la réplication logique PostgreSQL pour votre cluster de bases de données Aurora PostgreSQL

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez votre cluster de bases de données Aurora PostgreSQL.
3. Ouvrez l'onglet Configuration. Parmi les détails de l'instance, recherchez le lien Groupe de paramètres avec Groupe de paramètres de cluster DB comme Type.
4. Cliquez sur le lien pour ouvrir les paramètres personnalisés associés à votre cluster de bases de données Aurora PostgreSQL.
5. Dans le champ de recherche Parameters (Paramètres), tapez `rds` pour trouver le paramètre `rds.logical_replication`. La valeur par défaut de ce paramètre est `0`, ce qui signifie qu'il est désactivé par défaut.

6. Choisissez Edit parameters (Modifier les paramètres) pour accéder aux valeurs des propriétés, puis choisissez 1 dans le sélecteur pour activer la fonction. En fonction de votre utilisation prévue, vous devrez peut-être également modifier les paramètres suivants. Toutefois, dans de nombreux cas, les valeurs par défaut sont suffisantes.

   - `max_replication_slots` – Définissez ce paramètre sur une valeur au moins égale au nombre total prévu de publications et d'abonnements de réplication logique. Si vous utilisez AWS DMS, ce paramètre doit au moins être égal à vos tâches de capture des données de modification planifiées à partir du cluster, ainsi qu'aux publications et abonnements de réplication logique.
   - `max_wal_senderset` `max_logical_replication_workers` — Définissez ces paramètres sur une valeur au moins égale au nombre de slots de réplication logiques que vous souhaitez activer ou au nombre de AWS DMS tâches actives pour la capture des données de modification. Le fait de laisser un emplacement de réplication logique inactif empêche le vacuum de supprimer les tuples obsolètes des tables. Nous vous recommandons donc de surveiller les emplacements de réplication et de supprimer les emplacements inactifs, le cas échéant.
   - `max_worker_processes` – Définissez ce paramètre sur une valeur au moins égale au total des valeurs `max_logical_replication_workers`, `autovacuum_max_workers` et `max_parallel_workers`. Les processus de travail en arrière-plan peuvent affecter les charges de travail des applications sur les petites classes d'instance de base de données. Surveillez les performances de votre base de données si vous définissez `max_worker_processes` sur une valeur supérieure à la valeur par défaut. (La valeur par défaut est le résultat de `GREATEST(${DBInstanceVCPU*2},8}`, ce qui signifie que, par défaut, c'est huit ou deux fois l'équivalent CPU de la classe d'instance de base de données, selon la valeur la plus élevée).

   > **Note**
   >
   > Vous pouvez modifier des valeurs de paramètres dans un groupe de paramètres de base de données créé par le client. Vous ne pouvez pas modifier les valeurs de paramètres dans un groupe de paramètres de base de données par défaut.

7. Sélectionnez Enregistrer les modifications.

8. Redémarrez l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL afin que vos modifications prennent effet. Dans la console Amazon RDS, choisissez l'instance de base de données principale du cluster et choisissez Reboot (Redémarrer) dans le menu Actions.
9. Lorsque l'instance est disponible, vous pouvez vérifier que la réplication logique est activée, comme suit.

    a. Utilisez `psql` pour vous connecter à l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL.

    ```
    psql --host=your-db-cluster-instance-1.aws-region.rds.amazonaws.com --port=5432
     --username=postgres --password --dbname=labdb
    ```

    b. Vérifiez que la réplication logique a été activée à l'aide de la commande suivante.

    ```
    labdb=> SHOW rds.logical_replication;
     rds.logical_replication
    -------------------------
     on
    (1 row)
    ```

    c. Vérifiez que `wal_level` est défini sur `logical`.

    ```
    labdb=> SHOW wal_level;
      wal_level
    -----------
     logical
    (1 row)
    ```

Pour un exemple d'utilisation de la réplication logique pour maintenir une table de base de données synchronisée avec les modifications provenant d'un cluster de bases de données Aurora PostgreSQL source, consultez Exemple : utilisation de la réplication logique avec les clusters de bases de données Aurora PostgreSQL.

Désactivation de la réplication logique

Une fois vos tâches de réplication terminées, vous devez arrêter le processus de réplication, supprimer les emplacements de réplication et désactiver la réplication logique. Avant de supprimer des emplacements, assurez-vous qu'ils ne sont plus nécessaires. Les emplacements de réplication actifs ne peuvent pas être supprimés.

Désactiver la réplication logique

1. Supprimez tous les emplacements de réplication.

   Pour supprimer tous les emplacements de réplication, connectez-vous à l'éditeur et exécutez la commande SQL suivante.

   ```
   SELECT pg_drop_replication_slot(slot_name)
     FROM pg_replication_slots
    WHERE slot_name IN (SELECT slot_name FROM pg_replication_slots);
   ```

   Les emplacements de réplication ne peuvent pas être actifs lorsque vous exécutez cette commande.

2. Modifiez le groupe de paramètres personnalisé du cluster de bases de données associé à l'éditeur, comme indiqué dans Configuration de la réplication logique pour votre cluster de bases de données Aurora PostgreSQL, mais définissez le paramètre `rds.logical_replication` sur 0.

   Pour obtenir plus d'informations sur les groupes de paramètres personnalisés, consultez Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

3. Redémarrez le cluster de bases de données Aurora PostgreSQL de l'éditeur pour que les modifications apportées au paramètre `rds.logical_replication` s'appliquent.

Surveillance du cache à écriture simultanée et des emplacements logiques pour la réplication logique Aurora PostgreSQL

Surveillez le cache à écriture simultanée de la réplication logique et gérez les emplacements logiques afin d'améliorer les performances de votre cluster de bases de données Aurora PostgreSQL. Vous trouverez ci-dessous des informations supplémentaires sur le cache à écriture simultanée et les emplacements logiques.

Rubriques

- Surveillance du cache à écriture simultanée de la réplication logique Aurora PostgreSQL
- Gestion des emplacements logiques pour Aurora PostgreSQL

Surveillance du cache à écriture simultanée de la réplication logique Aurora PostgreSQL

Par défaut, Aurora PostgreSQL versions 14.5, 13.8, 12.12, et 11.17 et suivantes utilisent un cache à écriture simultanée pour améliorer les performances de la réplication logique. Sans le cache à écriture simultanée, Aurora PostgreSQL utilise la couche de stockage Aurora dans sa mise en œuvre du processus de réplication logique natif de PostgreSQL. Pour ce faire, il écrit des données WAL sur un support de stockage, puis lit les données sur ce support pour les décoder et les envoyer (répliquer) à ses cibles (abonnés). Cela peut entraîner des goulots d'étranglement pendant la réplication logique pour les clusters de bases de données Aurora PostgreSQL.

Le cache à écriture simultanée évite de dépendre trop de la couche de stockage Aurora. Au lieu de toujours écrire et lire à partir de cette couche, Aurora PostgreSQL utilise un tampon pour mettre en cache le flux WAL logique à utiliser pendant le processus de réplication, ce qui évite d'avoir à accéder au disque. Ce tampon est le cache PostgreSQL natif utilisé dans la réplication logique. Il est identifié dans les paramètres du cluster de bases de données Aurora PostgreSQL en tant que `rds.logical_wal_cache`.

Lorsque vous utilisez la réplication logique avec votre cluster de bases de données Aurora PostgreSQL (pour les versions qui prennent en charge le cache en écriture simultanée), vous pouvez surveiller le taux de réussite de la mise en cache pour voir si elle fonctionne bien pour votre cas d'utilisation. Pour ce faire, connectez-vous à l'instance en écriture de votre cluster de bases de données Aurora PostgreSQL à l'aide de `psql` et utilisez ensuite la fonction Aurora, `aurora_stat_logical_wal_cache`, comme indiqué dans l'exemple suivant.

```
SELECT * FROM aurora_stat_logical_wal_cache();
```

La fonction renvoie la sortie suivante.

```
name       | active_pid | cache_hit | cache_miss | blks_read | hit_rate |
 last_reset_timestamp
-----------+------------+-----------+------------+-----------+----------+--------------
test_slot1 | 79183      | 24        | 0          | 24        | 100.00%  | 2022-08-05
 17:39...
test_slot2 |            | 1         | 0          |  1        | 100.00%  | 2022-08-05
 17:34...
(2 rows)
```

Les valeurs `last_reset_timestamp` ont été raccourcies pour plus de lisibilité. Pour plus d'informations sur cette fonction, consultez aurora_stat_logical_wal_cache.

Aurora PostgreSQL fournit les deux fonctions suivantes pour surveiller le cache à écriture simultanée.

- La fonction `aurora_stat_logical_wal_cache` :pour la documentation de référence, consultez aurora_stat_logical_wal_cache.
- La fonction `aurora_stat_reset_wal_cache` :pour la documentation de référence, consultez aurora_stat_reset_wal_cache.

Si vous trouvez que la taille du cache WAL ajustée automatiquement n'est pas suffisante pour vos charges de travail, vous pouvez changer la valeur de `rds.logical_wal_cache` manuellement. Éléments à prendre en compte :

- Lorsque le paramètre `rds.logical_replication` est désactivé, `rds.logical_wal_cache` est défini sur zéro (0).
- Lorsque le paramètre `rds.logical_replication` est activé, `rds.logical_wal_cache` utilise la valeur par défaut, 16 Mo.
- Le paramètre `rds.logical_wal_cache` est statique et nécessite un redémarrage de l'instance de base de données pour que les modifications prennent effet. Ce paramètre est défini en termes de blocs de 8 Ko. Notez que toute valeur positive inférieure à 32 Ko est traitée comme équivalant à 32 Ko. Pour plus d'informations sur `wal_buffers`, consultez Write Ahead Log dans la documentation PostgreSQL.

Gestion des emplacements logiques pour Aurora PostgreSQL

L'activité de streaming est capturée dans la vue `pg_replication_origin_status`. Pour voir le contenu de cette vue, vous pouvez utiliser la fonction `pg_show_replication_origin_status()`, comme indiqué ci-dessous :

```
SELECT * FROM pg_show_replication_origin_status();
```

Vous pouvez obtenir la liste de vos emplacements logiques à l'aide de la requête SQL suivante.

```
SELECT * FROM pg_replication_slots;
```

Pour supprimer un emplacement logique, utilisez `pg_drop_replication_slot` avec le nom de l'option, comme indiqué dans la commande suivante.

```
SELECT pg_drop_replication_slot('test_slot');
```

Exemple : utilisation de la réplication logique avec les clusters de bases de données Aurora PostgreSQL

La procédure suivante vous indique comment démarrer la réplication logique entre deux clusters de bases de données Aurora PostgreSQL. L'éditeur et l'abonné doivent être configurés pour la réplication logique, comme indiqué dans Configuration de la réplication logique pour votre cluster de bases de données Aurora PostgreSQL.

Le cluster de bases de données Aurora PostgreSQL qui est l'éditeur désigné doit également autoriser l'accès à l'emplacement de réplication. Pour ce faire, modifiez le groupe de sécurité associé au cloud privé virtuel (VPC) du cluster de bases de données Aurora PostgreSQL basé sur le service Amazon VPC. Autorisez l'accès entrant en ajoutant le groupe de sécurité associé au VPC de l'abonné au groupe de sécurité de l'éditeur. Pour plus d'informations, consultez la rubrique Contrôler le trafic vers les ressources à l'aide de groupes de sécurité dans le Guide de l'utilisateur Amazon Virtual Private Cloud.

Une fois ces étapes préliminaires terminées, vous pouvez utiliser les commandes PostgreSQL `CREATE PUBLICATION` sur le serveur de publication et `CREATE SUBSCRIPTION` sur l'abonné, comme indiqué dans la procédure suivante.

Démarrer le processus de réplication logique entre deux clusters de bases de données Aurora PostgreSQL

Ces étapes supposent que vos clusters de bases de données Aurora PostgreSQL disposent d'une instance d'enregistreur avec une base de données dans laquelle créer les tables d'exemple.

1. Sur le cluster de bases de données Aurora PostgreSQL de l'éditeur

   a. Créez une table en utilisant l'instruction SQL suivante.

   ```
   CREATE TABLE LogicalReplicationTest (a int PRIMARY KEY);
   ```

   b. Insérez les données dans la base de données éditeur à l'aide de l'instruction SQL suivante.

```
INSERT INTO LogicalReplicationTest VALUES (generate_series(1,10000));
```

c. Vérifiez que les données existent dans la table à l'aide de l'instruction SQL suivante.

```
SELECT count(*) FROM LogicalReplicationTest;
```

d. Créez une publication pour cette table à l'aide de l'instruction `CREATE PUBLICATION`, comme suit.

```
CREATE PUBLICATION testpub FOR TABLE LogicalReplicationTest;
```

2. Sur le cluster de bases de données Aurora PostgreSQL de l'abonné

a. Créez la même table `LogicalReplicationTest` sur l'abonné que celle que vous avez créée sur l'éditeur, comme suit.

```
CREATE TABLE LogicalReplicationTest (a int PRIMARY KEY);
```

b. Vérifiez que cette table est vide.

```
SELECT count(*) FROM LogicalReplicationTest;
```

c. Créez un abonnement pour obtenir les modifications de la part de l'éditeur. Vous devez utiliser les informations suivantes concernant le cluster de bases de données Aurora PostgreSQL de l'éditeur.

- host – Instance de base de données en écriture du cluster de bases de données Aurora PostgreSQL de l'éditeur.
- port – Port sur lequel l'instance de base de données écoute. La valeur par défaut pour PostgreSQL est 5432.
- dbname – Nom de la base de données.

```
CREATE SUBSCRIPTION testsub CONNECTION
   'host=publisher-cluster-writer-endpoint port=5432 dbname=db-name user=user
 password=password'
   PUBLICATION testpub;
```

> **Note**
>
> Spécifiez un mot de passe autre que celui indiqué ici, en tant que bonne pratique de sécurité.

Une fois que l'abonnement est créé, un emplacement de réplication logique est créé chez l'éditeur.

d. Pour vérifier dans cet exemple que les données initiales sont répliquées sur l'abonné, utilisez l'instruction SQL suivante sur la base de données abonné.

```
SELECT count(*) FROM LogicalReplicationTest;
```

Toute modification ultérieure sur l'éditeur sera répliquée sur l'abonné.

La réplication logique affecte les performances. Nous vous recommandons de désactiver la réplication logique une fois vos tâches de réplication terminées.

Exemple : réplication logique à l'aide d'Aurora PostgreSQL et AWS Database Migration Service

Vous pouvez utiliser le AWS Database Migration Service (AWS DMS) pour répliquer une base de données ou une partie d'une base de données. AWS DMS À utiliser pour migrer vos données d'une base de données Aurora PostgreSQL vers une autre base de données open source ou commerciale. Pour plus d'informations AWS DMS, consultez le guide de AWS Database Migration Service l'utilisateur.

L'exemple suivant montre comment configurer une réplication logique à partir d'une base de données Aurora PostgreSQL en tant qu'éditeur, puis AWS DMS comment l'utiliser pour la migration. Cet exemple utilise les mêmes éditeur et abonné que ceux créés dans Exemple : utilisation de la réplication logique avec les clusters de bases de données Aurora PostgreSQL.

Pour configurer la réplication logique avec AWS DMS, vous devez obtenir des informations sur votre éditeur et votre abonné auprès d'Amazon RDS. En particulier, vous avez besoin d'informations sur l'instance de base de données en écriture de l'éditeur et l'instance de base de données de l'abonné.

Obtenez les informations suivantes pour l'instance de base de données en écriture de l'éditeur :

- Identifiant du cloud privé virtuel (VPC)
- Groupe de sous-réseaux
- Zone de disponibilité
- Groupe de sécurité VPC
- ID de l’instance de base de données

Obtenez les informations suivantes pour l’instance de base de données de l’abonné :

- ID de l’instance de base de données
- Moteur source

À utiliser AWS DMS pour la réplication logique avec Aurora PostgreSQL

1. Préparez la base de données de l'éditeur à utiliser AWS DMS.

   À cette fin, PostgreSQL 10.x et les bases de données ultérieures nécessitent que vous appliquiez les fonctions wrapper AWS DMS à la base de données éditeur. Pour plus d’informations sur cette étape et les étapes ultérieures, consultez les instructions dans Utilisation de PostgreSQL version 10.x et version ultérieure comme source pour AWS DMS dans le Guide de l’utilisateur AWS Database Migration Service .
2. Connectez-vous à la AWS DMS console AWS Management Console et ouvrez-la à l'adressehttps://console.aws.amazon.com/dms/v2. En haut à droite, choisissez la même AWS région dans laquelle se trouvent l'éditeur et l'abonné.
3. Créez une instance AWS DMS de réplication.

   Choisissez des valeurs identiques à celles de l’instance de base de données en écriture de votre éditeur. Tel est le cas des éléments suivants :

   - Pour VPC, choisissez le même VPC que pour l’instance de base de données en écriture.
   - Pour Replication Subnet Group (Groupe de sous-réseaux de réplication), choisissez un groupe de sous-réseaux possédant les mêmes valeurs que celui de l’instance de base de données en écriture. Créez-en un nouveau si nécessaire.
   - Pour Availability zone (Zone de disponibilité), choisissez la même zone que celle de l’instance de base de données en écriture.

- Pour VPC Security Group (Groupe de sécurité VPC), choisissez le même groupe que celui de l'instance de base de données en écriture.

4. Créez un AWS DMS point de terminaison pour la source.

   Spécifiez l'éditeur comme point de terminaison source à l'aide des paramètres suivants :

   - Pour Endpoint type (Type de point de terminaison), choisissez Source endpoint (Point de terminaison source).
   - Choisissez Select RDS DB Instance (Sélectionner une instance de base de données RDS).
   - Pour RDS Instance (Instance RDS), choisissez l'ID de base de données de l'instance de base de données en écriture de l'éditeur.
   - Pour Source engine (Moteur source), choisissez postgres.

5. Créez un AWS DMS point de terminaison pour la cible.

   Spécifiez l'éditeur comme point de terminaison cible à l'aide des paramètres suivants :

   - Pour Endpoint type (Type de point de terminaison), choisissez Target endpoint (Point de terminaison cible).
   - Choisissez Select RDS DB Instance (Sélectionner une instance de base de données RDS).
   - Pour RDS Instance (Instance RDS), choisissez l'ID de base de données de l'instance de base de données de l'abonné.
   - Choisissez une valeur pour Source engine (Moteur source). Par exemple, si l'abonné est une base de données RDS PostgreSQL, choisissez postgres. Si l'abonné est une base de données Aurora PostgreSQL, choisissez aurora-postgresql.

6. Créez une tâche AWS DMS de migration de base de données.

   Vous utilisez une tâche de migration de base de données pour spécifier les tables à migrer, pour mapper les données à l'aide d'un schéma cible et pour créer des tables sur la base de données cible. À tout le moins, utilisez les paramètres suivants pour Task configuration (Configuration de la tâche) :

   - Pour Replication instance (Instance de réplication), choisissez l'instance de réplication que vous avez créée à une étape précédente.
   - Pour Source database endpoint (Point de terminaison de la base de données source), choisissez l'éditeur source que vous avez créé à une étape précédente.

- Pour Target database endpoint (Point de terminaison de la base de données cible), choisissez l'abonné cible que vous avez créé lors d'une étape précédente.

  Le reste des détails de la tâche dépend de votre projet de migration. Pour plus d'informations sur la spécification de tous les détails relatifs aux tâches DMS, voir Utilisation des tâches AWS DMS dans le Guide de l'AWS Database Migration Service utilisateur.

Après avoir AWS DMS créé la tâche, elle commence à migrer les données de l'éditeur vers l'abonné.

Configuration de l'authentification IAM pour les connexions de réplication logiques

À partir des versions 11 et supérieures d'Aurora PostgreSQL, vous pouvez AWS utiliser l'authentification Identity and Access Management (IAM) pour les connexions de réplication. Cette fonctionnalité améliore la sécurité en vous permettant de gérer l'accès à la base de données à l'aide de rôles IAM plutôt que de mots de passe. Elle fonctionne au niveau du cluster et suit le même modèle de sécurité que l'authentification IAM standard.

L'authentification IAM pour les connexions de réplication est une fonctionnalité optionnelle. Pour l'activer, définissez le `rds.iam_auth_for_replication` paramètre sur `1` dans le groupe de paramètres de votre cluster de base de données. Comme il s'agit d'un paramètre dynamique, votre cluster de base de données n'a pas besoin de redémarrer, ce qui vous permet de tirer parti de l'authentification IAM avec les charges de travail existantes sans interruption de service. Avant d'activer cette fonctionnalité, vous devez respecter les conditions Prérequis répertoriées ci-dessous.

Prérequis

Pour utiliser l'authentification IAM pour les connexions de réplication, vous devez satisfaire à toutes les exigences suivantes :

- Votre cluster de base de données Aurora PostgreSQL doit être de version 11 ou ultérieure.
- Sur le cluster de base de données PostgreSQL Aurora de votre éditeur :
  - Activez l'authentification de base de données IAM.

    Pour de plus amples informations, veuillez consulter Activation et désactivation de l'authentification de base de données IAM.
  - Activez la réplication logique en définissant le `rds.logical_replication` paramètre sur`1`.

Pour de plus amples informations, veuillez consulter Configuration de la réplication logique pour votre cluster de bases de données Aurora PostgreSQL.

Dans la réplication logique, l'éditeur est le cluster de base de données Aurora PostgreSQL source qui envoie des données aux clusters d'abonnés. Pour plus d'informations, consultez Présentation de la réplication logique PostgreSQL avec Aurora.

> **Note**
>
> L'authentification IAM et la réplication logique doivent être activées sur le cluster de base de données PostgreSQL Aurora de votre éditeur. Si l'une ou l'autre n'est pas activée, vous ne pouvez pas utiliser l'authentification IAM pour les connexions de réplication.

Activation de l'authentification IAM pour les connexions de réplication

Procédez comme suit pour activer l'authentification IAM pour la connexion de réplication.

1. Vérifiez que votre cluster de base de données Aurora PostgreSQL répond à toutes les conditions requises pour l'authentification IAM avec des connexions de réplication. Pour en savoir plus, consultez Prérequis.
2. Configurez le `rds.iam_auth_for_replication` paramètre en modifiant le groupe de paramètres de votre cluster de base de données :
   - Définissez le paramètre `rds.iam_auth_for_replication` sur `1`. Ce paramètre dynamique ne nécessite pas de redémarrage.
3. Connectez-vous à votre base de données et accordez les rôles nécessaires à votre utilisateur de réplication :

   Les commandes SQL suivantes octroient les rôles nécessaires pour activer l'authentification IAM pour les connexions de réplication :

   ```
   -- Grant IAM authentication role
   GRANT rds_iam TO replication_user_name;
   -- Grant replication privileges
   ALTER USER replication_user_name WITH REPLICATION;
   ```

Après avoir effectué ces étapes, l'utilisateur spécifié doit utiliser l'authentification IAM pour les connexions de réplication.

> **Important**
>
> Lorsque vous activez cette fonctionnalité, les utilisateurs possédant à la fois des `rds_replication` rôles `rds_iam` et des rôles doivent utiliser l'authentification IAM pour les connexions de réplication. Cela s'applique que les rôles soient attribués directement à l'utilisateur ou hérités par le biais d'autres rôles.

Désactivation de l'authentification IAM pour les connexions de réplication

Vous pouvez désactiver l'authentification IAM pour les connexions de réplication en utilisant l'une des méthodes suivantes :

- Définissez le `rds.iam_auth_for_replication` paramètre sur `0` dans le groupe de paramètres de votre cluster de base de données
- Vous pouvez également désactiver l'une des fonctionnalités suivantes sur votre cluster de base de données Aurora PostgreSQL :
  - Désactivez la réplication logique en définissant le `rds.logical_replication` paramètre sur `0`
  - Désactiver l'authentification IAM

Lorsque vous désactivez cette fonctionnalité, les connexions de réplication peuvent utiliser les mots de passe de base de données pour l'authentification, si elles sont configurées.

> **Note**
>
> Les connexions de réplication pour les utilisateurs n'ayant pas le `rds_iam` rôle peuvent utiliser l'authentification par mot de passe même lorsque la fonctionnalité est activée.

Limites et considérations

Les limites et considérations suivantes s'appliquent lors de l'utilisation de l'authentification IAM pour les connexions de réplication.

- L'authentification IAM pour les connexions de réplication n'est disponible que pour les versions 11 et supérieures d'Aurora PostgreSQL.
- L'éditeur doit prendre en charge l'authentification IAM pour les connexions de réplication.
- Le jeton d'authentification IAM expire au bout de 15 minutes par défaut. Il se peut que vous deviez actualiser les connexions de réplication de longue durée avant l'expiration du jeton.

Transfert d’écriture local dans Aurora PostgreSQL

Le transfert d’écriture local (intracluster) permet à vos applications d’émettre des transactions de lecture/écriture directement sur un réplica Aurora. Les commandes d’écriture sont ensuite transférées à l’instance de base de données d’enregistreur pour être validées. Vous pouvez utiliser le transfert d’écriture local pour les applications qui font l’objet d’écritures occasionnelles et qui nécessitent une cohérence de lecture après écriture, ce qui permet de lire la dernière écriture dans une transaction.

Sans transfert d’écritures, vos applications doivent diviser entièrement l’ensemble du trafic de lecture et d’écriture, en conservant deux ensembles de connexions à la base de données pour envoyer le trafic au point de terminaison approprié. Les réplicas en lecture reçoivent les mises à jour de manière asynchrone de la part de l’instance d’enregistreur. De plus, comme le retard de réplication peut différer entre les réplicas en lecture, il est difficile d’obtenir une cohérence de lecture globale entre tous les réplicas. Vous devez traiter toutes les lectures qui nécessitent une cohérence de lecture après écriture sur l’instance de base de données d’enregistreur. Sinon, vous devez développer une logique d’application personnalisée complexe permettant de tirer parti de plusieurs réplicas en lecture pour assurer la capacité de mise à l’échelle tout en garantissant la cohérence.

Avec le transfert d’écriture, vous évitez d’avoir à diviser ces transactions ou d’avoir à les envoyer exclusivement à l’instance d’enregistreur. Vous n’avez pas non plus à développer une logique d’application complexe pour assure une cohérence de lecture après écriture.

Le transfert d’écriture local est disponible dans toutes les régions où Aurora PostgreSQL est proposé. Il est pris en charge par les versions Aurora PostgreSQL suivantes :

- 16.4 et versions 16 ultérieures
- 15.8 et versions 15 ultérieures
- 14.13 et versions 14 ultérieures

Le transfert d'écriture local est utilisé pour transférer les écritures provenant des réplicas qui se trouvent dans la même région. Pour transférer des écritures à partir d'un réplica global, consultez Utilisation du transfert d'écriture dans une base de données globale Amazon Aurora.

Rubriques

Limites et considérations relatives au transfert d'écriture local dans Aurora PostgreSQL

Les limites suivantes s'appliquent actuellement au transfert d'écriture local dans Aurora PostgreSQL.

- Le transfert d'écriture local n'est pas pris en charge avec le proxy RDS.
- Certaines instructions ne sont pas autorisées ou peuvent produire des résultats obsolètes lorsque vous les utilisez dans Aurora PostgreSQL avec transfert d'écriture. En outre, les fonctions et les procédures définies par l'utilisateur ne sont pas prises en charge. Par conséquent, le paramètre `EnableLocalWriteForwarding` est désactivé par défaut pour les clusters de bases de données. Avant de l'activer, vérifiez que votre code d'application n'est affecté par aucune de ces restrictions.
- Les types d'instructions SQL suivants ne sont pas pris en charge par le transfert d'écriture :

> **Note**
>
> Ces instructions peuvent être utilisées implicitement par vous-même dans votre application ou peuvent être déduites par le protocole PostgreSQL. Par exemple, la gestion des exceptions PL/SQL peut entraîner l'utilisation de SAVEPOINT, qui n'est pas une instruction prise en charge.

  - `ANALYZE`
  - `CLUSTER`
  - `COPY`

- Curseurs : les curseurs ne sont pas pris en charge. Assurez-vous donc de les fermer avant d'utiliser le transfert d'écriture local.
- Instructions DDL (Data Definition Language)
- `GRANT|REVOKE|REASSIGN OWNED|SECURITY LABEL`
- `LISTEN / NOTIFY`
- `LOCK`
- `SAVEPOINT`
- `SELECT INTO`
- `SET CONSTRAINTS`
- Mises à jour des séquences : `nextval()`, `setval()`
- `TRUNCATE`
- Commandes de validation en deux phases : `PREPARE TRANSACTION`, `COMMIT PREPARED`, `ROLLBACK PREPARED`
- Fonctions et procédures définies par l'utilisateur.
- `VACUUM`

Envisagez d'utiliser les types d'instructions SQL suivants avec le transfert d'écriture :

- Une instruction DML peut être composée de plusieurs parties, notamment d'une instruction `INSERT ... SELECT` et d'une instruction `DELETE ... WHERE`. Dans ce cas, l'instruction entière est transférée vers l'instance de base de données d'enregistreur pour y être exécutée.
- Instructions DML (Data Manipulation Language) comme `INSERT`, `DELETE` et `UPDATE`.
- Instructions `EXPLAIN` comprenant les instructions de cette liste
- Instructions `PREPARE` et `EXECUTE`.
- Instructions `SELECT FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE }`

Configuration d'Aurora PostgreSQL pour le transfert d'écriture local

À l'aide des sections suivantes, vous pouvez activer le transfert d'écriture local pour votre cluster de bases de données Amazon Aurora PostgreSQL, configurer les niveaux de cohérence et gérer les transactions avec le transfert d'écriture.

Activation du transfert d’écriture local

Par défaut, le transfert d’écriture local n’est pas activé pour les clusters de bases de données Aurora PostgreSQL. Vous activez le transfert d’écriture local au niveau du cluster, et non au niveau de l’instance.

Console

À l’aide de la AWS Management Console, cochez la case Activer le transfert d’écriture local sous Transfert d’écriture de réplica en lecture lorsque vous créez ou modifiez un cluster de bases de données.

AWS CLI

Pour activer le transfert d’écriture local à l’aide de l’interface AWS CLI, utilisez l’option `--enable-local-write-forwarding`. Cette option est utile lorsque vous créez un nouveau cluster de bases de données à l’aide de la commande `create-db-cluster`. Elle est également utile lorsque vous modifiez un cluster de bases de données existant à l’aide de la commande `modify-db-cluster`. Vous pouvez désactiver le transfert d’écriture local en utilisant l’option `--no-enable-local-write-forwarding` avec ces mêmes commandes CLI.

L’exemple suivant crée un cluster de bases de données Aurora PostgreSQL avec le transfert d’écriture local activé.

```
aws rds create-db-cluster \
--db-cluster-identifier write-forwarding-test-cluster \
--enable-local-write-forwarding \
--engine aurora-postgresql \
--engine-version 16.4 \
--master-username myuser \
--master-user-password mypassword \
--backup-retention 1
```

Vous créez ensuite des instances de base de données d’enregistreur et de lecteur afin de pouvoir utiliser le transfert d’écriture. Pour plus d’informations, consultez Création d’un cluster de bases de données Amazon Aurora.

API RDS

Pour activer le transfert d'écriture local à l'aide de l'API Amazon RDS, définissez le paramètre `EnableLocalWriteForwarding` sur `true`. Ce paramètre agit lorsque vous créez un nouveau cluster de bases de données à l'aide de l'opération `CreateDBCluster`. Il agit également lorsque vous modifiez un cluster de bases de données existant à l'aide de l'opération `ModifyDBCluster`. Vous pouvez désactiver le transfert d'écriture local en définissant le paramètre `EnableLocalWriteForwarding` sur `false`.

Activation du transfert d'écriture local pour les sessions de base de données

Le paramètre `apg_write_forward.consistency_mode` est un paramètre de base de données et un paramètre de cluster de bases de données qui permet le transfert d'écriture. Vous pouvez spécifier `SESSION`, `EVENTUAL`, `GLOBAL` ou `OFF` pour le niveau de cohérence de lecture. Pour en savoir plus sur les niveaux de cohérence, consultez la section Cohérence et isolement pour le transfert d'écriture local dans Aurora PostgreSQL.

Les règles suivantes s'appliquent à ce paramètre :

- La valeur par défaut est `SESSION`.
- Le transfert d'écriture local est disponible seulement si vous définissez `apg_write_forward.consistency_mode` sur `EVENTUAL`, `SESSION` ou `GLOBAL`. Ce paramètre n'est pertinent que dans les instances de lecteur des clusters de bases de données où le transfert d'écriture local est activé.
- Le réglage de la valeur sur `OFF` désactive le transfert d'écriture local dans la session.

Cohérence et isolement pour le transfert d'écriture local dans Aurora PostgreSQL

Vous pouvez contrôler le degré de cohérence de lecture sur un réplica en lecture. Vous pouvez ajuster le niveau de cohérence en lecture pour vous assurer que toutes les opérations d'écriture transférées de votre session sont visibles dans le réplica en lecture avant toute requête ultérieure. Vous pouvez également utiliser ce paramètre pour vous assurer que les requêtes sur le réplica en lecture voient toujours les mises à jour les plus récentes de l'instance de base de données de l'enregistreur. C'est le cas même pour celles soumises par d'autres sessions ou d'autres clusters. Pour spécifier ce type de comportement pour votre application, vous choisissez la valeur appropriée pour le paramètre de niveau session `apg_write_forward.consistency_mode`. Le paramètre `apg_write_forward.consistency_mode` n'a d'effet que sur les réplicas secondaires dans lesquels le transfert d'écriture local est activé.

> **Note**
>
> Pour le paramètre `apg_write_forward.consistency_mode`, vous pouvez spécifier les valeurs `SESSION`, `EVENTUAL`, `GLOBAL` ou `OFF`. Par défaut, cette valeur indique `SESSION`. Le réglage de la valeur sur `OFF` désactive le transfert d'écriture.

À mesure que vous augmentez le niveau de cohérence, votre application passe plus de temps à attendre que les modifications soient propagées entre les réplicas en lecture. Vous pouvez choisir l'équilibre entre une faible latence et l'assurance que les modifications apportées à d'autres emplacements seront entièrement disponibles avant l'exécution de vos requêtes.

Pour chaque paramètre de cohérence disponible, l'effet est le suivant :

- `SESSION` : une session sur un réplica en lecture qui utilise le transfert d'écriture local voit les résultats de toutes les modifications apportées au cours de cette session. Les modifications sont visibles que la transaction soit validée ou non. Si nécessaire, la requête attend que les résultats des opérations d'écriture transférées soient répliqués dans l'instance de base de données de lecteur actuelle. Elle n'attend pas les résultats mis à jour des opérations d'écriture effectuées dans d'autres sessions au sein du cluster de bases de données actuel.
- `EVENTUAL` : une session sur un réplica en lecture qui utilise le transfert d'écriture local peut voir des données légèrement obsolètes en raison d'un retard de réplication. Les résultats des opérations d'écriture dans la même session ne sont pas visibles tant que l'opération d'écriture n'est pas effectuée sur l'instance de base de données d'enregistreur et répliquée vers le réplica en lecture. La requête n'attend pas que les résultats mis à jour soient disponibles. Ainsi, elle peut récupérer les données plus anciennes ou les données mises à jour, en fonction de l'heure des instructions et de la durée du décalage de réplication.
- `GLOBAL` : une session sur un réplica en lecture voit les modifications apportées par cette session. Elle voit également toutes les modifications validées à partir de l'instance de base de données de l'enregistreur et des autres réplicas en lecture. Chaque requête peut attendre pendant une période qui varie en fonction du décalage de la session. La requête se poursuit lorsque le réplica en lecture est à jour avec toutes les données validées de l'instance de base de données de l'enregistreur, à compter du début de la requête.

> **Note**
>
> Le mode de cohérence globale a un impact sur la latence des requêtes exécutées au cours d'une session. Il applique un délai d'attente même lorsque la session n'a envoyé aucune requête d'écriture.

- `OFF` : le transfert d'écriture local est désactivé.

Dans les sessions qui utilisent le transfert d'écriture, vous pouvez utiliser les niveaux d'isolement `REPEATABLE READ` et `READ COMMITTED`. En revanche, le niveau d'isolement `SERIALIZABLE` n'est pas pris en charge.

Pour plus d'informations sur tous les paramètres impliqués dans le transfert d'écriture, consultez Paramètres par défaut pour le transfert d'écriture.

Modes d'accès aux transactions avec transfert d'écriture

Si le mode d'accès aux transactions est réglé sur lecture seule, le transfert d'écriture local n'est pas utilisé. Vous pouvez définir le mode d'accès en lecture et en écriture seules lorsque vous êtes connecté à un cluster de bases de données pour lequel le transfert d'écriture local est activé.

Pour plus d'informations sur les modes d'accès aux transactions, consultez SET TRANSACTION.

Utilisation du transfert d'écriture local pour Aurora PostgreSQL

Reportez-vous aux sections suivantes pour vérifier si le transfert d'écriture local est activé dans un cluster de bases de données, pour consulter les considérations en matière de compatibilité et pour découvrir les paramètres configurables et la configuration de l'authentification. Ces informations vous fourniront les détails nécessaires pour utiliser efficacement la fonctionnalité de transfert d'écriture local d'Aurora PostgreSQL.

> **Note**
>
> Lorsqu'une instance d'enregistreur dans un cluster utilisant le transfert d'écriture local est redémarrée, toutes les transactions et requêtes actives et transférées sur les instances de lecteur utilisant le transfert d'écriture local sont automatiquement fermées. Une fois que l'instance d'enregistreur est à nouveau disponible, vous pouvez réessayer ces transactions.

Vérification de l’activation du transfert d’écriture local dans un cluster de bases de données

Pour déterminer si vous pouvez utiliser le transfert d’écriture local dans un cluster de bases de données, vérifiez que l’attribut `LocalWriteForwardingStatus` du cluster est défini sur `enabled`.

Dans la AWS Management Console, dans l’onglet Configuration de la page de détails du cluster, vous pouvez voir l’état Activé pour Transfert local d’écriture de réplica en lecture.

Pour voir l’état du paramètre de transfert d’écriture local pour tous vos clusters, exécutez la commande AWS CLI suivante.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,LocalWriteForwardingStatus:LocalWriteForwardingStatus}

[
{
"LocalWriteForwardingStatus": "enabled",
"DBClusterIdentifier": "write-forwarding-test-cluster-1"
},
{
"LocalWriteForwardingStatus": "disabled",
"DBClusterIdentifier": "write-forwarding-test-cluster-2"
},
{
"LocalWriteForwardingStatus": "requested",
"DBClusterIdentifier": "test-global-cluster-2"
},
{
"LocalWriteForwardingStatus": "null",
"DBClusterIdentifier": "aurora-postgresql-v2-cluster"
}
]
```

Un cluster de bases de données peut avoir les valeurs suivantes pour `LocalWriteForwardingStatus` :

- `disabled` : le transfert d’écriture local est désactivé.
- `disabling` : le transfert d’écriture local est en cours de désactivation.

- `enabled` : le transfert d'écriture local est activé.
- `enabling` : le transfert d'écriture local est en cours d'activation.
- `null` : le transfert d'écriture local n'est pas disponible pour ce cluster de bases de données.
- `requested` : le transfert d'écriture local a été demandé, mais n'est pas encore actif.

Paramètres par défaut pour le transfert d'écriture

Les groupes de paramètres de cluster Aurora incluent des paramètres pour la fonctionnalité de transfert d'écriture local. Comme il s'agit de paramètres de cluster, toutes les instances de base de données de chaque cluster ont les mêmes valeurs pour ces variables. Les détails sur ces paramètres sont résumés dans le tableau suivant, avec des notes d'utilisation après le tableau.

Paramètre	Portée	Type	Valeur par défaut	Valeurs valides
`apg_write_forward.connect_timeout`	Session	secondes	30	0–2147483647
`apg_write_forward.consistency_mode`	Session	enum	Session	`SESSION`, `EVENTUAL`, `GLOBAL`, et `OFF`
`apg_write_forward.idle_in_transaction_session_timeout`	Session	millisecondes	86400000	0–2147483647
`apg_write_forward.idle_session_timeout`	Session	millisecondes	300000	0–2147483647
`apg_write_forward.max_forwarding_connections_percent`	Globale	int	25	1–100

Le paramètre `apg_write_forward.max_forwarding_connections_percent` est la limite supérieure des emplacements de connexion à la base de données qui peuvent être utilisés pour traiter les requêtes transmises par les lecteurs. Il est exprimé en pourcentage du paramètre `max_connections` de l'instance de base de données

d'enregistreur. Par exemple, si la valeur de `max_connections` est `800` et celle de `apg_write_forward.max_forwarding_connections_percent` est `10`, l'enregistreur autorise un maximum de 80 sessions transférées simultanées. Ces connexions proviennent du même groupe de connexions géré par le paramètre `max_connections`. Ce paramètre s'applique uniquement à l'instance de base de données d'enregistreur quand le transfert d'écriture local est activé sur le cluster.

Utilisez les paramètres suivants pour contrôler les demandes de transfert d'écriture local :

- `apg_write_forward.consistency_mode` : paramètre de niveau session qui contrôle le degré de cohérence en lecture sur le réplica en lecture. Les valeurs valides sont `SESSION`, `EVENTUAL`, `GLOBAL` ou `OFF`. Par défaut, cette valeur indique `SESSION`. Le réglage de la valeur sur `OFF` désactive le transfert d'écriture local dans la session. Pour en savoir plus sur les niveaux de cohérence, consultez la section Cohérence et isolement pour le transfert d'écriture local dans Aurora PostgreSQL. Ce paramètre n'est pertinent que dans les instances de lecteur où le transfert d'écriture local est activé.
- `apg_write_forward.connect_timeout` : nombre maximal de secondes pendant lesquelles le réplica en lecture attend lors de l'établissement d'une connexion à l'instance de base de données d'enregistreur avant d'abandonner. Une valeur de `0` correspond à un temps d'attente indéfini.
- `apg_write_forward.idle_in_transaction_session_timeout` : nombre de millisecondes pendant lesquelles l'instance de base de données d'enregistreur attend une activité sur une connexion transférée depuis un réplica en lecture ayant une transaction en cours avant de le fermer. Si la session reste inactive au-delà de cette durée, Aurora y met fin. La valeur `0` désactive le délai d'attente.
- `apg_write_forward.idle_session_timeout` : nombre de millisecondes pendant lesquelles l'instance de base de données d'enregistreur attend une activité sur une connexion transférée depuis un réplica en lecture avant de le fermer. Si la session reste inactive au-delà de cette durée, Aurora y met fin. La valeur `0` désactive le délai d'attente.

rdswriteforwarduser

`rdswriteforwarduser` est l'utilisateur qui nous permettra d'établir une connexion entre le réplica en lecture et l'instance de base de données d'enregistreur.

> **Note**
>
> `rdswriteforwarduser` héritera de ses privilèges CONNECT pour les bases de données clients via le rôle PUBLIC. Si les privilèges du rôle PUBLIC sont révoqués, vous devrez octroyer des privilèges GRANT CONNECT pour les bases de données vers lesquelles vous devez transférer les écritures.

Surveillance du transfert d'écriture local dans Aurora PostgreSQL

Les sections suivantes vous permettent de surveiller le transfert d'écriture local dans les clusters Aurora PostgreSQL, notamment les métriques et les événements d'attente CloudWatch pertinents afin de suivre les performances et d'identifier les problèmes potentiels.

Métriques Amazon CloudWatch et variables d'état Aurora PostgreSQL pour le transfert d'écriture

Les métriques Amazon CloudWatch suivantes s'appliquent aux instances de base de données d'enregistreur lorsque vous utilisez le transfert d'écriture sur un ou plusieurs réplicas en lecture.

Métrique CloudWatch	Unités et description
`AuroraLocal ForwardingWriterDML Throughput`	Nombre (par seconde). Nombre d'instructions DML transférées traitées chaque seconde par cette instance de base de données d'enregis treur.
`AuroraLocalForwardingWriter OpenSessions`	Nombre. Nombre de sessions ouvertes sur cette instance de base de données d'enregis treur traitant les requêtes transmises.
`AuroraLocal ForwardingWriterTo talSessions`	Nombre. Nombre total de sessions transféré es sur cette instance de base de données d'enregistreur.

Les métriques CloudWatch suivantes s'appliquent à chaque réplica en lecture. Ces métriques sont mesurées sur chaque instance de base de données de lecteur dans un cluster de bases de données où le transfert d'écriture est activé.

Métrique CloudWatch	Unité et description
`AuroraForwardingReplicaCommitThroughput`	Nombre (par seconde). Nombre d'engagements dans les sessions transmises chaque seconde par ce réplica.
`AuroraForwardingReplicaDMLLatency`	Millisecondes. Temps de réponse moyen en millisecondes des DML transférées sur le réplica.
`AuroraForwardingReplicaDMLThroughput`	Nombre (par seconde). Nombre d'instructions DML transférées que ce réplica traite chaque seconde.
`AuroraForwardingReplicaErrorSessionsLimit`	Nombre. Nombre de sessions rejetées par l'instance de base de données de l'enregistreur, car le nombre maximal de connexions ou de connexions de transfert d'écriture a été atteint.
`AuroraForwardingReplicaOpenSessions`	Nombre. Nombre de sessions qui utilisent le transfert d'écriture local sur une instance de réplica.
`AuroraForwardingReplicaReadWaitLatency`	Millisecondes. Durée moyenne en millisecondes que le réplica attend pour être cohérent avec le LSN de l'instance de base de données de l'enregistreur. Le temps d'attente de l'instance de base de données de lecteur dépend du paramètre `apg_write_forward.consistency_mode`. Pour plus d'informations sur ce paramètre, consultez the section called "Paramètres de configura

Métrique CloudWatch	Unité et description
	tion pour le transfert d'écriture dans Aurora PostgreSQL".

Événements d'attente pour le transfert d'écriture local dans Aurora PostgreSQL

Amazon Aurora génère les événements d'attente suivants lorsque vous utilisez le transfert d'écriture avec Aurora PostgreSQL.

Rubriques

- IPC:AuroraWriteForwardConnect
- IPC:AuroraWriteForwardConsistencyPoint
- IPC:AuroraWriteForwardExecute
- IPC:AuroraWriteForwardGetGlobalConsistencyPoint
- IPC:AuroraWriteForwardXactAbort
- IPC:AuroraWriteForwardXactCommit
- IPC:AuroraWriteForwardXactStart

IPC:AuroraWriteForwardConnect

L'événement `IPC:AuroraWriteForwardConnect` se produit lorsqu'un processus dorsal sur le réplica en lecture attend l'ouverture d'une connexion à l'instance de base de données d'enregistreur.

Causes probables de l'augmentation du nombre d'événements d'attente

Cet événement augmente à mesure que le nombre de tentatives de connexion du réplica en lecture au nœud d'enregistreur augmente.

Actions

Réduisez le nombre de connexions simultanées du réplica en lecture au nœud d'enregistreur.

IPC:AuroraWriteForwardConsistencyPoint

L'événement `IPC:AuroraWriteForwardConsistencyPoint` décrit la durée pendant laquelle une requête d'un nœud du réplica en lecture attend la réplication des résultats des opérations d'écriture

transférées dans la région actuelle. Cet événement n'est généré que si le paramètre de niveau session `apg_write_forward.consistency_mode` est défini sur l'une des valeurs suivantes :

- `SESSION` : les requêtes d'un réplica en lecture attendent les résultats de toutes les modifications apportées au cours de cette session.
- `GLOBAL` : les requêtes d'un réplica en lecture attendent les résultats des modifications apportées par cette session, ainsi que toutes les modifications validées de l'instance de base de données d'enregistreur et du réplica en lecture.

Pour plus d'informations sur la configuration du paramètre `apg_write_forward.consistency_mode`, consultez the section called "Paramètres de configuration pour le transfert d'écriture dans Aurora PostgreSQL".

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'allongement des temps d'attente sont les suivantes :

- Augmentation du retard de réplica, tel que mesuré par la métrique `ReplicaLag` d'Amazon CloudWatch. Pour plus d'informations sur cette métrique, consultez Surveillance de la réplication Aurora PostgreSQL.
- Charge accrue sur l'instance de base de données d'enregistreur ou sur le réplica en lecture.

Actions

Modifiez votre mode de cohérence en fonction des besoins de votre application.

IPC:AuroraWriteForwardExecute

L'événement `IPC:AuroraWriteForwardExecute` se produit lorsqu'un processus dorsal sur le réplica en lecture attend qu'une requête transférée se termine et obtienne des résultats du nœud d'enregistreur du cluster de bases de données.

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'augmentation du nombre d'événements d'attente sont les suivantes :

- Un grand nombre de lignes est récupéré à partir du nœud d'enregistreur.

- Une augmentation de la latence du réseau entre le nœud d'enregistreur de et le réplica en lecture augmente le temps nécessaire au réplica en lecture pour recevoir les données du nœud d'enregistreur.
- Une augmentation de la charge sur le réplica en lecture peut retarder la transmission de la demande de la requête entre le réplica en lecture et le nœud d'enregistreur.
- Une augmentation de la charge sur le nœud d'enregistreur peut retarder la transmission des données entre le nœud d'enregistreur et le nœud secondaire.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

- Optimisez les requêtes pour récupérer uniquement les données nécessaires.
- Optimisez les opérations DML (Data Manipulation Language) pour ne modifier que les données nécessaires.
- Si le réplica en lecture ou le nœud d'enregistreur est limité par le processeur ou la bande passante du réseau, envisagez de le remplacer par un type d'instance doté d'une plus grande capacité de processeur ou de bande passante.

IPC:AuroraWriteForwardGetGlobalConsistencyPoint

L'événement `IPC:AuroraWriteForwardGetGlobalConsistencyPoint` se produit lorsqu'un processus dorsal sur le réplica en lecture qui utilise le mode de cohérence GLOBAL attend d'obtenir le point de cohérence global auprès du nœud d'enregistreur avant d'exécuter une requête.

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'augmentation du nombre d'événements d'attente sont les suivantes :

- Une augmentation de la latence du réseau entre le réplica en lecture et le nœud d'enregistreur augmente le temps nécessaire au réplica en lecture pour recevoir les données du nœud d'enregistreur.
- Une augmentation de la charge sur le réplica en lecture peut retarder la transmission de la demande de la requête entre le réplica en lecture et le nœud d'enregistreur.
- Une augmentation de la charge sur le nœud d'enregistreur peut retarder la transmission des données entre le nœud d'enregistreur et le nœud secondaire.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

- Modifiez votre mode de cohérence en fonction des besoins de votre application.
- Si le réplica en lecture ou le nœud d'enregistreur est limité par le processeur ou la bande passante du réseau, envisagez de le remplacer par un type d'instance doté d'une plus grande capacité de processeur ou de bande passante.

IPC:AuroraWriteForwardXactAbort

L'événement `IPC:AuroraWriteForwardXactAbort` se produit lorsqu'un processus dorsal sur le réplica en lecture attend le résultat d'une requête de nettoyage à distance. Des requêtes de nettoyage sont émises pour remettre le processus dans l'état approprié après l'abandon d'une transaction de transfert d'écriture. Amazon Aurora les exécute soit parce qu'une erreur a été détectée, soit parce qu'un utilisateur a émis une commande `ABORT` explicite ou annulé une requête en cours d'exécution.

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'augmentation du nombre d'événements d'attente sont les suivantes :

- Une augmentation de la latence du réseau entre le réplica en lecture et le nœud d'enregistreur augmente le temps nécessaire au réplica en lecture pour recevoir les données du nœud d'enregistreur.
- Une augmentation de la charge sur le réplica en lecture peut retarder la transmission de la demande associée à la requête de nettoyage du réplica en lecture au nœud d'enregistreur.
- Une augmentation de la charge sur le nœud d'enregistreur peut retarder la transmission des données entre le nœud d'enregistreur et le nœud secondaire.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

- Recherchez la cause de l'annulation de la transaction.
- Si le réplica en lecture ou l'instance de base de données d'enregistreur est limité par le processeur ou la bande passante du réseau, envisagez de le remplacer par un type d'instance doté d'une plus grande capacité de processeur ou de bande passante.

IPC:AuroraWriteForwardXactCommit

L'événement `IPC:AuroraWriteForwardXactCommit` se produit lorsqu'un processus dorsal sur le réplica en lecture attend le résultat d'une commande transférée de type commit transaction.

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'augmentation du nombre d'événements d'attente sont les suivantes :

- Une augmentation de la latence du réseau entre le réplica en lecture et le nœud d'enregistreur augmente le temps nécessaire au réplica en lecture pour recevoir les données du nœud d'enregistreur.
- Une augmentation de la charge sur le réplica en lecture peut retarder la transmission de la demande de la requête entre le réplica en lecture et le nœud d'enregistreur.
- Une augmentation de la charge sur le nœud d'enregistreur peut retarder la transmission des données entre le nœud d'enregistreur et le nœud secondaire.

Actions

Si le réplica en lecture ou le nœud d'enregistreur est limité par le processeur ou la bande passante du réseau, envisagez de le remplacer par un type d'instance doté d'une plus grande capacité de processeur ou de bande passante.

IPC:AuroraWriteForwardXactStart

L'événement `IPC:AuroraWriteForwardXactStart` se produit lorsqu'un processus dorsal sur le réplica en lecture attend le résultat d'une commande start transaction transférée.

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'augmentation du nombre d'événements d'attente sont les suivantes :

- Une augmentation de la latence du réseau entre le réplica en lecture et le nœud d'enregistreur augmente le temps nécessaire au réplica en lecture pour recevoir les données du nœud d'enregistreur.
- Une augmentation de la charge sur le réplica en lecture peut retarder la transmission de la demande de la requête entre le réplica en lecture et le nœud d'enregistreur.
- Une augmentation de la charge sur le nœud d'enregistreur peut retarder la transmission des données entre le nœud d'enregistreur et le nœud secondaire.

Actions

Si le réplica en lecture ou le nœud d'enregistreur est limité par le processeur ou la bande passante du réseau, envisagez de le remplacer par un type d'instance doté d'une plus grande capacité de processeur ou de bande passante.

Utilisation d'Aurora PostgreSQL comme base de connaissances pour Amazon Bedrock

Vous pouvez utiliser un cluster de bases de données Aurora PostgreSQL comme base de connaissances pour Amazon Bedrock. Pour plus d'informations, consultez Création d'un stockage vectoriel dans Amazon Aurora. Une base de connaissances prend automatiquement les données de texte non structurées stockées dans un compartiment Amazon S3, les convertit en fragments de texte et en vecteurs, puis les stocke dans une base de données PostgreSQL. Avec les applications d'IA générative, vous pouvez utiliser des agents Amazon Bedrock pour interroger les données stockées dans la base de connaissances et exploiter les résultats de ces requêtes pour enrichir les réponses fournies par les modèles fondamentaux. Ce flux de travail s'appelle « génération à enrichissement contextuel (RAG) ». Pour plus d'informations, consultez Génération à enrichissement contextuel (RAG).

Pour obtenir des informations détaillées sur l'utilisation d'Aurora PostgreSQL pour créer des applications d'IA génératives à l'aide de RAG, consultez ce billet de blog.

Rubriques

- Prérequis
- Préparation d'Aurora PostgreSQL pour l'utiliser comme base de connaissances pour Amazon Bedrock
- Création d'une base de connaissances dans la console Bedrock
- Création rapide d'une base de connaissances Aurora PostgreSQL pour Amazon Bedrock

Prérequis

Familiarisez-vous avec les conditions préalables suivantes pour utiliser le cluster Aurora PostgreSQL comme base de connaissances pour Amazon Bedrock. Globalement, vous devez configurer les services suivants pour une utilisation avec Bedrock :

- Cluster de bases de données Amazon Aurora PostgreSQL créé dans l'une des versions suivantes :
  - 16.1 et toutes les versions ultérieures
  - 15.4 et versions ultérieures
  - 14.9 et versions ultérieures
  - 13.12 et versions ultérieures
  - 12.16 et versions ultérieures

  > **Note**
  >
  > Vous devez activer l'extension `pgvector` dans votre base de données cible et utiliser la version 0.5.0 ou une version ultérieure. Pour plus d'informations, consultez pgvector v0.5.0 avec indexation HNSW.

- API de données RDS
- Utilisateur géré dans AWS Secrets Manager. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager.

Préparation d'Aurora PostgreSQL pour l'utiliser comme base de connaissances pour Amazon Bedrock

Suivez les étapes décrites dans les sections ci-dessous afin de préparer Aurora PostgreSQL pour l'utiliser comme base de connaissances pour Amazon Bedrock.

Création et configuration d'Aurora PostgreSQL

Pour configurer Amazon Bedrock avec un cluster de bases de données Aurora PostgreSQL, vous devez d'abord créer un cluster de bases de données Aurora PostgreSQL et prendre note des champs importants pour pouvoir le configurer avec Amazon Bedrock. Pour plus d'informations sur la création d'un cluster de bases de données Aurora PostgreSQL, consultez Création et connexion à un cluster de bases de données Aurora PostgreSQL.

- Activez l'API de données lors de la création du cluster de bases de données Aurora PostgreSQL. Pour plus d'informations sur les versions prises en charge, consultez Utilisation de l'API de données Amazon RDS.
- Assurez-vous de noter les Amazon Resource Names (ARN) de votre cluster de bases de données Aurora PostgreSQL. Vous en aurez besoin pour configurer le cluster de bases de données à

utiliser avec Amazon Bedrock. Pour plus d'informations, consultez Amazon Resource Names (ARN).

Connexion à une base de données et installation de pgvector

Vous pouvez vous connecter à Aurora PostgreSQL à l'aide de n'importe quel utilitaire de connexion. Pour des informations plus détaillées sur ces utilitaires, consultez Connexion à un cluster de bases de données Amazon Aurora PostgreSQL. Vous pouvez également utiliser l'éditeur de requêtes de la console RDS pour exécuter les requêtes. Pour pouvoir utiliser cet éditeur, vous avez besoin d'un cluster de bases de données Aurora au niveau duquel l'API de données RDS est activée.

1. Connectez-vous à la base de données avec votre compte d'utilisateur principal et configurez pgvector. Utilisez la commande suivante si l'extension n'est pas installée :

   ```
   CREATE EXTENSION IF NOT EXISTS vector;
   ```

   Utilisez `pgvector` 0.5.0, qui prend en charge l'indexation HNSW, ou une version ultérieure. Pour plus d'informations, consultez pgvector v0.5.0 avec indexation HNSW.

2. Pour vérifier la version de `pg_vector` installée, exécutez la commande suivante :

   ```
   SELECT extversion FROM pg_extension WHERE extname='vector';
   ```

Configuration des objets et des privilèges de base de données

1. Créez un schéma spécifique que Bedrock pourra utiliser pour interroger les données. Utilisez la commande suivante pour créer un schéma :

   ```
   CREATE SCHEMA bedrock_integration;
   ```

2. Créez un rôle que Bedrock pourra utiliser pour interroger la base de données. Utilisez la commande suivante pour créer un rôle :

   ```
   CREATE ROLE bedrock_user WITH PASSWORD 'password' LOGIN;
   ```

> **Note**
>
> Prenez note de ce mot de passe, car vous en aurez besoin ultérieurement pour créer un mot de passe Secrets Manager.

Sur un client `psql`, utilisez les commandes suivantes pour créer un rôle :

```
CREATE ROLE bedrock_user LOGIN;
\PASSWORD password;
```

3. Accordez les autorisations `bedrock_user` nécessaires pour gérer le schéma `bedrock_integration`. Cela permettra de créer des tables ou des index dans le schéma.

```
GRANT ALL ON SCHEMA bedrock_integration to bedrock_user;
```

4. Connectez-vous en tant que `bedrock_user` et créez une table dans le `bedrock_integration schema`.

```
CREATE TABLE bedrock_integration.bedrock_kb (id uuid PRIMARY KEY, embedding
 vector(n), chunks text, metadata json, custom_metadata jsonb);
```

Cette commande crée la table `bedrock_kb` dans le schéma `bedrock_integration` avec les vectorisations Titan.

Remplacez n dans le type de données `vector(n)` par la dimension appropriée pour le modèle de vectorisation que vous utilisez. Reportez-vous aux recommandations ci-dessous pour sélectionner les dimensions appropriées :

- Pour le modèle Titan v2, utilisez `vector(1024)`, `vector(512)` ou `vector (256)`. Pour en savoir plus, consultez Texte des vectorisations Amazon Titan.
- Pour le modèle Titan v1.2, utilisez `vector(1536)`. Pour en savoir plus, consultez Plongement multimodal Amazon Titan G1.
- Pour le modèle Cohere Embed, utilisez `vector(1024)`. Pour en savoir plus, consultez Modèles Cohere Embed.
- Pour le modèle Cohere Embed Multilingual v3, utilisez `vector(1024)`.

Les quatre premières colonnes sont obligatoires. Pour le traitement des métadonnées, Bedrock écrit les données tirées de vos fichiers de métadonnées dans la colonne `custom_metadata`. Nous vous recommandons de créer cette colonne si vous prévoyez d'utiliser les métadonnées et le filtrage. Si vous ne créez pas de colonne `custom_metadata`, ajoutez des colonnes individuelles pour chaque attribut de métadonnées de votre table avant de commencer l'ingestion. Pour plus d'informations, consultez Configuration et personnalisation des requêtes et de la génération des réponses.

5. Suivez ces étapes pour créer les index requis que Bedrock utilisera pour interroger vos données :

   - Créez un index avec l'opérateur cosinus que Bedrock pourra utiliser pour interroger les données.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING hnsw (embedding
      vector_cosine_ops);
     ```

   - Nous vous recommandons de définir la valeur d'`ef_construction` sur 256 pour `pgvector` 0.6.0 et les versions ultérieures qui utilisent la création d'index en parallèle.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING hnsw (embedding
      vector_cosine_ops) WITH (ef_construction=256);
     ```

   - Créez un index que Bedrock pourra utiliser pour interroger les données textuelles.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING gin (to_tsvector('simple',
      chunks));
     ```

   - Si vous avez créé une colonne pour les métadonnées personnalisées, créez un index que Bedrock pourra utiliser pour interroger les métadonnées.

     ```
     CREATE INDEX ON bedrock_integration.bedrock_kb USING gin (custom_metadata);
     ```

Création d'un secret dans Secrets Manager

Secrets Manager vous permet de stocker vos informations d'identification Aurora afin qu'elles puissent être transmises en toute sécurité aux applications. Si vous n'avez pas choisi l'option AWS Secrets Manager lors de la création du cluster de bases de données Aurora PostgreSQL, vous

pouvez créer un secret maintenant. Pour plus d'informations sur la création d'un secret de base de données AWS Secrets Manager, consultez Secret de base de données AWS Secrets Manager.

Création d'une base de connaissances dans la console Bedrock

Lors de la préparation d'Aurora PostgreSQL afin de pouvoir l'utiliser comme stockage vectoriel pour une base de connaissances, recueillez les informations suivantes que vous devrez fournir à la console Amazon Bedrock.

- ARN du cluster de bases de données Amazon Aurora : ARN de votre cluster de bases de données.
- ARN du secret : ARN de la clé AWS Secrets Manager correspondant à votre cluster de bases de données.
- Nom de base de données : nom de votre base de données. Par exemple, vous pouvez utiliser la base de données par défaut *`postgres`*.
- Nom de la table : nous vous recommandons de fournir un nom qualifié de schéma lors de la création de la table à l'aide d'une commande similaire à la suivante :

  ```
  CREATE TABLE bedrock_integration.bedrock_kb;
  ```

  Cette commande crée la table `bedrock_kb` dans le schéma `bedrock_integration`.
- Lorsque vous créez la table, assurez-vous de la configurer avec les colonnes et les types de données spécifiés. Vous pouvez utiliser des noms de colonne de votre choix au lieu de ceux répertoriés dans le tableau. N'oubliez pas de prendre note des noms que vous avez choisis pour référence lors de la configuration de la base de connaissances.

Nom de la colonne	Type de données	Description
id	Clé primaire UUID	Contient des identifia nts uniques pour chaque enregistrement.
chunks	Texte	Contient les segments de texte brut provenant de vos sources de données.

Nom de la colonne	Type de données	Description
embedding	Vecteur	Contient les intégrations vectorielles des sources de données.
metadata	JSON	Contient les métadonnées nécessaires pour effectuer l'attribution de la source et pour permettre l'ingestion et l'interrogation des données.
custom_metadata	JSONB	(Facultatif) Définit la colonne cible dans laquelle Amazon Bedrock écrit les détails des métadonnées à partir de vos sources de données.

Avec ces informations, vous pouvez désormais créer une base de connaissances dans la console Bedrock. Pour des informations plus détaillées sur la configuration d'un index vectoriel et la création d'une base de connaissances, consultez Création d'un stockage vectoriel dans Amazon Aurora et Création d'une base de connaissances dans Amazon Aurora.

Après avoir ajouté Aurora comme base de connaissances, vous pouvez désormais ingérer vos sources de données pour effectuer des recherches et des requêtes. Pour plus d'informations, consultez Ingestion de vos sources de données dans la base de connaissances.

Création rapide d'une base de connaissances Aurora PostgreSQL pour Amazon Bedrock

Le flux de travail de génération à enrichissement contextuel (RAG) d'Amazon Bedrock repose sur des données vectorielles stockées dans une base de données Aurora PostgreSQL pour optimiser l'extraction de contenu. Auparavant, la configuration d'Aurora PostgreSQL en tant que stockage de données vectorielles pour les bases de connaissances Bedrock était un processus en plusieurs étapes qui nécessitait de nombreuses actions manuelles sur différentes interfaces utilisateur. Il était donc difficile pour les scientifiques des données et les développeurs de tirer parti d'Aurora pour leurs projets Bedrock.

Pour améliorer l'expérience utilisateur, AWS a conçu une option de création rapide basée sur CloudFormation, qui simplifie le processus de configuration. Avec cette option de création rapide Aurora, vous pouvez désormais provisionner en un seul clic un cluster de bases de données Aurora PostgreSQL préconfiguré comme stockage vectoriel pour vos bases de connaissances Amazon Bedrock.

Rubriques

Régions et versions d'Aurora PostgreSQL prises en charge

L'option de création rapide Aurora est disponible dans toutes les régions AWS qui prennent en charge les bases de connaissances Amazon Bedrock. Par défaut, elle crée un cluster de bases de données Aurora PostgreSQL avec la version 15.7. Pour plus d'informations sur les régions prises en charge, consultez Modèles et régions pris en charge pour les bases de connaissances Amazon Bedrock.

Comprendre le processus de création rapide

Le processus de création rapide fournit automatiquement les ressources suivantes pour configurer une base de données Amazon Aurora PostgreSQL en tant que stockage de données vectorielles pour votre base de connaissances Bedrock :

Un cluster de bases de données Aurora PostgreSQL dans votre compte, configuré avec les paramètres par défaut.

- Les unités de capacité Aurora (ACU) sont définies entre 0 et 16. Votre stockage vectoriel peut ainsi être réduit verticalement à zéro lorsqu'il n'est pas utilisé, ce qui permet d'économiser sur les coûts de calcul. Les ACU peuvent être ajustées ultérieurement dans la console Amazon RDS.
- Index HNSW (Hierarchical Navigable Small World) utilisant la distance euclidienne comme mesure de similarité pour les vectorisations Bedrock stockées dans Aurora.
- L'instance de base de données est une instance v2 sans serveur.
- Le cluster est associé au VPC et aux sous-réseaux par défaut, et l'API de données RDS est activée.

- Les informations d’identification de l’administrateur du cluster sont gérées par AWS Secrets Manager.

Outre les paramètres par défaut, les paramètres suivants sont définis pour vous. Au cours du processus, vous verrez des écrans expliquant le flux de travail.

- Alimentation du cluster Aurora avec les objets de base de données nécessaires :
  - Créez l’extension pgvector, le schéma, le rôle et les tables nécessaires pour la base de connaissances Bedrock.
  - Inscrivez un utilisateur de base de données à privilèges limités pour que Bedrock interagisse avec le cluster.
- Une bannière de progression vous permettant de suivre l’état des sous-événements suivants s’affiche tout au long du processus de mise en service des ressources :
  - Création du cluster Aurora
  - Alimentation du cluster Aurora
  - Création d’une base de connaissances

  Cette bannière reste visible jusqu’à ce que la base de connaissances soit entièrement créée, même si vous quittez la page et que vous y revenez.
- Vous pouvez cliquer sur `View details` sur la bannière de progression pour voir le statut de chaque étape. Pour plus d’informations sur les événements survenus lors de la création de la base de connaissances, cliquez sur le lien CloudFormation figurant sur l’écran d’affichage des détails. Une fois le processus terminé, votre nouvelle base de connaissances Bedrock est prête à être utilisée.
- Les identifiants de pile pour toutes les ressources de création rapide se trouvent dans les balises de la base de connaissances Bedrock, au cas où vous auriez besoin de les référencer.

Une base de connaissances Bedrock, avec la configuration du cluster Aurora nouvellement provisionné lors de la création du stockage vectoriel.

Avantages offerts par l’utilisation de l’option de création rapide Aurora

- Le processus de création rapide basé sur CloudFormation réduit considérablement le temps et la complexité nécessaires à l’utilisation d’Aurora comme stockage vectoriel.

- Aurora offre d’excellentes performances, une capacité de mise à l’échelle vectorielle et des avantages en termes de coûts avec la possibilité de réduire à zéro les frais de calcul lorsqu’il n’est pas utilisé.
- Le processus de création rapide rationalise l’expérience de bout en bout, ce qui vous permet de créer et de configurer facilement des bases de connaissances Bedrock à l’aide d’Aurora.
- Les clients peuvent s’appuyer sur le modèle CloudFormation pour personnaliser le provisionnement avec leurs propres configurations.

Limites du processus de création rapide Aurora

- Avec l’option de création rapide Aurora, le cluster de bases de données est provisionné avec les configurations par défaut. Toutefois, il est possible que ces paramètres par défaut ne répondent pas à vos exigences spécifiques ou au cas d’utilisation prévu. La création rapide ne propose aucune option permettant de modifier les configurations pendant le processus de provisionnement. Elles sont définies automatiquement pour rationaliser l’expérience de déploiement. Si vous devez personnaliser la configuration du cluster de bases de données Aurora, procédez après le déploiement initial par création rapide dans la console Amazon RDS.
- L’option de création rapide simplifie le processus de configuration, mais le délai de création du cluster de bases de données Aurora est toujours d’une dizaine de minutes, comme dans le cas d’un déploiement manuel. Cela s’explique par le délai nécessaire pour provisionner l’infrastructure Aurora.
- L’option de création rapide est conçue pour l’expérimentation et la configuration rapide. Les ressources créées par le biais de la création rapide peuvent ne pas être adaptées à une utilisation en production, et vous ne pouvez pas les migrer directement vers un environnement de production dans votre VPC.

Intégration d’Amazon Aurora PostgreSQL avec d’autres services AWS

Amazon Aurora s’intègre avec d’autres services AWS pour vous permettre d’étendre votre cluster de bases de données Aurora PostgreSQL afin d’utiliser des fonctionnalités supplémentaires dans le cloud AWS. Votre cluster de bases de données Aurora PostgreSQL peut utiliser des services AWS pour effectuer les opérations suivantes :

- Collecter, afficher et évaluer rapidement les performances de vos instances de base de données Aurora PostgreSQL avec Amazon RDS Performance Insights. Performance Insights complète les fonctions de surveillance existantes d'Amazon RDS. Ce service illustre les performances de votre base de données et facilite votre analyse des problèmes qui les impactent. Grâce au tableau de bord de Performance Insights, vous pouvez visualiser la charge de la base de données et la filtrer par attentes, instructions SQL, hôtes ou utilisateurs. Pour plus d'informations sur Performance Insights, consultez Surveillance de la charge de la base de données avec Performance Insights sur .
- Configurez votre cluster de bases de données Aurora PostgreSQL de façon à publier des données de journaux dans Amazon CloudWatch Logs. CloudWatch Logs fournit un stockage hautement durable pour vos enregistrements de journaux. CloudWatch Logs vous permet d'effectuer une analyse en temps réel des données de journaux et d'utiliser CloudWatch pour créer des alarmes et afficher des métriques. Pour plus d'informations, consultez Publication de journaux Aurora PostgreSQL sur Amazon CloudWatch Logs.
- Importez des données à partir d'un compartiment Amazon S3 vers un cluster de bases de données Aurora PostgreSQL ou exportez des données à partir d'un cluster de bases de données Aurora PostgreSQL vers un compartiment Amazon S3. Pour plus d'informations, consultez Importation de données Amazon S3 dans une d'un cluster de base de données Aurora PostgreSQL et Exportation de données à partir d'un cluster de bases de données Aurora PostgreSQL vers Amazon S3.
- Ajoutez des prédictions basées sur le machine learning à des applications de base de données à l'aide du langage SQL. Le machine learning Aurora utilise une intégration hautement optimisée entre la base de données Aurora et les services de machine learning (ML) AWS SageMaker AI et Amazon Comprehend. Pour plus d'informations, consultez Utilisation du machine learning Amazon Aurora avec Aurora PostgreSQL.
- Appelez des fonctions AWS Lambda à partir d'un cluster de bases de données Aurora PostgreSQL. Pour ce faire, utilisez l'extension PostgreSQL `aws_lambda` fournie avec Aurora PostgreSQL. Pour plus d'informations, consultez Invocation d'une AWS Lambda fonction depuis une instance de base de données Aurora PostgreSQL pour PostgreSQL.
- Intégrez les requêtes d'Amazon Redshift et Aurora PostgreSQL. Pour plus d'informations, consultez Commencer à utiliser les requêtes fédérées dans PostgreSQL dans le Guide du développeur de base de données Amazon Redshift.

Importation de données Amazon S3 dans une d'un cluster de base de données Aurora PostgreSQL

Vous pouvez importer des données qui ont été stockées à l'aide d'Amazon Simple Storage Service dans une table sur une instance de cluster de base de données Aurora PostgreSQL. Pour ce faire, vous devez d'abord installer l'extension `aws_s3` Aurora PostgreSQL . Cette extension fournit les fonctions que vous utilisez pour importer des données à partir d'un compartiment Amazon S3. Un compartiment est un conteneur Amazon S3 pour les objets et les fichiers. Les données peuvent résider dans un fichier CSV (valeur séparée par des virgules), un fichier texte ou un fichier compressé (gzip). Vous apprendrez ensuite comment installer l'extension et comment importer des données d'Amazon S3 dans un tableau.

Votre base de données doit exécuter PostgreSQL version 10.7 ou supérieure pour importer depuis Simple Storage Service (Amazon S3) vers . Aurora PostgreSQL.

Si vous n'avez pas de données stockées sur Amazon S3, vous devez d'abord créer un compartiment et y stocker les données. Pour en savoir plus, consulter les rubriques suivantes dans le Guide de l'utilisateur d'Amazon Simple Storage Service.

- Créez un compartiment
- Ajout d'un objet dans un compartiment

L'importation entre comptes depuis Amazon S3 est prise en charge. Pour plus d’informations, consultez Octroi d'autorisations entre comptes dans le Guide de l'utilisateur Amazon Simple Storage Service.

Vous pouvez utiliser la clé gérée par le client pour le chiffrement lors de l'importation de données depuis S3. Pour plus d'informations, consultez Clés KMS stockées dans AWS KMS dans le Guide de l'utilisateur Amazon Simple Storage Service.

> **Note**
>
> L'importation des données à partir d'Amazon S3 n'est pas prise en charge pour Aurora Serverless v1. Elle est prise en charge pour Aurora Serverless v2.

Rubriques

Installation de l'extension aws_s3

Avant de pouvoir utiliser Amazon S3 avec votre cluster de base de données Aurora PostgreSQL , vous devez installer l'extension. Cette extension fournit des fonctions pour importer des données depuis un compartiment Amazon S3. Il fournit également des fonctions pour exporter des données depuis une instance d'un cluster de base de données Aurora PostgreSQL vers un compartiment Amazon S3. Pour de plus amples informations, consultez Exportation de données à partir d’un cluster de bases de données Aurora PostgreSQL vers Amazon S3. L'extension `aws_s3` dépend de certaines des fonctions d'aide de l'extension `aws_commons`, qui est installée automatiquement lorsque cela est nécessaire.

Pour installer l'extension **`aws_s3`**

1. Utilisez psql (ou pgAdmin) pour vous connecter à l'instance de base de données en écriture de votre cluster de base de données Aurora PostgreSQL en tant qu'utilisateur disposant de privilèges `rds_superuser`. Si vous avez conservé le nom par défaut pendant le processus d'installation, vous vous connectez en tant que `postgres`.

   ```
   psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Pour installer l'extension, exécutez la commande suivante.

   ```
   postgres=> CREATE EXTENSION aws_s3 CASCADE;
   NOTICE: installing required extension "aws_commons"
   CREATE EXTENSION
   ```

3. Pour vérifier que l'extension est installée, vous pouvez utiliser la métacommande psql `\dx`.

   ```
   postgres=> \dx
          List of installed extensions
       Name     | Version |   Schema   |                Description
   ```

```
-------------+---------+------------+--------------------------------------------
 aws_commons | 1.2     | public     | Common data types across AWS services
 aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
 plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

Les fonctions d'importation de données depuis Amazon S3 et d'exportation de données vers Amazon S3 sont désormais disponibles.

Présentation de l’importation de données à partir de données Amazon S3

Pour importer des données S3 dans Aurora PostgreSQL

Tout d’abord, rassemblez les informations que vous devez fournir à la fonction. Il s’agit notamment du nom de la table sur votre instance de votre cluster de bases de données Aurora PostgreSQL, ainsi que du nom du compartiment, du chemin d’accès au fichier, du type de fichier et de la Région AWS où sont stockées les données Amazon S3. Pour plus d’informations, consultez View an object (Afficher un objet) dans le Guide de l’utilisateur Amazon Simple Storage Service.

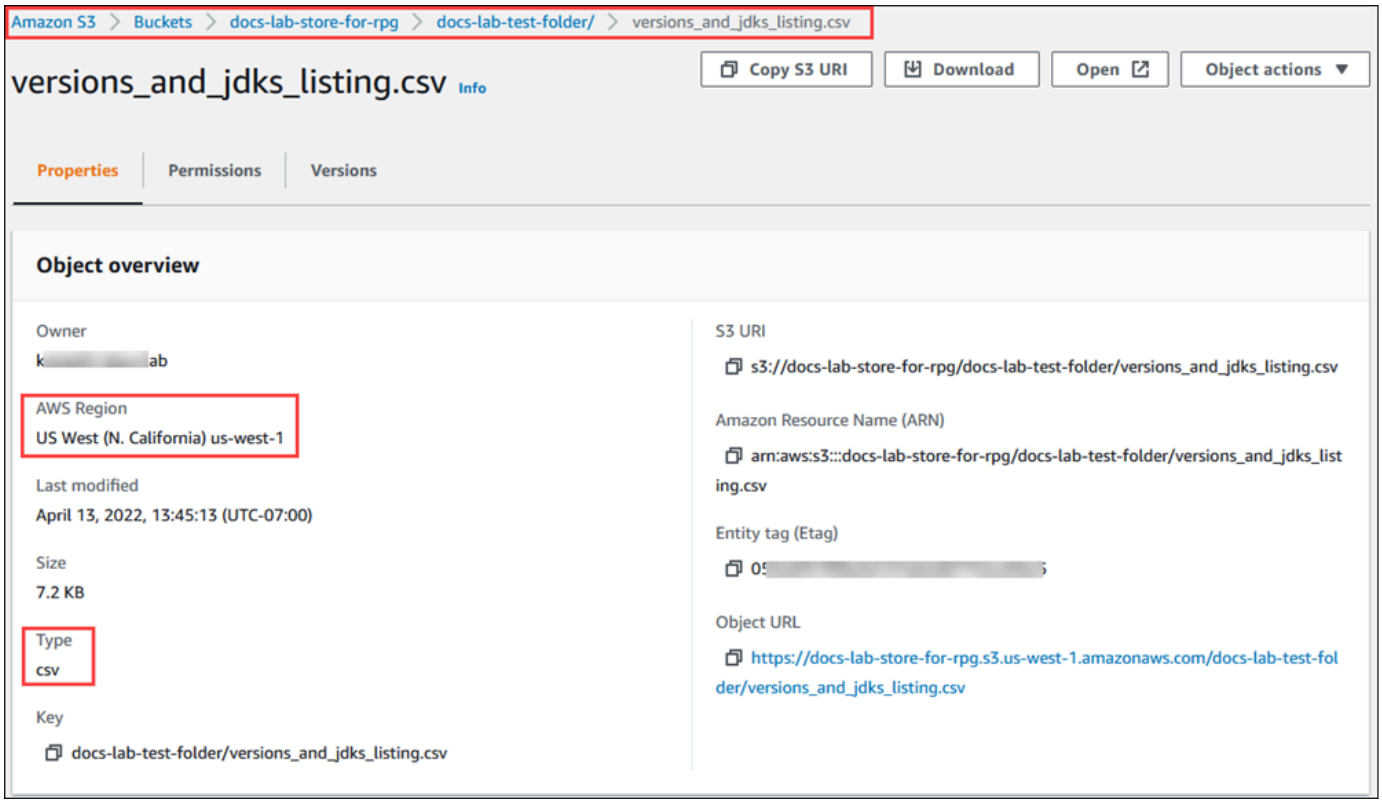
> **Note**
>
> L’importation de données partitionnées depuis Amazon S3 n’est pas prise en charge actuellement.

1. Obtenez le nom de la table dans laquelle la fonction `aws_s3.table_import_from_s3` doit importer les données. À titre d’exemple, la commande suivante crée une table `t1` qui peut être utilisée dans les étapes suivantes.

   ```
   postgres=> CREATE TABLE t1
       (col1 varchar(80),
       col2 varchar(80),
       col3 varchar(80));
   ```

2. Obtenez les détails sur le compartiment Amazon S3 et les données à importer. Pour ce faire, ouvrez la console Amazon S3 à l’adresse https://console.aws.amazon.com/s3/, et choisissez Buckets (Compartiments). Trouvez le compartiment contenant vos données dans la liste. Sélectionnez le compartiment, ouvrez sa page Object overview (Présentation des objets), puis choisissez Properties (Propriétés).

Notez le nom du compartiment, le chemin, la Région AWS, et le type de fichier. Vous aurez besoin du nom Amazon Resource Name (ARN) pour configurer l'accès à Amazon S3 via un rôle IAM. Pour obtenir plus d'informations, consultez Configuration de l'accès à un compartiment Amazon S3. L'image suivante montre un exemple.



3. Vous pouvez vérifier le chemin d'accès aux données sur le compartiment Amazon S3 en utilisant la commande AWS CLI `aws s3 cp`. Si les informations sont correctes, cette commande télécharge une copie du fichier Amazon S3.

```
aws s3 cp s3://amzn-s3-demo-bucket/sample_file_path ./
```

4. Configurez les autorisations sur votre cluster de bases de données Aurora PostgreSQL pour permettre l'accès au fichier sur le compartiment Amazon S3. Pour cela, utilisez des informations d'identification de sécurité ou un rôle Gestion des identités et des accès AWS (IAM). Pour plus d'informations, consultez Configuration de l'accès à un compartiment Amazon S3.

5. Fournissez le chemin et les autres détails de l'objet Amazon S3 recueillis (voir l'étape 2) à la fonction `create_s3_uri` pour construire un objet URI Amazon S3. Pour en savoir plus sur cette fonction, consultez aws_commons.create_s3_uri. Voici un exemple de construction de cet objet pendant une session psql.

```
postgres=> SELECT aws_commons.create_s3_uri(
   'docs-lab-store-for-rpg',
   'versions_and_jdks_listing.csv',
   'us-west-1'
) AS s3_uri \gset
```

Dans l'étape suivante, vous transmettez cet objet (`aws_commons._s3_uri_1`) à la fonction `aws_s3.table_import_from_s3` pour importer les données dans la table.

6. Appelez la fonction `aws_s3.table_import_from_s3` pour importer les données d'Amazon S3 dans votre table. Pour obtenir des informations de référence, consultez aws_s3.table_import_from_s3. Pour obtenir des exemples, consultez Importation de données d'Amazon S3 vers votre cluster de base de données Aurora PostgreSQL.

Configuration de l'accès à un compartiment Amazon S3

Pour importer des données à partir d'un fichier Amazon S3, vous devez accorder au cluster de bases de données Aurora PostgreSQL une autorisation d'accès au compartiment Amazon S3 contenant le fichier. Pour accorder l'accès à un compartiment Amazon S3, vous pouvez employer une des deux méthodes décrites dans les rubriques suivantes.

Rubriques

- Utilisation d'un rôle IAM pour accéder à un compartiment Amazon S3
- Utilisation d'informations d'identification de sécurité pour accéder à un compartiment Amazon S3
- Résolution des problèmes d'accès à Amazon S3

Utilisation d'un rôle IAM pour accéder à un compartiment Amazon S3

Avant de charger des données à partir d'un fichier Amazon S3, accordez à votre cluster de bases de données Aurora PostgreSQL l'autorisation d'accéder au compartiment Amazon S3 dans lequel se trouve le fichier. De cette façon, vous n'avez pas à gérer d'informations d'identification supplémentaires ni à les fournir dans l'appel de fonction aws_s3.table_import_from_s3.

Pour ce faire, créez une politique IAM qui donne accès au compartiment Amazon S3. Créez un rôle IAM et attachez la politique à ce rôle. Attribuez ensuite le rôle IAM à votre cluster de base de données.

> **Note**
>
> Vous ne pouvez pas associer un rôle IAM à un cluster de bases de données Aurora Serverless v1, de sorte que les étapes suivantes ne s'appliquent pas.

Pour permettre à un cluster de bases de données Aurora PostgreSQL d'accéder à Amazon S3 via un rôle IAM

1. Créez une politique IAM.

   Celle-ci fournit au compartiment et à l'objet les autorisations permettant à votre cluster de bases de données Aurora PostgreSQL d'accéder à Amazon S3.

   Incluez à la politique les actions obligatoires suivantes pour permettre le transfert de fichiers d'un compartiment Amazon S3 vers Aurora PostgreSQL :

   - `s3:GetObject`
   - `s3:ListBucket`

   Incluez à la politique les ressources suivantes pour identifier le compartiment Amazon S3 et les objets qu'il contient. Voici le format Amazon Resource Name (ARN) permettant d'accéder à Amazon S3 :

   - arn:aws:s3 : *`amzn-s3-demo-bucket`*
   - arn:aws:s3 : :1 /* *`amzn-s3-demo-bucket`*

   Pour obtenir plus d'informations sur la création d'une politique IAM pour Aurora PostgreSQL, consultez Création et utilisation d'une politique IAM pour l'accès à une base de données IAM. Consultez également Didacticiel : création et attachement de votre première politique gérée par le client dans le Guide de l'utilisateur IAM.

   La AWS CLI commande suivante crée une politique IAM nommée `rds-s3-import-policy` avec ces options. Elle accorde un accès à un compartiment nommé *`amzn-s3-demo-bucket`*.

> **Note**
>
> Notez le Amazon Resource Name (ARN) de la politique renvoyée par cette commande. Vous en aurez besoin par la suite pour attacher la politique à un rôle IAM.

Example

Pour Linux, macOS ou Unix :

```
aws iam create-policy \
   --policy-name rds-s3-import-policy \
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
           "s3:ListBucket"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket",
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

Pour Windows :

```
aws iam create-policy ^
   --policy-name rds-s3-import-policy ^
   --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3import",
         "Action": [
           "s3:GetObject",
```

```
            "s3:ListBucket"
          ],
          "Effect": "Allow",
          "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
          ]
        }
      ]
    }'
```

2. Créez un rôle IAM.

   L'objectif est ici de permettre à Aurora PostgreSQL d'endosser ce rôle IAM pour accéder à vos compartiments Amazon S3. Pour plus d'informations, consultez Création d'un rôle pour déléguer des autorisations à un utilisateur IAM dans le Guide de l'utilisateur IAM.

   Nous vous recommandons d'utiliser les clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount` dans des politiques basées sur les ressources pour limiter les autorisations du service à une ressource spécifique. C'est le moyen le plus efficace de se protéger contre le problème du député confus.

   Si vous utilisez les deux clés de contexte de condition globale et que la valeur de `aws:SourceArn` contient l'ID de compte, la valeur de `aws:SourceAccount` et le compte indiqué dans la valeur de `aws:SourceArn` doivent utiliser le même ID de compte lorsqu'il est utilisé dans la même déclaration de politique.

   - Utilisez `aws:SourceArn` si vous souhaitez un accès interservices pour une seule ressource.
   - Utilisez `aws:SourceAccount` si vous souhaitez autoriser une ressource de ce compte à être associée à l'utilisation interservices.

   Dans la politique, veillez à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource. L'exemple suivant montre comment procéder à l'aide de la AWS CLI commande pour créer un rôle nommé`rds-s3-import-role`.

   Example

   Pour Linux, macOS ou Unix :

```
aws iam create-role \
```

```
   --role-name rds-s3-import-role \
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:clustername"
                }
             }
       }
     ]
   }'
```

Pour Windows :

```
aws iam create-role ^
   --role-name rds-s3-import-role ^
   --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:clustername"
                }
             }
       }
     ]
```

```
    }'
```

3. Attachez la politique IAM que vous avez créée au rôle IAM que vous venez de créer.

   La AWS CLI commande suivante associe la politique créée à l'étape précédente au rôle nommé `rds-s3-import-role` Remplacer *`your-policy-arn`* par l'ARN de stratégie que vous avez noté à l'étape précédente.

   Example

   Pour Linux, macOS ou Unix :

```
aws iam attach-role-policy \
   --policy-arn your-policy-arn \
   --role-name rds-s3-import-role
```

   Pour Windows :

```
aws iam attach-role-policy ^
   --policy-arn your-policy-arn ^
   --role-name rds-s3-import-role
```

4. Ajoutez le rôle IAM au cluster de base de données.

   Pour ce faire, utilisez le AWS Management Console ou AWS CLI, comme décrit ci-dessous.

Console

Pour ajouter un rôle IAM au cluster de base de données PostgreSQL à l’aide de la console

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Choisissez le nom du cluster de base de données PostgreSQL pour afficher ses détails.
3. Sous l’onglet Connectivity & security (Connectivité et sécurité), accédez à la section Manage IAM roles (Gérer les rôles IAM) et choisissez le rôle à ajouter sous Add IAM roles to this cluster (Ajouter des rôles IAM à ce cluster/cette instance).
4. Sous Feature (Fonction), choisissez s3Import.
5. Choisissez Add role (Ajouter un rôle).

AWS CLI

Pour ajouter un rôle IAM à un cluster de bases de données PostgreSQL à l’aide de CLI

- Utilisez la commande suivante pour ajouter le rôle au cluster de bases de données PostgreSQL nommé `my-db-cluster`. Remplacez *`your-role-arn`* par l’ARN de rôle que vous avez noté lors d’une étape précédente. Utilisez `s3Import` comme valeur de l’option `--feature-name`.

  Example

  Pour Linux, macOS ou Unix :

```
aws rds add-role-to-db-cluster \
   --db-cluster-identifier my-db-cluster \
   --feature-name s3Import \
   --role-arn your-role-arn   \
   --region your-region
```

  Pour Windows :

```
aws rds add-role-to-db-cluster ^
   --db-cluster-identifier my-db-cluster ^
   --feature-name s3Import ^
   --role-arn your-role-arn ^
   --region your-region
```

API RDS

Pour ajouter un rôle IAM pour une de cluster de base de données PostgreSQL à l'aide de l'API Amazon RDS, appelez l'opération. AddRoleToDBCluster

Utilisation d’informations d’identification de sécurité pour accéder à un compartiment Amazon S3

Si vous préférez, au lieu de donner à accès un compartiment Amazon S3 avec un rôle IAM, vous pouvez utiliser des informations d’identification de sécurité. Pour ce faire, spécifiez le paramètre `credentials` dans l’appel de fonction aws_s3.table_import_from_s3.

Le `credentials` paramètre est une structure de type contenant `aws_commons._aws_credentials_1` des AWS informations d'identification. Utilisez la fonction

aws_commons.create_aws_credentials pour définir la clé d'accès et la clé secrète dans une structure `aws_commons._aws_credentials_1`, comme indiqué ci-après.

```
postgres=> SELECT aws_commons.create_aws_credentials(
   'sample_access_key', 'sample_secret_key', '')
AS creds \gset
```

Après avoir créé la structure `aws_commons._aws_credentials_1` , utilisez la fonction aws_s3.table_import_from_s3 avec le paramètre `credentials` pour importer les données, comme indiqué ci-après.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   :'creds'
);
```

Vous pouvez également inclure l'appel de fonction aws_commons.create_aws_credentials en ligne au sein de l'appel de fonction `aws_s3.table_import_from_s3`.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't', '', '(format csv)',
   :'s3_uri',
   aws_commons.create_aws_credentials('sample_access_key', 'sample_secret_key', '')
);
```

Résolution des problèmes d'accès à Amazon S3

Si vous rencontrez des problèmes de connexion lorsque vous tentez d'importer des données depuis Amazon S3, consultez les recommandations suivantes :

- Résolution des problèmes liés à Identity and Access Amazon Aurora
- Dépannage d'Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.
- Dépannage d'Amazon S3 et IAM dans le Guide de l'utilisateur IAM

Importation de données d'Amazon S3 vers votre cluster de base de données Aurora PostgreSQL

Vous importez des données depuis votre compartiment Amazon S3 en utilisant la fonction `table_import_from_s3` de l'extension aws_s3. Pour obtenir des informations de référence, consultez aws_s3.table_import_from_s3.

> **Note**
>
> Les exemples suivants utilisent la méthode du rôle IAM pour donner accès au compartiment Amazon S3. Les appels de fonction `aws_s3.table_import_from_s3` n'incluent donc aucun paramètre d'informations d'identification.

L'exemple suivant montre un exemple typique.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   't1',
   '',
   '(format csv)',
   :'s3_uri'
);
```

Les paramètres sont les suivants :

- `t1` – Nom de la table du cluster de base de base de données PostgreSQL dans laquelle copier les données.
- `''` – Liste facultative des colonnes de la table de base de données. Vous pouvez utiliser ce paramètre pour indiquer quelles colonnes des données S3 sont copiées dans quelles colonnes de table. Si aucune colonne n'est spécifiée, toutes les colonnes sont copiées dans la table. Pour obtenir un exemple d'utilisation d'une liste de colonnes, veuillez consulter Importation d'un fichier Amazon S3 qui utilise un délimiteur personnalisé.
- `(format csv)` – Arguments PostgreSQL COPY. Le processus de copie utilise les arguments et le format de la commande PostgreSQL COPY pour importer les données. Les choix de format comprennent les valeurs séparées par des virgules (CSV) comme dans cet exemple, le texte et les données binaires. Par défaut, il s'agit de texte.

- `s3_uri` – Structure contenant les informations d'identification du fichier Amazon S3. Pour obtenir un exemple d'utilisation de la fonction aws_commons.create_s3_uri pour créer une structure `s3_uri`, consultez Présentation de l'importation de données à partir de données Amazon S3.

Pour de plus amples informations sur cette fonction, veuillez consulter aws_s3.table_import_from_s3.

La fonction `aws_s3.table_import_from_s3` retourne du texte. Pour spécifier d'autres types de fichiers à importer à partir d'un compartiment Amazon S3, consultez l'un des exemples suivants.

> **Note**
>
> L'importation d'un fichier de 0 octet entraîne une erreur.

Rubriques

- Importation d'un fichier Amazon S3 qui utilise un délimiteur personnalisé
- Importation d'un fichier compressé Amazon S3 (gzip)
- Importation d'un fichier codé Amazon S3

Importation d'un fichier Amazon S3 qui utilise un délimiteur personnalisé

L'exemple suivant montre comment importer un fichier qui utilise un délimiteur personnalisé. Il montre également comment définir l'emplacement de destination des données dans la table de base de données à l'aide du paramètre `column_list` de la fonction aws_s3.table_import_from_s3.

Pour cet exemple, supposons que les informations suivantes sont organisées en colonnes délimitées par une barre verticale dans le fichier Amazon S3.

```
1|foo1|bar1|elephant1
2|foo2|bar2|elephant2
3|foo3|bar3|elephant3
4|foo4|bar4|elephant4
...
```

Pour importer un fichier qui utilise un délimiteur personnalisé

1. Créez une table dans la base de données pour les données importées.

```
postgres=> CREATE TABLE test (a text, b text, c text, d text, e text);
```

2. Utilisez le format suivant de la fonction aws_s3.table_import_from_s3 pour importer des données à partir du fichier Amazon S3.

   Vous pouvez inclure l'appel de fonction aws_commons.create_s3_uri en ligne au sein de l'appel de fonction `aws_s3.table_import_from_s3` pour spécifier le fichier.

```
postgres=> SELECT aws_s3.table_import_from_s3(
   'test',
   'a,b,d,e',
   'DELIMITER ''|''',
   aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'pipeDelimitedSampleFile', 'us-
east-2')
);
```

Les données se retrouvent désormais dans la table dans les colonnes suivantes.

```
postgres=> SELECT * FROM test;
a | b | c | d | e
---+------+---+---+------+-----------
1 | foo1 | | bar1 | elephant1
2 | foo2 | | bar2 | elephant2
3 | foo3 | | bar3 | elephant3
4 | foo4 | | bar4 | elephant4
```

Importation d'un fichier compressé Amazon S3 (gzip)

L'exemple suivant montre comment importer un fichier compressé avec gzip à partir d'Amazon S3. Le fichier que vous importez doit comporter les métadonnées Amazon S3 suivantes :

- Clé : `Content-Encoding`
- Valeur : `gzip`

Si vous chargez le fichier à l'aide de la AWS Management Console, les métadonnées sont généralement appliquées par le système. Pour plus d'informations sur le chargement de fichiers sur Amazon S3 à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API, veuillez consulter Chargement d'objets dans le Guide de l'utilisateur Amazon Simple Storage Service.

Pour de plus amples informations sur les métadonnées Amazon S3 et les métadonnées fournies par le système, veuillez consulter Editing object metadata in the Amazon S3 console dans le Guide de l'utilisateur Amazon Simple Storage Service.

Importez le fichier gzip dans votrecluster Aurora PostgreSQL comme décrit ci-après.

```
postgres=> CREATE TABLE test_gzip(id int, a text, b text, c text, d text);
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_gzip', '', '(format csv)',
 'amzn-s3-demo-bucket', 'test-data.gz', 'us-east-2'
);
```

Importation d'un fichier codé Amazon S3

L'exemple suivant montre comment importer un fichier codé en Windows-1252 à partir d'Amazon S3.

```
postgres=> SELECT aws_s3.table_import_from_s3(
 'test_table', '', 'encoding ''WIN1252''',
 aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'SampleFile', 'us-east-2')
);
```

Références de fonctions

Fonctions

- aws_s3.table_import_from_s3
- aws_commons.create_s3_uri
- aws_commons.create_aws_credentials

aws_s3.table_import_from_s3

Importe les données Amazon S3 vers une table Aurora PostgreSQL. L'extension `aws_s3` fournit la fonction `aws_s3.table_import_from_s3`. La valeur renvoyée est du texte.

Syntaxe

Les paramètres requis sont `table_name`, `column_list` et `options`. Ils identifient la table de base de données et spécifient la façon dont les données sont copiées dans la table.

Vous pouvez également utiliser les paramètres suivants :

- Le paramètre `s3_info` spécifie le fichier Amazon S3 à importer. Lorsque vous utilisez ce paramètre, l'accès à Amazon S3 est fourni par un rôle IAM pour le cluster de base de données PostgreSQL.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1
)
```

- Le paramètre `credentials` spécifie les informations d'identification permettant d'accéder à Amazon S3. Lorsque vous utilisez ce paramètre, vous n'utilisez pas de rôle IAM.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   s3_info aws_commons._s3_uri_1,
   credentials aws_commons._aws_credentials_1
)
```

Paramètres

table_name

Chaîne de texte obligatoire contenant le nom de la table de base de données PostgreSQL dans laquelle importer les données.

column_list

Chaîne de texte obligatoire contenant la liste facultative des colonnes de la table de base de données PostgreSQL dans lesquelles copier les données. Si la chaîne est vide, toutes les colonnes de la table sont utilisées. Pour obtenir un exemple, veuillez consulter Importation d'un fichier Amazon S3 qui utilise un délimiteur personnalisé.

options

Chaîne de texte obligatoire contenant les arguments de la commande `COPY` de PostgreSQL. Ces arguments spécifient la façon dont les données sont copiées dans la table PostgreSQL. Pour plus d'informations, consultez la documentation sur la commande COPY de PostgreSQL.

s3_info

Type composite `aws_commons._s3_uri_1` contenant les informations suivantes sur l'objet S3 :

- `bucket` – Nom du compartiment Amazon S3 contenant le fichier.
- `file_path` – Nom du fichier Amazon S3, avec le chemin d'accès à celui-ci.
- `region` – Région AWS dans laquelle se trouve le fichier. Pour obtenir la liste des noms de régions AWS et les valeurs associées, consultez Régions et zones de disponibilité.

credentials

Type composite `aws_commons._aws_credentials_1` contenant les informations d'identification suivantes à utiliser pour l'opération d'importation :

- Clé d'accès
- Clé secrète
- Jeton de session

Pour plus d'informations sur la création d'une structure composite `aws_commons._aws_credentials_1`, veuillez consulter aws_commons.create_aws_credentials.

Syntaxe alternative

Pour faciliter le test, vous pouvez utiliser un ensemble étendu de paramètres au lieu des paramètres `s3_info` et `credentials`. Plusieurs variations de syntaxe supplémentaires pour la fonction `aws_s3.table_import_from_s3` sont fournies ci-dessous.

- Au lieu d'utiliser le paramètre `s3_info` pour identifier un fichier Amazon S3, utilisez la combinaison des paramètres `bucket`, `file_path` et `region`. Sous cette forme, l'accès à Amazon S3 est fourni par un rôle IAM sur l'instance de base de données PostgreSQL.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text
)
```

- Au lieu d'utiliser le paramètre `credentials` pour spécifier l'accès à Amazon S3, utilisez la combinaison des paramètres `access_key`, `session_key` et `session_token`.

```
aws_s3.table_import_from_s3 (
   table_name text,
   column_list text,
   options text,
   bucket text,
   file_path text,
   region text,
   access_key text,
   secret_key text,
   session_token text
)
```

Autres paramètres

bucket

Chaîne de texte comportant le nom du compartiment Amazon S3 qui contient le fichier.

file_path

Chaîne de texte contenant le nom du fichier Amazon S3, avec le chemin d'accès à celui-ci.

region

Une chaîne de texte identifiant l'emplacement de la Région AWS du fichier. Pour obtenir la liste des noms de Région AWS et les valeurs associées, consultez Régions et zones de disponibilité.

access_key

Chaîne de texte contenant la clé d'accès à utiliser pour l'opération d'importation. La valeur par défaut est NULL.

secret_key

Chaîne de texte contenant la clé secrète à utiliser pour l'opération d'importation. La valeur par défaut est NULL.

session_token

(Facultatif) Chaîne de texte contenant la clé de session à utiliser pour l'opération d'importation. La valeur par défaut est NULL.

aws_commons.create_s3_uri

Crée une structure `aws_commons._s3_uri_1` pour contenir les informations relatives au fichier Amazon S3. Utilisez les résultats de la fonction `aws_commons.create_s3_uri` dans le paramètre `s3_info` de la fonction aws_s3.table_import_from_s3.

Syntaxe

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

Paramètres

bucket

Chaîne de texte obligatoire contenant le nom du compartiment Amazon S3 pour le fichier.

file_path

Chaîne de texte obligatoire contenant le nom du fichier Amazon S3, avec le chemin d'accès à celui-ci.

region

Chaîne de texte obligatoire contenant la Région AWS dans laquelle se trouve le fichier. Pour obtenir la liste des noms de Région AWS et les valeurs associées, consultez Régions et zones de disponibilité.

aws_commons.create_aws_credentials

Définit une clé d'accès et une clé secrète dans une structure `aws_commons._aws_credentials_1`. Utilisez les résultats de la fonction `aws_commons.create_aws_credentials` dans le paramètre `credentials` de la fonction aws_s3.table_import_from_s3.

Syntaxe

```
aws_commons.create_aws_credentials(
```

```
   access_key text,
   secret_key text,
   session_token text
)
```

Paramètres

access_key

Chaîne de texte obligatoire contenant la clé d'accès à utiliser pour l'importation d'un fichier Amazon S3. La valeur par défaut est NULL.

secret_key

Chaîne de texte obligatoire contenant la clé secrète à utiliser pour l'importation d'un fichier Amazon S3. La valeur par défaut est NULL.

session_token

Chaîne de texte facultative contenant le jeton de session à utiliser pour l'importation d'un fichier Amazon S3. La valeur par défaut est NULL. Si vous saisissez le paramètre `session_token` facultatif, vous pouvez utiliser les informations d'identification temporaires.

Exportation de données à partir d’un cluster de bases de données Aurora PostgreSQL vers Amazon S3

Vous pouvez interroger des données à partir d’un cluster de bases de données Aurora PostgreSQL et les exporter directement dans des fichiers stockés dans un compartiment Amazon S3. Pour ce faire, vous devez d’abord installer l’extension `aws_s3` Aurora PostgreSQL . Cette extension vous fournit les fonctions que vous utilisez pour exporter les résultats des requêtes vers Amazon S3. Vous trouverez ci-dessous comment installer l’extension et comment exporter des données vers Amazon S3.

Vous pouvez exporter à partir d’une instance de base de données ou d’une instance Aurora Serverless v2 mise en service. Ces étapes ne sont pas prises en charge pour Aurora Serverless v1.

> **Note**
>
> L’exportation intercompte vers Amazon S3 n’est pas prise en charge.

Toutes les versions actuellement disponibles d'Aurora PostgreSQL prennent en charge l'exportation de données vers Amazon Simple Storage Service. Pour des informations détaillées sur les versions, consultez Mises à jour d'Amazon Aurora PostgreSQL dans Notes de mise à jour d'Aurora PostgreSQL.

Si vous n'avez pas de compartiment configuré pour votre exportation, consultez les rubriques suivantes du Guide de l'utilisateur d'Amazon Simple Storage Service.

- Configuration d'Amazon S3
- Création d'un compartiment

Par défaut, les données exportées depuis Aurora PostgreSQL vers Amazon S3 utilisent un chiffrement côté serveur avec une Clé gérée par AWS. Vous pouvez également utiliser une clé gérée par le client que vous avez créée auparavant. Si vous utilisez le chiffrement de compartiment, Amazon S3 doit être chiffré avec une clé AWS Key Management Service (AWS KMS) (SSE-KMS). Actuellement, les compartiments chiffrés avec des clés gérées par Amazon S3 (SSE-S3) ne sont pas pris en charge.

> **Note**
>
> Vous pouvez enregistrer les données d'instantané de la base de données et du cluster de bases de données sur Amazon S3 à l'aide d'AWS Management Console, d'AWS CLI ou de l'API Amazon RDS. Pour plus d'informations, consultez Exportation de données d'instantanés de cluster de bases de données vers Amazon S3.

Rubriques

- Installation de l'extension aws_s3
- Présentation de l'exportation de données vers Amazon S3
- Spécification du chemin d'accès au fichier Amazon S3 vers lequel effectuer l'exportation
- Configuration de l'accès à un compartiment Amazon S3
- Exportation de données de requête à l'aide de la fonction aws_s3.query_export_to_s3
- Références de fonctions
- Résolution des problèmes d'accès à Amazon S3

Installation de l’extension aws_s3

Avant de pouvoir utiliser Amazon Simple Storage Service avec votre cluster de bases de données Aurora PostgreSQL, vous devez installer l’extension `aws_s3`. Cette extension fournit des fonctions pour exporter des données depuis l’instance en écriture d’un cluster de bases de données Aurora PostgreSQL vers un compartiment Amazon S3. Il fournit également des fonctions pour importer des données depuis un compartiment Amazon S3. Pour plus d’informations, consultez Importation de données Amazon S3 dans une d'un cluster de base de données Aurora PostgreSQL. L’extension `aws_s3` dépend de certaines des fonctions d’aide de l’extension `aws_commons`, qui est installée automatiquement lorsque cela est nécessaire.

Pour installer l’extension **`aws_s3`**

1. Utilisez psql (ou pgAdmin) pour vous connecter à l’instance de base de données en écriture de votre cluster de bases de données Aurora PostgreSQL en tant qu’utilisateur disposant de privilèges `rds_superuser`. Si vous avez conservé le nom par défaut pendant le processus d’installation, vous vous connectez en tant que `postgres`.

   ```
   psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Pour installer l’extension, exécutez la commande suivante.

   ```
   postgres=> CREATE EXTENSION aws_s3 CASCADE;
   NOTICE: installing required extension "aws_commons"
   CREATE EXTENSION
   ```

3. Pour vérifier que l’extension est installée, vous pouvez utiliser la métacommande psql `\dx`.

   ```
   postgres=> \dx
          List of installed extensions
       Name     | Version |   Schema   |                Description
   -------------+---------+------------+-------------------------------------------
    aws_commons | 1.2     | public     | Common data types across AWS services
    aws_s3      | 1.1     | public     | AWS S3 extension for importing data from S3
    plpgsql     | 1.0     | pg_catalog | PL/pgSQL procedural language
   (3 rows)
   ```

Les fonctions d'importation de données depuis Amazon S3 et d'exportation de données vers Amazon S3 sont désormais disponibles.

Assurez-vous que votre version de Aurora PostgreSQL prend en charge les exportations vers Amazon S3

Vous pouvez vérifier que votre version d'Aurora PostgreSQL prend en charge l'exportation vers Amazon S3 en utilisant la commande `describe-db-engine-versions`. L'exemple suivant vérifie si la version 10.14 peut être exportée vers Amazon S3.

```
aws rds describe-db-engine-versions --region us-east-1 \
--engine aurora-postgresql --engine-version 10.14 | grep s3Export
```

Si la sortie inclut la chaîne `"s3Export"`, le moteur prend en charge les exportations Amazon S3. Sinon, le moteur ne les prend pas en charge.

Présentation de l'exportation de données vers Amazon S3

Pour exporter des données stockées dans un Aurora PostgreSQL vers un compartiment Amazon S3, procédez comme suit.

Pour exporter des données Aurora PostgreSQL vers S3

1. Identifiez un chemin d'accès de fichier Amazon S3 à utiliser pour exporter des données. Pour plus d'informations sur ce processus, consultez Spécification du chemin d'accès au fichier Amazon S3 vers lequel effectuer l'exportation.
2. Fournissez une autorisation d'accès au compartiment Amazon S3.

   Pour exporter des données vers un fichier Amazon S3, vous devez accorder au cluster de bases de données Aurora PostgreSQL l'autorisation d'accéder au compartiment Amazon S3 que l'exportation utilisera pour le stockage. Cette opération comprend les étapes suivantes :

   1. Créez une politique IAM donnant accès à un compartiment Amazon S3 vers lequel vous souhaitez exporter.
   2. Créez un rôle IAM.
   3. Attachez la politique que vous avez créée au rôle que vous avez créé.
   4. Ajoutez ce rôle IAM à votre cluster de base de données .

Pour plus d'informations sur ce processus, consultez Configuration de l'accès à un compartiment Amazon S3.

3. Identifiez une requête de base de données pour obtenir les données. Exportez les données de requête en appelant la fonction `aws_s3.query_export_to_s3`.

   Après avoir terminé les tâches de préparation précédentes, utilisez la fonction aws_s3.query_export_to_s3 pour exporter les résultats de requête vers Amazon S3. Pour plus d'informations sur ce processus, consultez Exportation de données de requête à l'aide de la fonction aws_s3.query_export_to_s3.

Spécification du chemin d'accès au fichier Amazon S3 vers lequel effectuer l'exportation

Spécifiez les informations suivantes pour identifier l'emplacement dans Amazon S3 vers lequel vous souhaitez exporter des données :

- Nom du compartiment – Un compartiment est un conteneur d'objets ou de fichiers Amazon S3.

  Pour plus d'informations sur le stockage de données avec Amazon S3, consultez Création d'un compartiment et Utilisation des objets dans le Guide de l'utilisateur Amazon Simple Storage Service.

- Chemin d'accès au fichier – Le chemin d'accès au fichier identifie l'emplacement de stockage de l'exportation dans le compartiment Amazon S3. Le chemin d'accès au fichier se compose des éléments suivants :
  - Préfixe de chemin facultatif qui identifie un chemin d'accès à un dossier virtuel.
  - Préfixe de fichier qui identifie un ou plusieurs fichiers à stocker. Les exportations les plus volumineuses sont stockées dans plusieurs fichiers, chacun ayant une taille maximale d'environ 6 Go. Les noms de fichiers supplémentaires ont le même préfixe de fichier mais en ajoutant `_part`*`XX`*. *`XX`* représente 2, puis 3, et ainsi de suite.

  Par exemple, un chemin d'accès de fichier avec un dossier `exports` et un préfixe de fichier `query-1-export` sera représenté par `/exports/query-1-export`.

- Région AWS (facultatif) – Région AWS où se trouve le compartiment Amazon S3. Si vous ne spécifiez pas de valeur de région AWS, Aurora enregistre vos fichiers dans Amazon S3 dans la même région AWS que celle où se trouve le cluster de bases de données.

> **Note**
>
> Actuellement, la région AWS doit être la même que celle du cluster de bases de données à l'origine de l'exportation.

Pour obtenir la liste des noms de régions AWS et les valeurs associées, consultez Régions et zones de disponibilité.

Pour conserver les informations de fichier Amazon S3 sur l'emplacement de stockage de l'exportation, vous pouvez utiliser la fonction aws_commons.create_s3_uri pour créer une structure composite `aws_commons._s3_uri_1` comme suit.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

Vous fournissez ultérieurement cette valeur `s3_uri_1` en tant que paramètre dans l'appel à la fonction aws_s3.query_export_to_s3. Pour obtenir des exemples, consultez Exportation de données de requête à l'aide de la fonction aws_s3.query_export_to_s3.

Configuration de l'accès à un compartiment Amazon S3

Pour exporter des données vers Amazon S3, accordez à votre cluster l'autorisation d'accéder au compartiment Amazon S3 dans lequel les fichiers doivent être stockés.

Pour cela, procédez comme suit :

Pour donner à un cluster de base de données PostgreSQL l'accès à Amazon S3 via un rôle IAM

1. Créez une politique IAM.

   Cette stratégie fournit le compartiment et les autorisations d'objet permettant à votre cluster de base de données PostgreSQL d'accéder à Amazon S3.

   Dans le cadre de la création de cette politique, procédez comme suit :

a. Incluez dans la stratégie les actions obligatoires suivantes pour permettre le transfert de fichiers de votre cluster de base de données PostgreSQL vers un compartiment Amazon S3 :

   - `s3:PutObject`
   - `s3:AbortMultipartUpload`

b. Incluez l'Amazon Resource Name (ARN) qui identifie le compartiment Amazon S3 et les objets du compartiment. Le format ARN pour l'accès à Amazon S3 est le suivant : `arn:aws:s3:::amzn-s3-demo-bucket/*`

Pour plus d'informations sur la création d'une politique IAM pour Aurora PostgreSQL, consultez Création et utilisation d'une politique IAM pour l'accès à une base de données IAM. Consultez également Didacticiel : création et attachement de votre première politique gérée par le client dans le Guide de l'utilisateur IAM.

La AWS CLI commande suivante crée une politique IAM nommée `rds-s3-export-policy` avec ces options. Elle accorde un accès à un compartiment nommé *amzn-s3-demo-bucket*.

> **Warning**
>
> Nous vous recommandons de configurer votre base de données dans un VPC privé dont les politiques de point de terminaison sont configurées pour accéder à des compartiments spécifiques. Pour plus d'informations, consultez Utilisation des stratégies de point de terminaison pour Amazon S3 dans le Guide de l'utilisateur Amazon VPC. Nous vous recommandons vivement de ne pas créer de politique avec accès à toutes les ressources. Cet accès peut constituer une menace pour la sécurité des données. Si vous créez une stratégie qui accorde à `S3:PutObject` un accès à toutes les ressources à l'aide de `"Resource":"*"`, un utilisateur disposant de privilèges d'exportation peut exporter des données vers tous les compartiments de votre compte. En outre, l'utilisateur peut exporter des données vers n'importe quel compartiment accessible publiquement en écriture dans votre région AWS .

Après avoir créé la politique, notez son ARN (Amazon Resource Name). Vous en aurez besoin par la suite pour attacher la politique à un rôle IAM.

```
aws iam create-policy  --policy-name rds-s3-export-policy  --policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "s3export",
         "Action": [
           "s3:PutObject*",
           "s3:ListBucket",
           "s3:GetObject*",
           "s3:DeleteObject*",
           "s3:GetBucketLocation",
           "s3:AbortMultipartUpload"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket/*"
         ]
       }
     ]
   }'
```

2. Créez un rôle IAM.

   L'objectif est ici de permettre à Aurora PostgreSQL d'endosser ce rôle IAM en votre nom pour accéder à vos compartiments Amazon S3. Pour plus d'informations, consultez Création d'un rôle pour déléguer des autorisations à un utilisateur IAM dans le Guide de l'utilisateur IAM.

   Nous vous recommandons d'utiliser les clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount` dans des politiques basées sur les ressources pour limiter les autorisations du service à une ressource spécifique. C'est le moyen le plus efficace de se protéger contre le problème du député confus.

   Si vous utilisez les deux clés de contexte de condition globale et que la valeur de `aws:SourceArn` contient l'ID de compte, la valeur de `aws:SourceAccount` et le compte indiqué dans la valeur de `aws:SourceArn` doivent utiliser le même ID de compte lorsqu'il est utilisé dans la même déclaration de politique.

   - Utilisez `aws:SourceArn` si vous souhaitez un accès interservices pour une seule ressource.
   - Utilisez `aws:SourceAccount` si vous souhaitez autoriser une ressource de ce compte à être associée à l'utilisation interservices.

Dans la politique, veillez à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource. L'exemple suivant montre comment procéder à l'aide de la AWS CLI commande pour créer un rôle nommé`rds-s3-export-role`.

Example

Pour Linux, macOS ou Unix :

```
aws iam create-role  \
    --role-name rds-s3-export-role  \
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
          },
         "Action": "sts:AssumeRole",
         "Condition": {
             "StringEquals": {
                "aws:SourceAccount": "111122223333",
                "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:dbname"
                }
             }
       }
     ]
   }'
```

Pour Windows :

```
aws iam create-role  ^
    --role-name rds-s3-export-role  ^
    --assume-role-policy-document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
            "Service": "rds.amazonaws.com"
```

```
          },
          "Action": "sts:AssumeRole",
          "Condition": {
              "StringEquals": {
                 "aws:SourceAccount": "111122223333",
                 "aws:SourceArn": "arn:aws:rds:us-
east-1:111122223333:cluster:dbname"
                 }
              }
       }
     ]
   }'
```

3. Attachez la politique IAM que vous avez créée au rôle IAM que vous venez de créer.

   La AWS CLI commande suivante associe la politique créée précédemment au rôle nommé `rds-s3-export-role`. Remplacer *`your-policy-arn`* par l'ARN de stratégie que vous avez noté lors d'une étape précédente.

   ```
   aws iam attach-role-policy  --policy-arn your-policy-arn  --role-name rds-s3-
   export-role
   ```

4. Ajoutez le rôle IAM au cluster de base de données. Pour ce faire, utilisez le AWS Management Console ou AWS CLI, comme décrit ci-dessous.

Console

Pour ajouter un rôle IAM au cluster de base de données PostgreSQL à l'aide de la console

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Choisissez le nom du cluster de base de données PostgreSQL pour afficher ses détails.
3. Dans l'onglet Connectivité & sécurité de la section Gérer les rôles IAM, choisissez le rôle à ajouter sous Ajouter des rôles IAM à cette instance.
4. Sous Fonctionnalité, choisissez s3Export.
5. Choisissez Ajouter un rôle.

AWS CLI

Pour ajouter un rôle IAM à un cluster de bases de données PostgreSQL à l'aide de CLI

- Utilisez la commande suivante pour ajouter le rôle au cluster de bases de données PostgreSQL nommé `my-db-cluster`. Remplacez *`your-role-arn`* par l'ARN de rôle que vous avez noté lors d'une étape précédente. Utilisez `s3Export` comme valeur de l'option `--feature-name`.

  Example

  Pour Linux, macOS ou Unix :

```
aws rds add-role-to-db-cluster \
   --db-cluster-identifier my-db-cluster \
   --feature-name s3Export \
   --role-arn your-role-arn   \
   --region your-region
```

  Pour Windows :

```
aws rds add-role-to-db-cluster ^
   --db-cluster-identifier my-db-cluster ^
   --feature-name s3Export ^
   --role-arn your-role-arn ^
   --region your-region
```

Exportation de données de requête à l'aide de la fonction aws_s3.query_export_to_s3

Exportez vos données PostgreSQL vers Amazon S3 en appelant la fonction aws_s3.query_export_to_s3.

Rubriques

- Prérequis
- Appel de aws_s3.query_export_to_s3
- Exportation vers un fichier CSV qui utilise un délimiteur personnalisé
- Exportation vers un fichier binaire avec encodage

Prérequis

Avant d'utiliser la fonction `aws_s3.query_export_to_s3`, assurez-vous de remplir les conditions préalables suivantes :

- Installez les extensions PostgreSQL requises comme décrit dans Présentation de l'exportation de données vers Amazon S3.
- Déterminez vers quel emplacement Amazon S3 exporter vos données comme décrit dans Spécification du chemin d'accès au fichier Amazon S3 vers lequel effectuer l'exportation.
- Assurez-vous que le cluster de bases de données dispose d'un accès à Amazon S3 comme décrit dans Configuration de l'accès à un compartiment Amazon S3.

Les exemples suivants utilisent une table de base de données appelée `sample_table`. Ces exemples exportent les données dans un compartiment appelé *`amzn-s3-demo-bucket`*. Les exemples de table et de données sont créés avec les instructions SQL suivantes dans psql.

```
psql=> CREATE TABLE sample_table (bid bigint PRIMARY KEY, name varchar(80));
psql=> INSERT INTO sample_table (bid,name) VALUES (1, 'Monday'), (2,'Tuesday'), (3,
 'Wednesday');
```

Appel de aws_s3.query_export_to_s3

Ce qui suit montre les techniques de base permettant d'appeler la fonction aws_s3.query_export_to_s3.

Ces exemples utilisent la variable `s3_uri_1` pour identifier une structure contenant les informations identifiant le fichier Amazon S3. Utilisez la fonction aws_commons.create_s3_uri pour créer la structure.

```
psql=> SELECT aws_commons.create_s3_uri(
   'amzn-s3-demo-bucket',
   'sample-filepath',
   'us-west-2'
) AS s3_uri_1 \gset
```

Bien que les paramètres varient pour les deux appels de fonction `aws_s3.query_export_to_s3` suivants, les résultats sont les mêmes pour ces exemples. Toutes les lignes de la table `sample_table` sont exportées dans un compartiment appelé *`amzn-s3-demo-bucket`*.

```
psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1');

psql=> SELECT * FROM aws_s3.query_export_to_s3('SELECT * FROM
 sample_table', :'s3_uri_1', options :='format text');
```

Les paramètres sont décrits comme suit :

- `'SELECT * FROM sample_table'` – Le premier paramètre est une chaîne de texte obligatoire contenant une requête SQL. Le moteur PostgreSQL exécute cette requête. Les résultats de la requête sont copiés dans le compartiment S3 identifié dans d'autres paramètres.
- `:'s3_uri_1'` – Ce paramètre est une structure qui identifie le fichier Amazon S3. Cet exemple utilise une variable pour identifier la structure créée précédemment. Vous pouvez plutôt créer la structure en incluant l'appel de fonction `aws_commons.create_s3_uri` en ligne dans l'appel de fonction `aws_s3.query_export_to_s3` comme suit.

  ```
  SELECT * from aws_s3.query_export_to_s3('select * from sample_table',
     aws_commons.create_s3_uri('amzn-s3-demo-bucket', 'sample-filepath', 'us-west-2')
  );
  ```
- `options :='format text'` – Le paramètre `options` est une chaîne de texte facultative contenant des arguments `COPY` PostgreSQL. Le processus de copie utilise les arguments et le format de la commande PostgreSQL COPY.

Si le fichier spécifié n'existe pas dans le compartiment Amazon S3, il est créé. Si le fichier existe déjà, il est remplacé. La syntaxe d'accès aux données exportées dans Amazon S3 est la suivante.

```
s3-region://bucket-name[/path-prefix]/file-prefix
```

Les exportations les plus volumineuses sont stockées dans plusieurs fichiers, chacun ayant une taille maximale d'environ 6 Go. Les noms de fichiers supplémentaires ont le même préfixe de fichier mais en ajoutant `_part`*XX*. *XX* représente 2, puis 3, et ainsi de suite. Par exemple, supposons que vous spécifiiez le chemin d'accès où vous stockez les fichiers de données comme suit.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
```

Si l'exportation doit créer trois fichiers de données, le compartiment Amazon S3 contient les fichiers de données suivants.

```
s3-us-west-2://amzn-s3-demo-bucket/my-prefix
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part2
s3-us-west-2://amzn-s3-demo-bucket/my-prefix_part3
```

Pour obtenir la référence complète de cette fonction et les moyens supplémentaires de l'appeler, consultez aws_s3.query_export_to_s3. Pour plus d'informations sur l'accès aux fichiers dans Amazon S3, consultez Afficher un objet dans le Guide de l'utilisateur Amazon Simple Storage Service.

Exportation vers un fichier CSV qui utilise un délimiteur personnalisé

L'exemple suivant montre comment appeler la fonction aws_s3.query_export_to_s3 pour exporter des données vers un fichier qui utilise un délimiteur personnalisé. L'exemple utilise les arguments de la commande PostgreSQL COPY pour spécifier le format CSV (valeur séparée par des virgules) et un délimiteur deux-points (:).

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format csv, delimiter $$:$$');
```

Exportation vers un fichier binaire avec encodage

L'exemple suivant montre comment appeler la fonction aws_s3.query_export_to_s3 pour exporter des données vers un fichier binaire ayant un encodage Windows-1253.

```
SELECT * from aws_s3.query_export_to_s3('select * from basic_test', :'s3_uri_1',
 options :='format binary, encoding WIN1253');
```

Références de fonctions

Fonctions

- aws_s3.query_export_to_s3
- aws_commons.create_s3_uri

aws_s3.query_export_to_s3

Exporte un résultat de requête PostgreSQL vers un compartiment Amazon S3. L'extension `aws_s3` fournit la fonction `aws_s3.query_export_to_s3`.

Les deux paramètres requis sont `query` et `s3_info`. Ils définissent la requête à exporter et identifient le compartiment Amazon S3 vers lequel effectuer l'exportation. Un paramètre facultatif

appelé `options` permet de définir différents paramètres d'exportation. Pour obtenir des exemples d'utilisation de la fonction `aws_s3.query_export_to_s3`, consultez Exportation de données de requête à l'aide de la fonction aws_s3.query_export_to_s3.

Syntaxe

```
aws_s3.query_export_to_s3(
    query text,
    s3_info aws_commons._s3_uri_1,
    options text,
    kms_key text
)
```

Paramètres d'entrée

query

Chaîne de texte obligatoire contenant une requête SQL exécutée par le moteur PostgreSQL. Les résultats de cette requête sont copiés dans un compartiment S3 identifié dans le paramètre `s3_info`.

s3_info

Type composite `aws_commons._s3_uri_1` contenant les informations suivantes sur l'objet S3 :

- `bucket` – Nom du compartiment Amazon S3 contenant le fichier.
- `file_path` – Nom du fichier Amazon S3 et chemin d'accès à celui-ci.
- `region` – Région AWS dans laquelle se trouve le compartiment. Pour obtenir la liste des noms de régions AWS et les valeurs associées, consultez Régions et zones de disponibilité.

  Actuellement, cette valeur doit être la même région AWS que celle du cluster de bases de données à l'origine de l'exportation. La valeur par défaut est la région AWS du cluster de bases de données à l'origine de l'exportation.

Pour créer une structure composite `aws_commons._s3_uri_1`, consultez aws_commons.create_s3_uri fonction.

options

Chaîne de texte facultative contenant les arguments de la commande `COPY` de PostgreSQL. Ces arguments spécifient la façon dont les données doivent être copiées lors de l'exportation. Pour plus d'informations, consultez la documentation sur la commande COPY de PostgreSQL.

kms_key text

Chaîne de texte facultative contenant la clé KMS gérée par le client du compartiment S3 vers lequel exporter les données.

Autres paramètres d'entrée

Pour faciliter le test, vous pouvez utiliser un ensemble étendu de paramètres au lieu du paramètre `s3_info`. Plusieurs variations de syntaxe supplémentaires pour la fonction `aws_s3.query_export_to_s3` sont fournies ci-dessous.

Au lieu d'utiliser le paramètre `s3_info` pour identifier un fichier Amazon S3, utilisez la combinaison des paramètres `bucket`, `file_path` et `region`.

```
aws_s3.query_export_to_s3(
    query text,
    bucket text,
    file_path text,
    region text,
    options text,
    kms_key text
)
```

query

Chaîne de texte obligatoire contenant une requête SQL exécutée par le moteur PostgreSQL. Les résultats de cette requête sont copiés dans un compartiment S3 identifié dans le paramètre `s3_info`.

bucket

Chaîne de texte obligatoire comportant le nom du compartiment Amazon S3 qui contient le fichier.

file_path

Chaîne de texte obligatoire contenant le nom du fichier Amazon S3, avec le chemin d'accès à celui-ci.

region

Chaîne de texte facultative contenant la région AWS dans laquelle se trouve le compartiment. Pour obtenir la liste des noms de régions AWS et les valeurs associées, consultez Régions et zones de disponibilité.

Actuellement, cette valeur doit être la même région AWS que celle du cluster de bases de données à l'origine de l'exportation. La valeur par défaut est la région AWS du cluster de bases de données à l'origine de l'exportation.

options

Chaîne de texte facultative contenant les arguments de la commande `COPY` de PostgreSQL. Ces arguments spécifient la façon dont les données doivent être copiées lors de l'exportation. Pour plus d'informations, consultez la documentation sur la commande COPY de PostgreSQL.

kms_key text

Chaîne de texte facultative contenant la clé KMS gérée par le client du compartiment S3 vers lequel exporter les données.

Paramètres de sortie

```
aws_s3.query_export_to_s3(
    OUT rows_uploaded bigint,
    OUT files_uploaded bigint,
    OUT bytes_uploaded bigint
)
```

rows_uploaded

Nombre de lignes de table qui ont été téléchargées avec succès vers Amazon S3 pour la requête donnée.

files_uploaded

Nombre de fichiers téléchargés vers Amazon S3. Les fichiers sont créés avec des tailles d'environ 6 Go. Chaque fichier supplémentaire créé voit l'élément `_part`*XX* ajouté à son nom. *XX* représente 2, puis 3, et ainsi de suite.

bytes_uploaded

Nombre total d'octets téléchargés vers Amazon S3.

Exemples

```
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-
demo-bucket', 'sample-filepath');
```

```
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-
demo-bucket', 'sample-filepath','us-west-2');
psql=> SELECT * from aws_s3.query_export_to_s3('select * from sample_table', 'amzn-s3-
demo-bucket', 'sample-filepath','us-west-2','format text');
```

aws_commons.create_s3_uri

Crée une structure `aws_commons._s3_uri_1` pour contenir les informations relatives au fichier Amazon S3. Vous utilisez les résultats de la fonction `aws_commons.create_s3_uri` dans le paramètre `s3_info` de la fonction aws_s3.query_export_to_s3. Pour obtenir un exemple d'utilisation de la fonction `aws_commons.create_s3_uri`, consultez Spécification du chemin d'accès au fichier Amazon S3 vers lequel effectuer l'exportation.

Syntaxe

```
aws_commons.create_s3_uri(
   bucket text,
   file_path text,
   region text
)
```

Paramètres d'entrée

bucket

Chaîne de texte obligatoire contenant le nom du compartiment Amazon S3 pour le fichier.

file_path

Chaîne de texte obligatoire contenant le nom du fichier Amazon S3, avec le chemin d'accès à celui-ci.

region

Chaîne de texte obligatoire contenant la région AWS dans laquelle se trouve le fichier. Pour obtenir la liste des noms de régions AWS et les valeurs associées, consultez Régions et zones de disponibilité.

Résolution des problèmes d'accès à Amazon S3

Si vous rencontrez des problèmes de connexion lorsque vous essayez d'exporter des données vers Amazon S3, confirmez d'abord que les règles d'accès sortant du groupe de sécurité VPC associé à

votre instance de base de données permettent la connectivité réseau. Plus précisément, le groupe de sécurité doit comporter une règle qui autorise l'instance de base de données à envoyer du trafic TCP au port 443 et à toute adresse IPv4 (0.0.0.0/0). Pour de plus amples informations, consultez Créer un groupe de sécurité qui autorise l'accès au cluster de bases de données dans le VPC.

Voir également les recommandations suivantes :

- Résolution des problèmes liés à Identity and Access Amazon Aurora
- Dépannage d'Amazon S3 dans le Guide de l'utilisateur Amazon Simple Storage Service.
- Dépannage d'Amazon S3 et IAM dans le Guide de l'utilisateur IAM

Invocation d'une AWS Lambda fonction depuis une instance de base de données Aurora PostgreSQL pour PostgreSQL

AWS Lambda est un service de calcul piloté par les événements qui vous permet d'exécuter du code sans provisionner ni gérer de serveurs. Il peut être utilisé avec de nombreux AWS services, notamment Aurora PostgreSQL . Par exemple, vous pouvez utiliser des fonctions Lambda pour traiter les notifications d'événements à partir d'une base de données ou pour charger des données à partir de fichiers chaque fois qu'un nouveau fichier est chargé sur Amazon S3. Pour en savoir plus sur Lambda, consultez Qu'est-ce que c'est ? AWS Lambda dans le Guide AWS Lambda du développeur.

> **Note**
>
> L'appel de AWS Lambda fonctions est pris en charge dans Aurora PostgreSQL 11.9 et versions ultérieures (y compris). Aurora Serverless v2

Vous trouverez ci-après des résumés des étapes nécessaires.

Pour plus d'informations sur les fonctions Lambda, consultez Mise en route avec Lambda et Principes de base d'AWS Lambda dans le Guide du développeur AWS Lambda .

Rubriques

- Étape 1 : configurer votre instance de base de données Aurora PostgreSQL RDS pour PostgreSQL vers AWS Lambda
- Étape 2 : configurer IAM pour votre instance de base de données Aurora PostgreSQL pour PostgreSQL et AWS Lambda

- Étape 3 : installer l’extension aws_lambda pour un cluster de bases de données Aurora PostgreSQL
- Étape 4 : utiliser les fonctions d’assistance Lambda avec votre cluster de bases de données Aurora PostgreSQL (Facultatif)
- Étape 5 : appeler une fonction Lambda à partir de votre cluster de bases de données Aurora PostgreSQL.
- Étape 6 : accorder aux autres utilisateurs l’autorisation d’appeler les fonctions Lambda
- Exemples : appel de fonctions Lambda à partir de votre cluster de bases de données Aurora PostgreSQL
- Messages d'erreur de fonction Lambda
- Référence des fonctions et des paramètres AWS Lambda

Étape 1 : configurer votre instance de base de données Aurora PostgreSQL RDS pour PostgreSQL vers AWS Lambda

Les fonctions Lambda s'exécutent toujours au sein d'un Amazon VPC appartenant au service. AWS Lambda Lambda applique des règles d’accès réseau et de sécurité à ce VPC, le maintient et le surveille automatiquement. Votre cluster de bases de données Aurora PostgreSQL envoie du trafic réseau vers le VPC du service Lambda. La façon dont vous configurez cela dépend de si votre instance de base de données primaire du cluster de bases de données Aurora est publique ou privée.

- PubliclyAccessible `true` Pour trouver la valeur de cette propriété, vous pouvez utiliser la describe-db-instances AWS CLI commande. Vous pouvez également utiliser la AWS Management Console afin d’ouvrir l’onglet Connectivity & security (Connectivité et sécurité) et vérifier que l’option Publicly accessible (Accessible publiquement) est définie sur Yes (Oui). Pour vérifier que l’instance se trouve dans le sous-réseau public de votre VPC, vous pouvez utiliser la AWS Management Console ou AWS CLI.

  Pour configurer l'accès à Lambda, vous utilisez le AWS Management Console ou AWS CLI pour créer une règle sortante sur le groupe de sécurité de votre VPC. La règle de sortie indique que TCP peut utiliser le port 443 pour envoyer des paquets à n'importe quelle IPv4 adresse (0.0.0.0/0).
- Cluster de base de données Aurora PostgreSQL — Dans ce cas, la propriété PubliclyAccessible `false` « » de l'instance est ou se trouve dans un sous-réseau privé. Pour permettre à l’instance de fonctionner avec Lambda, vous pouvez utiliser une passerelle traduction d’adresses réseau

(NAT). Pour plus d'informations, consultez Passerelles NAT. Vous pouvez également configurer votre VPC avec un point de terminaison VPC pour Lambda. Pour plus d'informations, consultez Points de terminaison VPC dans le Guide de l'utilisateur Amazon VPC. Le point de terminaison répond aux appels faits par votre cluster de bases de données Aurora PostgreSQL à vos fonctions Lambda.

Votre VPC peut désormais interagir avec le AWS Lambda VPC au niveau du réseau. Ensuite, vous configurez les autorisations à l'aide d'IAM.

Étape 2 : configurer IAM pour votre instance de base de données Aurora PostgreSQL pour PostgreSQL et AWS Lambda

L'appel de fonctions Lambda depuis votre cluster de bases de données Aurora PostgreSQL requiert certains privilèges. Pour configurer les privilèges requis, nous vous recommandons de créer une politique IAM qui permet d'appeler des fonctions Lambda, d'attribuer cette politique à un rôle, puis d'appliquer le rôle à votre cluster de bases de données. Cette approche accorde au cluster de bases de données des privilèges pour appeler la fonction Lambda spécifiée en votre nom. Les étapes suivantes expliquent comment procéder à l'aide de l' AWS CLI.

Pour configurer les autorisations IAM pour l'utilisation de votre clusteravec Lambda

1. Utilisez la AWS CLI commande create-policy pour créer une politique IAM qui permet à votre instance de base de données Aurora PostgreSQL d'appeler la fonction Lambda spécifiée. (L'ID d'instruction (Sid) est une description facultative pour votre instruction de politique et n'a aucun effet sur l'utilisation.) Cette politique accorde à votre cluster de base de données Aurora les autorisations minimales requises pour appeler la fonction Lambda spécifiée.

```
aws iam create-policy  --policy-name rds-lambda-policy --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "AllowAccessToExampleFunction",
        "Effect": "Allow",
        "Action": "lambda:InvokeFunction",
        "Resource": "arn:aws:lambda:aws-region:444455556666:function:my-function"
        }
    ]
}'
```

Vous pouvez également utiliser la politique `AWSLambdaRole` prédéfinie qui vous permet d'appeler n'importe laquelle de vos fonctions Lambda. Pour plus d'informations, consultez la rubrique Politiques IAM basées sur l'identité pour Lambda.

2. Utilisez la AWS CLI commande create-role pour créer un rôle IAM que la politique peut assumer lors de l'exécution.

```
aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

3. Appliquez la politique au rôle à l'aide de la attach-role-policy AWS CLI commande.

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::444455556666:policy/rds-lambda-policy \
    --role-name rds-lambda-role --region aws-region
```

4. https://awscli.amazonaws.com/v2/documentation/api/latest/reference/rds/add-role-to-db-cluster.html AWS CLI Cette dernière étape permet aux utilisateurs de base de données de votre cluster de bases de données d'appeler des fonctions Lambda.

```
aws rds add-role-to-db-cluster \
       --db-cluster-identifier my-cluster-name \
       --feature-name Lambda \
       --role-arn  arn:aws:iam::444455556666:role/rds-lambda-role   \
       --region aws-region
```

Une fois le VPC et les configurations IAM terminées, vous pouvez désormais installer l'extension `aws_lambda`. (Notez que vous pouvez installer l'extension à tout moment, mais tant que vous n'avez pas configuré la prise en charge du VPC et les privilèges IAM corrects, l'extension `aws_lambda` n'ajoute rien aux fonctionnalités de votre cluster de bases de données Aurora PostgreSQL.)

Étape 3 : installer l'extension **aws_lambda** pour un cluster de bases de données Aurora PostgreSQL

Cette extension offre à votre cluster de bases de données Aurora PostgreSQL la capacité d'appeler des fonctions Lambda depuis PostgreSQL.

Pour installer l'extension **aws_lambda** dans votre cluster de bases de données Aurora PostgreSQL

Utilisez la ligne de commande `psql` de PostgreSQL ou l'outil pgAdmin afin de vous connecter à votre cluster de bases de données Aurora PostgreSQL.

1. Connectez-vous à votre cluster de bases de données Aurora PostgreSQL en tant qu'utilisateur doté de privilèges `rds_superuser`. L'utilisateur `postgres` par défaut est illustré dans l'exemple.

   ```
   psql -h cluster-instance.444455556666.aws-region.rds.amazonaws.com -U postgres -p
    5432
   ```

2. Installez l'extension `aws_lambda`. L'extension `aws_commons` est également requise. Elle fournit des fonctions d'assistance pour `aws_lambda` et de nombreuses autres extensions Aurora pour PostgreSQL. Si elle n'est pas déjà sur votre cluster de bases de données Aurora PostgreSQL, elle est installée avec `aws_lambda` comme illustré ci-dessous.

   ```
   CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
   NOTICE:  installing required extension "aws_commons"
   CREATE EXTENSION
   ```

L'extension `aws_lambda` est installée dans l'instance de base de données primaire de votre cluster de bases de données Aurora PostgreSQL. Vous pouvez désormais créer des structures de commodité pour appeler vos fonctions Lambda.

Étape 4 : utiliser les fonctions d'assistance Lambda avec votre cluster de bases de données Aurora PostgreSQL (Facultatif)

Vous pouvez utiliser les fonctions d'assistance de l'extension `aws_commons` pour préparer les entités que vous pouvez appeler plus facilement depuis PostgreSQL. Pour ce faire, vous avez besoin des informations suivantes concernant vos fonctions Lambda :

- Function name (Nom de la fonction) — Le nom, l'Amazon Resource Name (ARN), la version ou l'alias de la fonction Lambda. La politique IAM créée dans Étape 2 : configurer IAM pour votre cluster et Lambda nécessite l'ARN, nous vous recommandons donc d'utiliser l'ARN de votre fonction.
- AWS Région

Pour conserver les informations de nom de la fonction Lambda, utilisez la fonction aws_commons.create_lambda_function_arn. Cette fonction d'assistance crée une structure composite `aws_commons._lambda_function_arn_1` avec les détails requis par la fonction d'appel. Vous trouverez ci-dessous trois autres approches pour configurer cette structure composite.

```
SELECT aws_commons.create_lambda_function_arn(
   'my-function',
   'aws-region'
) AS aws_lambda_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   '111122223333:function:my-function',
   'aws-region'
) AS lambda_partial_arn_1 \gset
```

```
SELECT aws_commons.create_lambda_function_arn(
   'arn:aws:lambda:aws-region:111122223333:function:my-function'
) AS lambda_arn_1 \gset
```

N'importe laquelle de ces valeurs peut être utilisée dans les appels à la fonction aws_lambda.invoke. Pour obtenir des exemples, consultez Étape 5 : appeler une fonction Lambda à partir de votre cluster de bases de données Aurora PostgreSQL..

Étape 5 : appeler une fonction Lambda à partir de votre cluster de bases de données Aurora PostgreSQL.

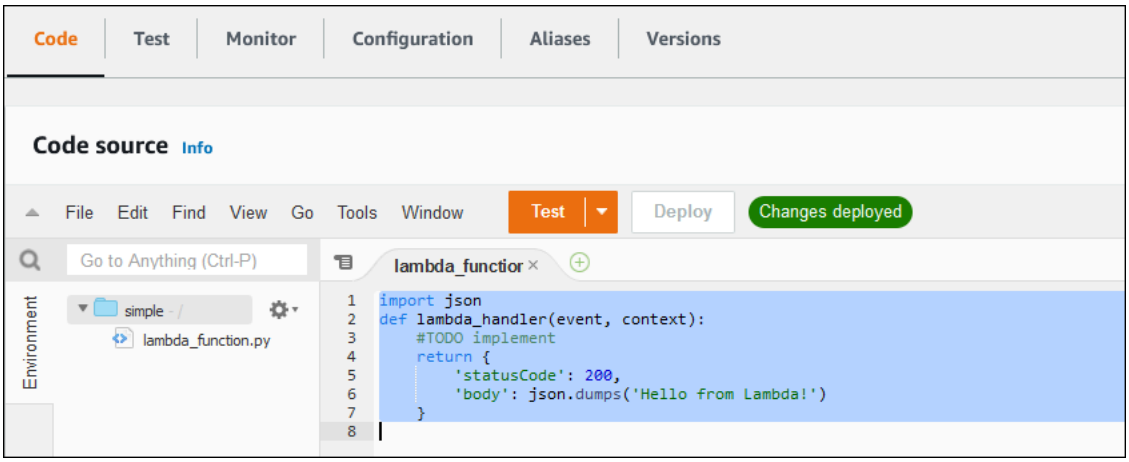
La fonction `aws_lambda.invoke` se comporte de manière synchrone ou asynchrone, en fonction du `invocation_type`. Les deux alternatives à ce paramètre sont `RequestResponse` (valeur par défaut) et `Event`, comme suit.

- **`RequestResponse`** — Ce type d'appel est synchrone. Il s'agit du comportement par défaut lorsque l'appel est effectué sans spécifier de type d'appel. La charge utile de réponse inclut les

résultats de la fonction `aws_lambda.invoke`. Utilisez ce type d'appel lorsque votre flux de travail nécessite la réception des résultats de la fonction Lambda avant de continuer.

- **`Event`** — Ce type d'appel est asynchrone. La réponse n'inclut pas de charge utile contenant des résultats. Utilisez ce type d'appel lorsque votre flux de travail n'a pas besoin de résultat de la fonction Lambda pour continuer le traitement.

Pour tester simplement votre configuration, vous pouvez vous connecter à votre instance de base de données en utilisant `psql` et appeler un exemple de fonction depuis la ligne de commande. Supposons que l'une des fonctions de base soit configurée sur votre service Lambda, telle que la fonction simple Python affichée dans la capture d'écran suivante.



Pour invoquer un exemple de fonction

1. Connectez-vous à votre instance de base de données primaire avec `psql` ou pgAdmin.

```
psql -h cluster.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
```

2. Appelez la fonction en utilisant son ARN.

```
SELECT * from
 aws_lambda.invoke(aws_commons.create_lambda_function_arn('arn:aws:lambda:aws-
region:444455556666:function:simple', 'us-west-1'), '{"body": "Hello from
 Postgres!"}'::json );
```

La réponse se présente comme suit.

```
status_code |                          payload                          |
 executed_version | log_result
```

```
-------------+------------------------------------------------------------
+------------------+------------
         200 | {"statusCode": 200, "body": "\"Hello from Lambda!\""} | $LATEST
     |
(1 row)
```

Si votre tentative d'appel ne réussit pas, consultez Messages d'erreur de fonction Lambda .

Étape 6 : accorder aux autres utilisateurs l'autorisation d'appeler les fonctions Lambda

À ce stade des procédures, vous êtes le seul, en tant que `rds_superuser`, à pouvoir appeler vos fonctions Lambda. Pour permettre à d'autres utilisateurs d'appeler les fonctions que vous avez créées, vous devez leur accorder des autorisations.

Pour accorder à d'autres personnes l'autorisation d'appeler les fonctions Lambda

1. Connectez-vous à votre instance de base de données primaire avec `psql` ou pgAdmin.

   ```
   psql -h cluster.444455556666.aws-region.rds.amazonaws.com -U postgres -p 5432
   ```

2. Exécutez les commandes SQL suivantes :

   ```
   postgres=>  GRANT USAGE ON SCHEMA aws_lambda TO db_username;
   GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA aws_lambda TO db_username;
   ```

Exemples : appel de fonctions Lambda à partir de votre cluster de bases de données Aurora PostgreSQL

Ci-dessous, vous pouvez trouver plusieurs exemples d'appel de la fonction aws_lambda.invoke. La plupart des exemples utilisent la structure composite `aws_lambda_arn_1` que vous créez dans Étape 4 : utiliser les fonctions d'assistance Lambda avec votre cluster de bases de données Aurora PostgreSQL (Facultatif) pour simplifier la transmission des détails de la fonction. Pour obtenir un exemple d'appel asynchrone, reportez-vous à la section Exemple : appel asynchrone (Event) de fonctions Lambda. Tous les autres exemples répertoriés utilisent l'appel synchrone.

Pour en savoir plus sur les types d'appel Lambda, consultez Appel de fonctions Lambda dans le Guide du développeur AWS Lambda. Pour plus d'informations sur `aws_lambda_arn_1`, consultez aws_commons.create_lambda_function_arn.

Liste d'exemples

Exemple : appel synchrone (RequestResponse) de fonctions Lambda

Voici deux exemples d'appel synchrone de fonction Lambda. Les résultats de ces appels de fonction `aws_lambda.invoke` sont identiques.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
```

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse');
```

Les paramètres sont décrits comme suit :

- `:'aws_lambda_arn_1'` — Ce paramètre identifie la structure composite créée dans Étape 4 : utiliser les fonctions d'assistance Lambda avec votre cluster de bases de données Aurora PostgreSQL (Facultatif), avec la fonction d'assistance `aws_commons.create_lambda_function_arn`. Vous pouvez également créer cette structure en ligne dans votre appel `aws_lambda.invoke` comme suit.

  ```
  SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-function',
   'aws-region'),
  '{"body": "Hello from Postgres!"}'::json
  );
  ```

- `'{"body": "Hello from PostgreSQL!"}'::json` : données utiles JSON à passer à la fonction Lambda.
- `'RequestResponse'` – Type d'appel Lambda.

Exemple : appel asynchrone (Event) de fonctions Lambda

Voici un exemple d'appel de fonction Lambda asynchrone. Le type d'appel `Event` planifie l'appel de fonction Lambda avec la charge utile d'entrée spécifiée et renvoie une réponse immédiatement. Utiliser le type d'appel `Event` dans certains flux de travail qui ne dépendent pas des résultats de la fonction Lambda.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'Event');
```

Exemple : capture du journal d'exécution Lambda dans une réponse de fonction

Vous pouvez inclure les 4 derniers Ko du journal d'exécution dans la réponse de la fonction à l'aide du paramètre `log_type` dans votre appel de fonction `aws_lambda.invoke`. Par défaut, ce paramètre est défini sur `None`, mais vous pouvez spécifier `Tail` afin de capturer les résultats du journal d'exécution Lambda dans la réponse, comme indiqué ci-dessous.

```
SELECT *, select convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

Définissez le paramètre aws_lambda.invoke de la fonction `log_type` sur `Tail` pour inclure le journal d'exécution dans la réponse. La valeur par défaut du paramètre `log_type` est `None`.

Le `log_result` qui est retourné est une chaîne `base64` encodée. Vous pouvez décoder le contenu à l'aide d'une combinaison des fonctions PostgreSQL `decode` et `convert_from`.

Pour plus d'informations sur `log_type`, consultez aws_lambda.invoke.

Exemple : inclusion du contexte client dans une fonction Lambda

La fonction `aws_lambda.invoke` possède un paramètre `context` que vous pouvez utiliser pour transférer des informations séparées de la charge utile, comme indiqué ci-dessous.

```
SELECT *, convert_from(decode(log_result, 'base64'), 'utf-8') as log FROM
 aws_lambda.invoke(:'aws_lambda_arn_1', '{"body": "Hello from Postgres!"}'::json,
 'RequestResponse', 'Tail');
```

Pour inclure le contexte client, utilisez un objet JSON pour le paramètre aws_lambda.invoke de la fonction `context`.

Pour plus d'informations sur le paramètre `context`, consultez la référence aws_lambda.invoke.

Exemple : appel d'une version spécifique d'une fonction Lambda

Vous pouvez spécifier une version particulière d'une fonction Lambda en incluant le paramètre `qualifier` avec l'appel `aws_lambda.invoke`. Vous trouverez ci-dessous un exemple de ce procédé qui utilise '*custom_version*' comme alias pour la version.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json, 'RequestResponse', 'None', NULL, 'custom_version');
```

Vous pouvez également fournir un qualificatif de fonction Lambda avec les informations de nom de la fonction à la place, comme suit.

```
SELECT * FROM aws_lambda.invoke(aws_commons.create_lambda_function_arn('my-
function:custom_version', 'us-west-2'),
'{"body": "Hello from Postgres!"}'::json);
```

Pour plus d'informations sur `qualifier` et d'autres paramètres, consultez la référence aws_lambda.invoke.

Messages d'erreur de fonction Lambda

Dans la liste suivante, vous trouverez des informations sur les messages d'erreur, avec les causes et les solutions possibles.

- Problèmes de configuration de VPC

  Les problèmes de configuration du VPC peuvent entraîner les messages d'erreur suivants lors de la tentative de connexion :

  ```
  ERROR:  invoke API failed
  DETAIL: AWS Lambda client returned 'Unable to connect to endpoint'.
  CONTEXT:  SQL function "invoke" statement 1
  ```

  Une cause fréquente de cette erreur est un groupe de sécurité VPC mal configuré. Assurez-vous que vous disposez d'une règle sortante pour TCP ouverte sur le port 443 de votre groupe de sécurité VPC afin que votre VPC puisse se connecter au VPC Lambda.
- Manque d'autorisations nécessaires pour appeler les fonctions Lambda

Si l'un des messages d'erreur suivants s'affiche, l'utilisateur (rôle) qui appelle la fonction ne dispose pas des autorisations nécessaires.

```
ERROR:  permission denied for schema aws_lambda
```

```
ERROR:  permission denied for function invoke
```

Un utilisateur (rôle) doit recevoir des autorisations spécifiques pour appeler les fonctions Lambda. Pour de plus amples informations, consultez Étape 6 : accorder aux autres utilisateurs l'autorisation d'appeler les fonctions Lambda.

- Traitement inapproprié des erreurs dans vos fonctions Lambda4

  Si une fonction Lambda lance une exception pendant le traitement de la demande, `aws_lambda.invoke` échoue avec une erreur PostgreSQL telle que la suivante.

```
SELECT * FROM aws_lambda.invoke('aws_lambda_arn_1', '{"body": "Hello from
 Postgres!"}'::json);
ERROR:  lambda invocation failed
DETAIL:  "arn:aws:lambda:us-west-2:555555555555:function:my-function" returned error
 "Unhandled", details: "<Error details string>".
```

  Assurez-vous de gérer les erreurs dans vos fonctions Lambda ou dans votre application PostgreSQL.

Référence des fonctions et des paramètres AWS Lambda

Voici la référence pour les fonctions et les paramètres à utiliser pour invoquer Lambda avec Aurora PostgreSQL .

Fonctions et paramètres

- aws_lambda.invoke
- aws_commons.create_lambda_function_arn
- Paramètres aws_lambda

aws_lambda.invoke

Exécute une fonction Lambda pour un cluster de bases de données Aurora PostgreSQL .

Pour plus de détails sur l'appel de fonctions Lambda, consultez également la section Appel dans le Manuel du développeur AWS Lambda.

Syntaxe

JSON

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSON,
IN region TEXT DEFAULT NULL,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSON,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSON DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSON,
OUT executed_version TEXT,
OUT log_result TEXT)
```

JSONB

```
aws_lambda.invoke(
IN function_name TEXT,
IN payload JSONB,
IN region TEXT DEFAULT NULL,
```

```
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT)
```

```
aws_lambda.invoke(
IN function_name aws_commons._lambda_function_arn_1,
IN payload JSONB,
IN invocation_type TEXT DEFAULT 'RequestResponse',
IN log_type TEXT DEFAULT 'None',
IN context JSONB DEFAULT NULL,
IN qualifier VARCHAR(128) DEFAULT NULL,
OUT status_code INT,
OUT payload JSONB,
OUT executed_version TEXT,
OUT log_result TEXT
)
```

Paramètres d'entrée

function_name

Nom d'identification de la fonction Lambda. La valeur peut être le nom de la fonction, un ARN ou un ARN partiel. Pour obtenir la liste des formats possibles, consultez Formats de nom de fonction Lambda dans le Manuel du développeur AWS Lambda.

payload

Entrée de la fonction Lambda. Le format peut être JSON ou JSONB. Pour plus d'informations, consultez la documentation PostgreSQL sur les types JSON.

région

(Facultatif) Région Lambda de la fonction. Par défaut, Aurora résout la Région AWS à partir de l'ARN complet dans le `function_name` ou utilise la Région de l'instance de base de données Aurora PostgreSQL. Si cette valeur de région est en conflit avec celle fournie dans l'ARN `function_name`, une erreur est déclenchée.

invocation_type

Type d'appel de la fonction Lambda. La valeur est sensible à la casse. Les valeurs possibles sont notamment les suivantes :

- `RequestResponse` – Valeur par défaut Ce type d'appel d'une fonction Lambda est synchrone et renvoie une charge utile de réponse dans le résultat. Utilisez le type d'appel `RequestResponse` lorsque votre flux de travail dépend de la réception immédiate du résultat de la fonction Lambda.
- `Event` – Ce type d'appel d'une fonction Lambda est asynchrone et retourne une réponse immédiatement sans retourner de charge utile. Utilisez le type d'appel `Event` lorsque vous n'avez pas besoin des résultats de la fonction Lambda avant que votre flux de travail ne progresse.
- `DryRun` – Ce type d'appel teste l'accès sans exécuter la fonction Lambda.

log_type

Type de journal Lambda à renvoyer dans le paramètre de sortie `log_result`. La valeur est sensible à la casse. Les valeurs possibles sont notamment les suivantes :

- Tail – Le paramètre de sortie `log_result` renvoyé inclura les 4 derniers Ko du journal d'exécution.
- None – Aucune information de journal Lambda n'est renvoyée.

context

Contexte client au format JSON ou JSONB. Les champs à utiliser incluent alors `custom` et `env`.

qualifier

Qualificateur qui identifie la version d'une fonction Lambda à appeler. Si cette valeur est en conflit avec celle fournie dans l'ARN `function_name`, une erreur est déclenchée.

Paramètres de sortie

status_code

Code de réponse d'état HTTP. Pour plus d'informations, consultez Éléments de réponse à l'appel de la fonction Lambda dans le AWS LambdaManuel du développeur .

payload

Informations renvoyées à partir de la fonction Lambda exécutée. Le format est en JSON ou JSONB.

executed_version

Version de la fonction Lambda exécutée.

log_result

Informations du journal d'exécution renvoyées si la valeur `log_type` est `Tail` lorsque la fonction Lambda a été appelée. Le résultat contient les 4 derniers Ko du journal d'exécution codé en Base64.

aws_commons.create_lambda_function_arn

Crée une structure `aws_commons._lambda_function_arn_1` pour contenir les informations de nom de fonction Lambda. Vous pouvez utiliser les résultats de la fonction `aws_commons.create_lambda_function_arn` dans le paramètre `function_name` de la fonction aws_lambda.invoke.

Syntaxe

```
aws_commons.create_lambda_function_arn(
    function_name TEXT,
    region TEXT DEFAULT NULL
    )
    RETURNS aws_commons._lambda_function_arn_1
```

Paramètres d'entrée

function_name

Chaîne de texte obligatoire contenant le nom de la fonction Lambda. La valeur peut être un nom de fonction, un ARN partiel ou un ARN complet.

région

Chaîne de texte facultative contenant la région AWS dans laquelle se trouve la fonction Lambda. Pour obtenir la liste des noms de régions et les valeurs associées, consultez Régions et zones de disponibilité.

Paramètres aws_lambda

Dans le tableau, vous pouvez trouver les paramètres associés à la fonction `aws_lambda`.

Paramètre	Description
`aws_lambda.connect_timeout_ms`	Il s'agit d'un paramètre dynamique qui définit le temps d'attente maximal lors de la connexion à AWS Lambda. Les valeurs par défaut sont `1000`. Les valeurs autorisées pour ce paramètre sont comprises entre 1 et 900 000.
`aws_lambda.request_timeout_ms`	Il s'agit d'un paramètre dynamique qui définit le temps d'attente maximal lors de la réception d'une réponse d'AWS Lambda. Les valeurs par défaut sont `3000`. Les valeurs autorisées pour ce paramètre sont comprises entre 1 et 900 000.
`aws_lambda.endpoint_override`	Indique le point de terminaison qui peut être utilisé pour se connecter à AWS Lambda. Une chaîne vide sélectionne le point de terminaison AWS Lambda par défaut pour la région. La base de données doit être redémarrée pour que ce changement de paramètre statique soit pris en compte.

Publication de journaux Aurora PostgreSQL sur Amazon CloudWatch Logs

Vous pouvez configurer votre cluster de bases de données Aurora PostgreSQL pour exporter régulièrement des données de journal vers Amazon CloudWatch Logs. Dans ce cas, les événements du journal PostgreSQL de votre cluster de bases de données Aurora PostgreSQL sont automatiquement publiés sur Amazon CloudWatch, en tant que Amazon CloudWatch Logs. Dans CloudWatch, vous pouvez trouver les données de journal exportées dans un Log group (Groupe de journaux) pour votre cluster de bases de données Aurora PostgreSQL. Le groupe de journaux contient un ou plusieurs flux de journaux qui contiennent les événements du journal PostgreSQL de chaque instance du cluster.

La publication des journaux dans CloudWatch Logs vous permet de conserver les enregistrements des journaux PostgreSQL de votre cluster dans un stockage hautement durable. Grâce aux données de journal disponibles dans CloudWatch Logs, vous pouvez évaluer et améliorer les opérations de votre cluster. Vous pouvez également utiliser CloudWatch pour créer des alarmes et afficher des

métriques. Pour en savoir plus, consultez Surveillance des événements de journaux dans Amazon CloudWatch.

> **Note**
>
> La publication de vos journaux PostgreSQL sur CloudWatch Logs consomme du stockage, et vous devez payer pour ce stockage. Veillez à supprimer tous les CloudWatch Logs dont vous n'avez plus besoin.

Désactiver l'option de journal d'exportation pour un cluster de bases de données Aurora PostgreSQL existant n'affecte pas les données qui sont déjà contenues dans CloudWatch Logs. Les journaux existants restent disponibles dans CloudWatch Logs en fonction de vos paramètres de conservation des journaux. Pour en savoir plus sur CloudWatch Logs, consultez What is Amazon CloudWatch Logs? (Qu'est-ce qu'Amazon CloudWatch Logs ?)

Aurora PostgreSQL prend en charge la publication de journaux dans CloudWatch Logs pour les versions suivantes.

- 16.1 et toutes les versions ultérieures
- 15.2 et versions 15 ultérieures
- 14.3 et versions 14 ultérieures
- 13.3 et versions 13 ultérieures
- 12.8 et versions 12 ultérieures
- 11.9 et versions 11 ultérieures

Pour plus d'informations sur l'activation de l'option de publication des journaux sur CloudWatch Logs, la surveillance des événements des journaux dans CloudWatch Logs et l'analyse des journaux à l'aide de CloudWatch Logs Insights, consultez les rubriques suivantes.

Rubriques

- Activation de l'option de publication des journaux vers Amazon CloudWatch
- Surveillance des événements de journaux dans Amazon CloudWatch
- Analyse des journaux PostgreSQL à l'aide de CloudWatch Logs Insights

Activation de l’option de publication des journaux vers Amazon CloudWatch

Pour publier le journal PostgreSQL de votre cluster de bases de données Aurora PostgreSQL dans CloudWatch Logs, choisissez l’option Log export (Exportation de journal) pour le cluster. Vous pouvez choisir le paramètre d’exportation du journal lorsque vous créez votre cluster de bases de données Aurora PostgreSQL. Ou bien, vous pouvez modifier le cluster ultérieurement. Lorsque vous modifiez un cluster existant, ses journaux PostgreSQL de chaque instance sont publiés dans le cluster CloudWatch à partir de ce moment-là. Pour Aurora PostgreSQL, le journal PostgreSQL (`postgresql.log`) est le seul journal qui est publié sur Amazon CloudWatch.

Vous pouvez utiliser la AWS Management Console, AWS CLI, ou l’API RDS pour activer la fonction d’exportation de journaux pour votre cluster de bases de données Aurora PostgreSQL.

Console

Vous choisissez l’option d’exportation des journaux pour commencer à publier les journaux PostgreSQL de votre cluster de bases de données Aurora PostgreSQL vers CloudWatch Logs.

Pour activer la fonction d’exportation des journaux à partir de la console

1. Ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Choisissez le cluster de bases de données Aurora PostgreSQL dont vous voulez publier les données de journal dans CloudWatch Logs.
4. Sélectionnez Modify.
5. Dans la section Log exports (Exportations des journaux), choisissez PostgreSQL log (Journal Postgresql).
6. Choisissez Continuer, puis Modifier le cluster sur la page récapitulative.

AWS CLI

Vous pouvez activer l’option d’exportation des journaux pour commencer à publier les journaux d’Aurora PostgreSQL vers Amazon CloudWatch Logs avec AWS CLI. Pour cela, exécutez la commande modify-db-cluster de AWS CLI avec les options suivantes :

- `--db-cluster-identifier` — Identifiant du cluster de bases de données.
- `--cloudwatch-logs-export-configuration` — Paramètre de configuration des types de journaux à définir pour l’exportation vers CloudWatch Logs pour le cluster de bases de données.

Vous pouvez également publier des journaux Aurora PostgreSQL en exécutant l'une des commandes de l'AWS CLI suivantes :

- create-db-cluster
- restore-db-cluster-from-s3
- restore-db-cluster-from-snapshot
- restore-db-cluster-to-point-in-time

Exécutez l'une de ces commandes de l'AWS CLI avec les options suivantes :

- `--db-cluster-identifier` — Identifiant du cluster de bases de données.
- `--engine` — Moteur de base de données.
- `--enable-cloudwatch-logs-exports` — Paramètre de configuration des types de journaux à activer pour l'exportation vers CloudWatch Logs pour le cluster de bases de données.

D'autres options peuvent être requises en fonction de la commande d'AWS CLI que vous exécutez.

Example

La commande suivante crée un cluster de bases de données Aurora MySQL qui publie des fichiers journaux sur CloudWatch Logs.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --engine aurora-postgresql \
    --enable-cloudwatch-logs-exports postgresql
```

Pour Windows :

```
aws rds create-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --engine aurora-postgresql ^
    --enable-cloudwatch-logs-exports postgresql
```

Example

La commande suivante modifie un cluster de bases de données Aurora PostgreSQL existant afin qu'il publie des fichiers journaux sur CloudWatch Logs. La valeur `--cloudwatch-logs-export-configuration` n'est pas un objet JSON. La clé de cet objet est `EnableLogTypes`, et sa valeur est `postgresql`, ainsi que `instance`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-db-cluster \
    --cloudwatch-logs-export-configuration '{"EnableLogTypes":
["postgresql","instance"]}'
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier my-db-cluster ^
    --cloudwatch-logs-export-configuration '{\"EnableLogTypes\":[\"postgresql\",
\"instance\"]}'
```

> **Note**
>
> Lorsque vous utilisez l'invite de commande Windows, veillez à utiliser des guillemets doubles (") d'échappement dans le code JSON en les préfixant d'une barre oblique inverse (\).

Example

L'exemple suivant modifie un cluster de bases de données Aurora PostgreSQL existant afin de désactiver la publication de fichiers journaux sur CloudWatch Logs. La valeur `--cloudwatch-logs-export-configuration` n'est pas un objet JSON. La clé de cet objet est `DisableLogTypes`, et sa valeur est `postgresql`, ainsi que `instance`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbinstance \
    --cloudwatch-logs-export-configuration '{"DisableLogTypes":
["postgresql","instance"]}'
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbinstance ^
    --cloudwatch-logs-export-configuration "{\"DisableLogTypes\":[\"postgresql\",
\"instance\"]}"
```

> **Note**
>
> Lorsque vous utilisez l'invite de commandes Windows, vous devez utiliser des guillemets doubles (") d'échappement dans le code JSON en les préfixant d'une barre oblique inverse (\).

API RDS

Vous pouvez activer l'option d'exportation des journaux pour commencer à publier les journaux d'Aurora PostgreSQL avec l'API RDS. Pour cela, exécutez l'opération ModifyDBCluster avec les options suivantes :

- `DBClusterIdentifier` — Identifiant du cluster de bases de données.
- `CloudwatchLogsExportConfiguration` — Paramètre de configuration des types de journaux à activer pour l'exportation vers CloudWatch Logs pour le cluster de bases de données.

Vous pouvez également publier des journaux Aurora PostgreSQL avec l'API RDS en exécutant l'une des opérations d'API RDS suivantes :

- CreateDBCluster
- RestoreDBClusterFromS3
- RestoreDBClusterFromSnapshot
- RestoreDBClusterToPointInTime

Exécutez l'action de l'API RDS avec les paramètres suivants :

- `DBClusterIdentifier` — Identifiant du cluster de bases de données.
- `Engine` — Moteur de base de données.

- `EnableCloudwatchLogsExports` — Paramètre de configuration des types de journaux à activer pour l'exportation vers CloudWatch Logs pour le cluster de bases de données.

D'autres paramètres peuvent être requis en fonction de la commande d'AWS CLI que vous exécutez.

Surveillance des événements de journaux dans Amazon CloudWatch

Les événements du journal d'Aurora PostgreSQL étant publiés et disponibles sous forme de journaux Amazon CloudWatch Logs, vous pouvez visualiser et surveiller les événements à l'aide d'Amazon CloudWatch. Pour plus d'informations sur la surveillance, consultez Affichage des données de journaux envoyées à Cloudwatch Logs.

Lorsque vous activez les exportations de journaux, un nouveau groupe de journaux est automatiquement créé en utilisant le préfixe `/aws/rds/cluster/` avec le nom de votre Aurora PostgreSQL et le type de journal, comme dans le modèle suivant.

```
/aws/rds/cluster/your-cluster-name/postgresql
```

À titre d'exemple, supposons qu'un cluster de bases de données Aurora PostgreSQL nommé `docs-lab-apg-small` exporte son journal vers Amazon CloudWatch Logs. Son nom de groupe de journaux dans Amazon CloudWatch est indiqué ci-dessous.

```
/aws/rds/cluster/docs-lab-apg-small/postgresql
```

Si un groupe de journaux de ce nom existe déjà, Aurora l'utilise pour exporter les données de journaux du cluster de bases de données Aurora. Chaque instance de base de données dans le cluster de bases de données Aurora PostgreSQL charge son journal PostgreSQL dans le groupe de journaux en tant que flux de journaux distinct. Vous pouvez examiner le groupe de journaux et ses flux de journaux à l'aide des différents outils graphiques et analytiques disponibles dans Amazon CloudWatch.

Par exemple, vous pouvez rechercher des informations dans les événements du journal de votre cluster de bases de données Aurora PostgreSQL et filtrer les événements à l'aide de la console CloudWatch Logs, de AWS CLI, ou de l'API CloudWatch Logs. Pour plus d'informations, consultez Searching and filtering log data (Recherche et filtrage des données de journal) dans le Guide de l'utilisateur d'Amazon CloudWatch Logs

Par défaut, les nouveaux groupes de journaux sont créés en utilisant l'option Never expire (Ne jamais expirer) pour leur période de conservation. Vous pouvez utiliser la console CloudWatch Logs, l'AWS

CLI ou l'API CloudWatch Logs pour modifier la période de conservation des journaux. Pour en savoir plus, consultez Change log data retention in CloudWatch Logs (Modification de la rétention des données de journaux dans CloudWatch Logs) dans le Guide de l'utilisateur Amazon CloudWatch Logs.

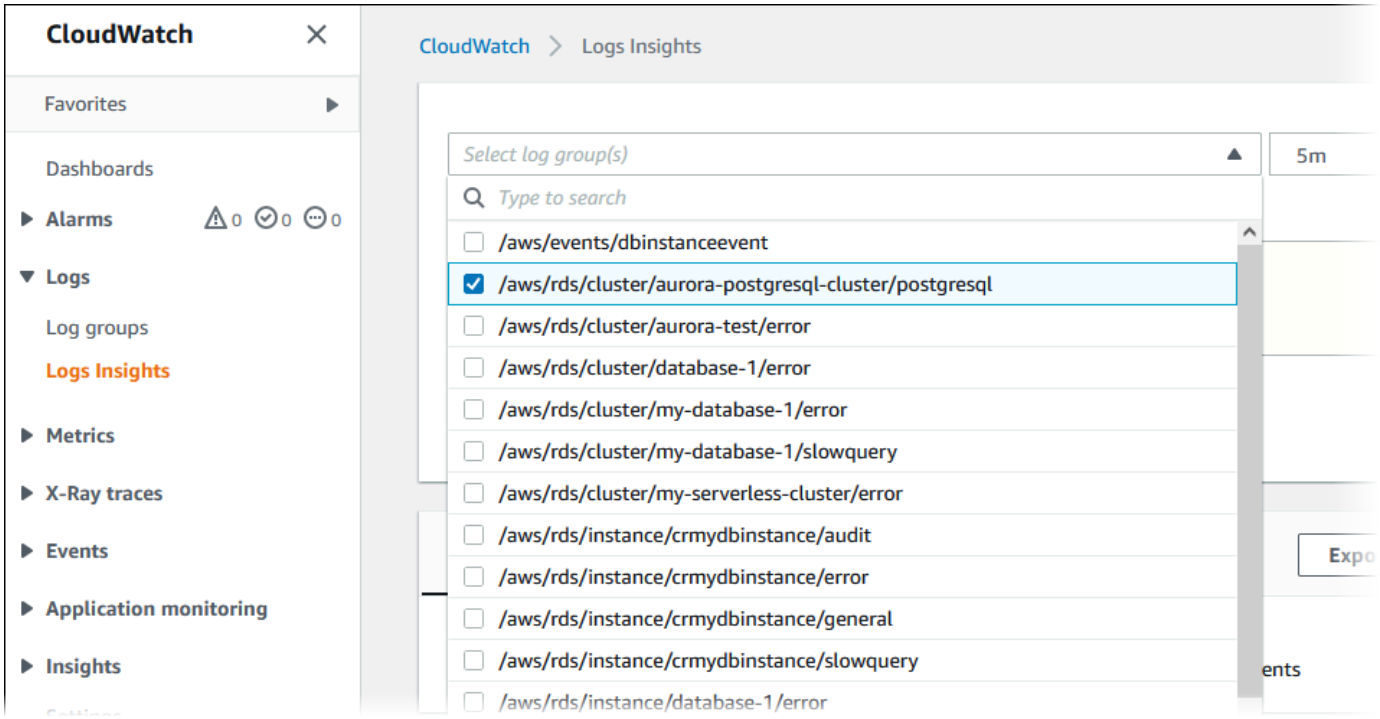
> **Tip**
>
> Vous pouvez utiliser une configuration automatique, telle que AWS CloudFormation, pour créer des groupes de journaux avec des périodes de rétention de journaux, des filtres de métriques et des autorisations d'accès personnalisés.

Analyse des journaux PostgreSQL à l'aide de CloudWatch Logs Insights

Avec les journaux PostgreSQL de votre cluster de bases de données Aurora PostgreSQL publiés en tant que journaux CloudWatch, vous pouvez utiliser CloudWatch Logs Insights pour rechercher et analyser de manière interactive vos données de journal dans Amazon CloudWatch Logs. CloudWatch Logs Insights comprend un langage de requête, des exemples de requêtes et d'autres outils pour analyser vos données de journal afin que vous puissiez identifier les problèmes potentiels et vérifier les corrections. Pour en savoir plus, consultez Analyzing log data with CloudWatch Logs Insights (Analyser les données des journaux avec CloudWatch Logs Insights) dans le Guide de l'utilisateur d'Amazon CloudWatch Logs

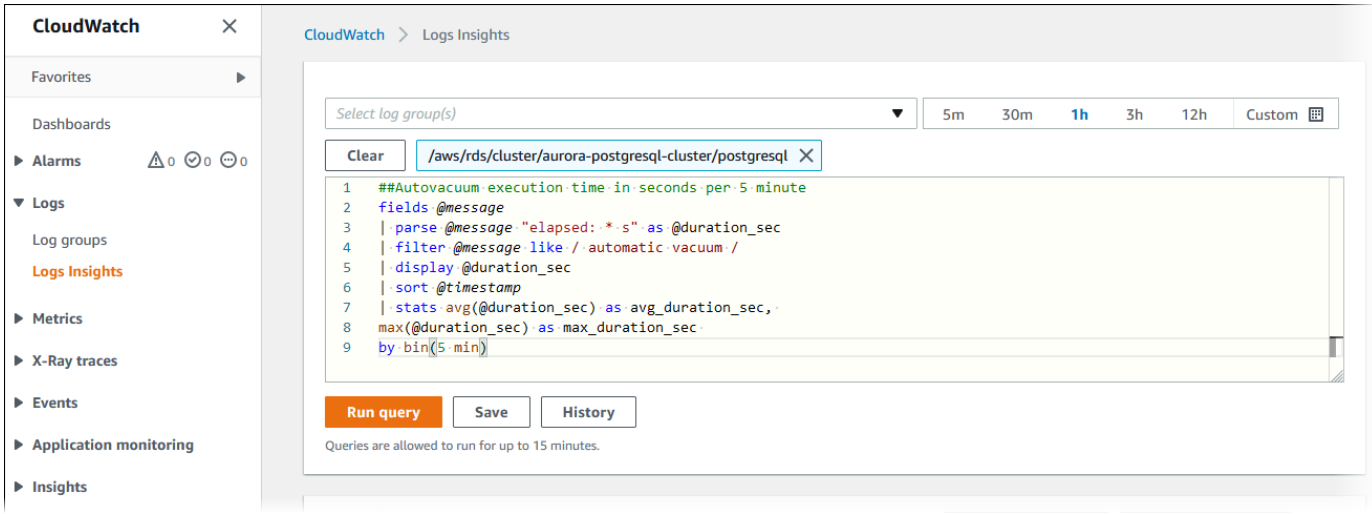
Pour analyser des journaux PostgreSQL à l'aide de CloudWatch Logs Insights

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation, ouvrez Logs (Journaux) et choisissez Log Insights.
3. Dans Select log group(s) (Sélectionner des groupes de journaux), sélectionnez le groupe de journaux pour votre cluster de bases de données Aurora PostgreSQL.

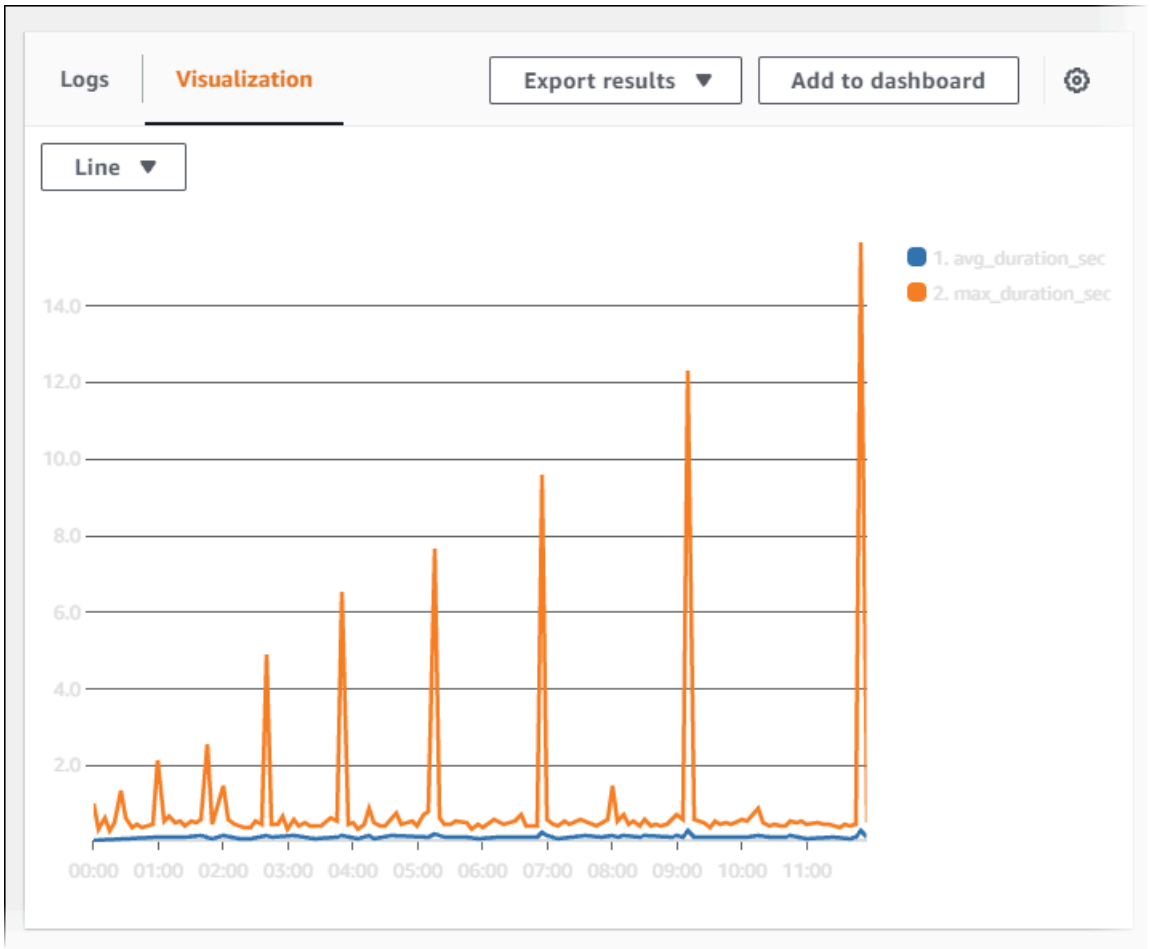


4. Dans l'éditeur de requête, supprimez la requête affichée, saisissez les informations suivantes, puis choisissez Run query (Exécuter la requête).

```
##Autovacuum execution time in seconds per 5 minute
fields @message
| parse @message "elapsed: * s" as @duration_sec
| filter @message like / automatic vacuum /
| display @duration_sec
| sort @timestamp
| stats avg(@duration_sec) as avg_duration_sec,
max(@duration_sec) as max_duration_sec
by bin(5 min)
```

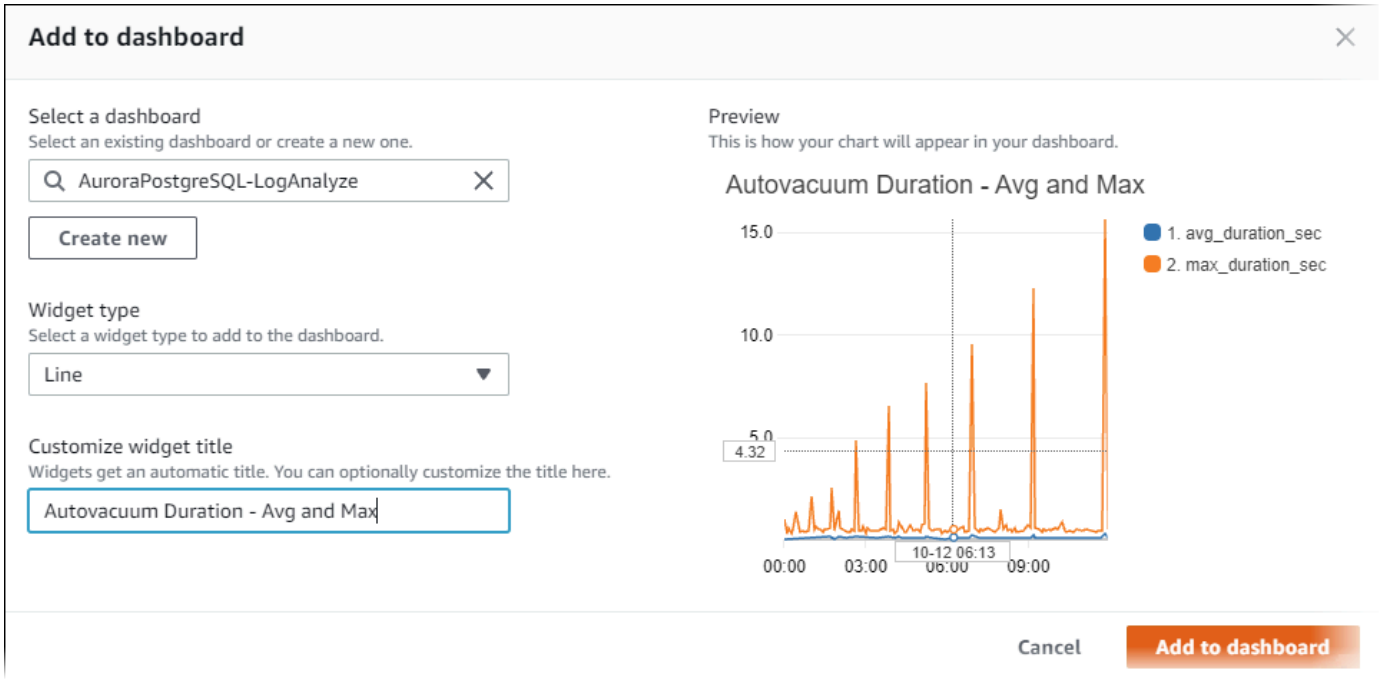


5. Choisissez l'onglet Visualisation.



6. Choisissez Add to dashboard (Ajouter au tableau de bord).
7. Dans Select a dashboard (Sélectionner un tableau de bord), sélectionnez un tableau de bord ou saisissez un nom pour créer un tableau de bord.

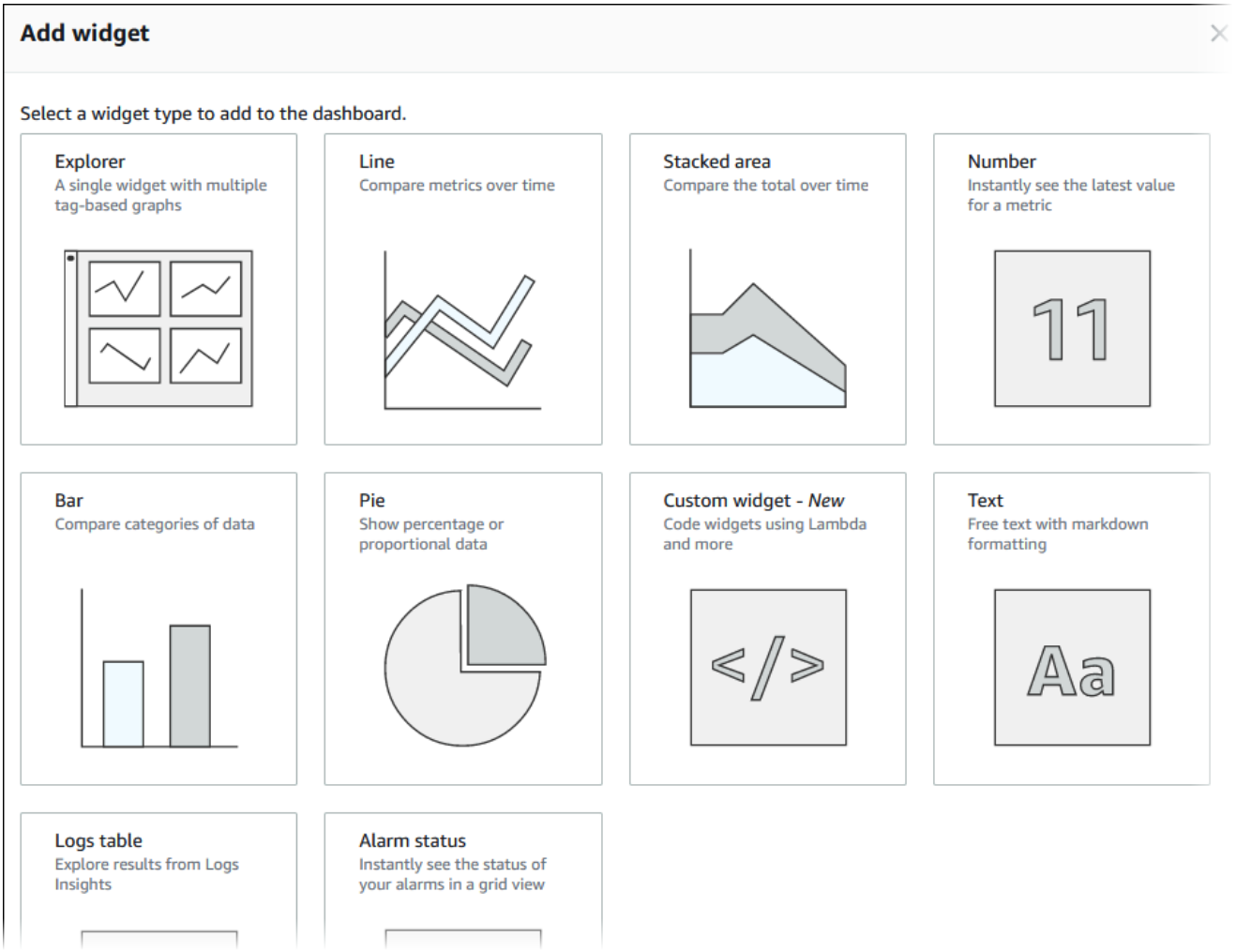
8. Dans Widget type (Type de widget), choisissez un type de widget pour votre visualisation.



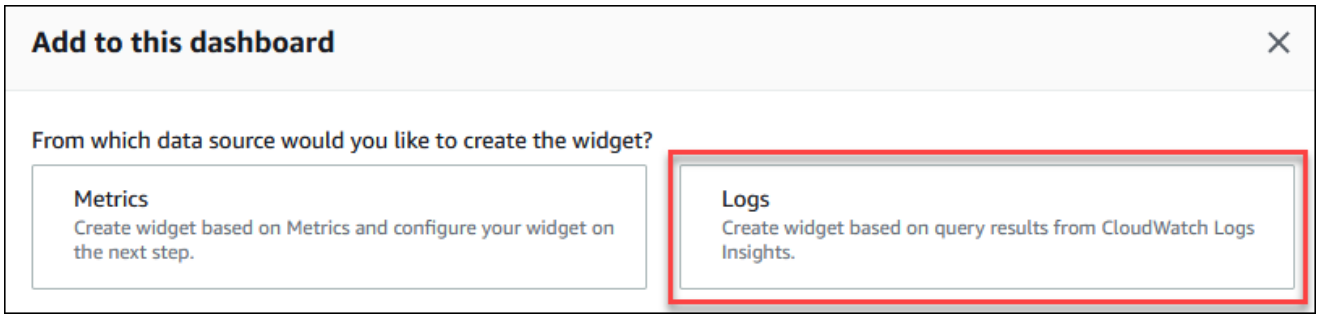
9. (Facultatif) Ajoutez d'autres widgets en fonction des résultats de vos requêtes de journaux.

   a. Sélectionnez Add widget (Ajouter un widget).

   b. Choisissez un type de widget, tel que Line (Ligne).



c. Dans la fenêtre Add to this dashboard (Ajouter à ce tableau de bord), choisissez Logs (Journaux).

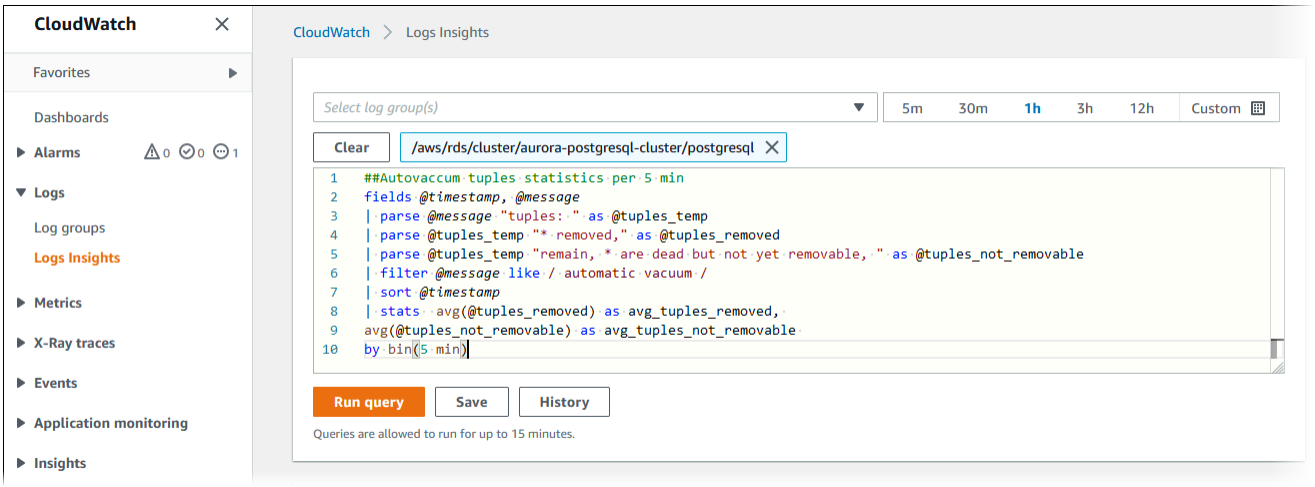


d. Dans Select log group(s) (Sélectionner des groupes de journaux), sélectionnez le groupe de journaux pour votre cluster de bases de données.

e. Dans l'éditeur de requête, supprimez la requête affichée, saisissez les informations suivantes, puis choisissez Run query (Exécuter la requête).

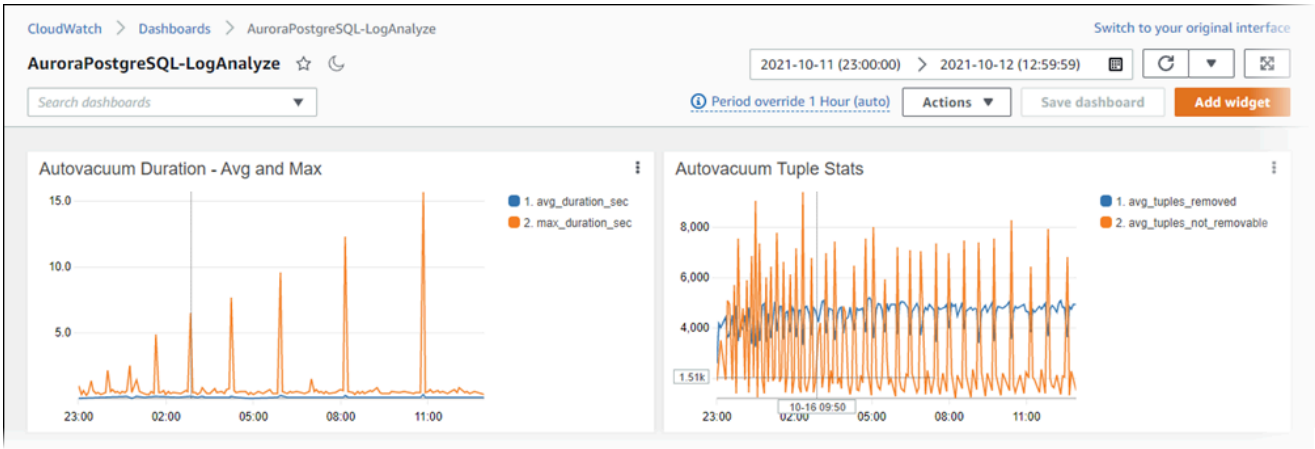
```
##Autovacuum tuples statistics per 5 min
```

```
fields @timestamp, @message
| parse @message "tuples: " as @tuples_temp
| parse @tuples_temp "* removed," as @tuples_removed
| parse @tuples_temp "remain, * are dead but not yet removable, " as
 @tuples_not_removable
| filter @message like / automatic vacuum /
| sort @timestamp
| stats  avg(@tuples_removed) as avg_tuples_removed,
avg(@tuples_not_removable) as avg_tuples_not_removable
by bin(5 min)
```



f. Choisissez Create widget (Créer un widget).

Votre tableau de bord doit ressembler à l'image suivante.



Surveillance des plans d’exécution des requêtes et des pics de mémoire pour Aurora PostgreSQL

Vous pouvez surveiller les plans d’exécution des requêtes dans votre instance de base de données Aurora PostgreSQL afin de détecter ceux qui contribuent à la charge actuelle de la base de données et de suivre leurs statistiques de performance au fil du temps à l’aide du paramètre `aurora_compute_plan_id`. Chaque fois qu’une requête est exécutée, un identifiant est attribué au plan d’exécution utilisé par cette requête. Cet identifiant sera utilisé lors des exécutions ultérieures du même plan.

L’`aurora_compute_plan_id` est désactivé (`OFF`) par défaut dans le groupe de paramètres de base de données à partir d’Aurora PostgreSQL 14.10, 15.5 et versions ultérieures. Pour attribuer un identifiant de plan, définissez `aurora_compute_plan_id` sur `ON` dans le groupe de paramètres.

Cet identifiant de plan est utilisé dans plusieurs utilitaires ayant un objectif différent.

Vous pouvez surveiller l’utilisation des pics de mémoire des requêtes dans votre instance de base de données afin de détecter les requêtes qui contribuent à une utilisation élevée de la mémoire de base de données à partir des versions suivantes :

- 16.3 et toutes les versions ultérieures
- 15.7 et versions ultérieures
- 14.12 et versions ultérieures

Chaque fois qu’une requête est exécutée, le pic de mémoire utilisé par la requête est suivi. Les requêtes sont généralement exécutées plusieurs fois. Des valeurs d’utilisation moyenne, minimale et maximale de la mémoire pour toutes les exécutions peuvent être consultées pour chaque requête.

Rubriques

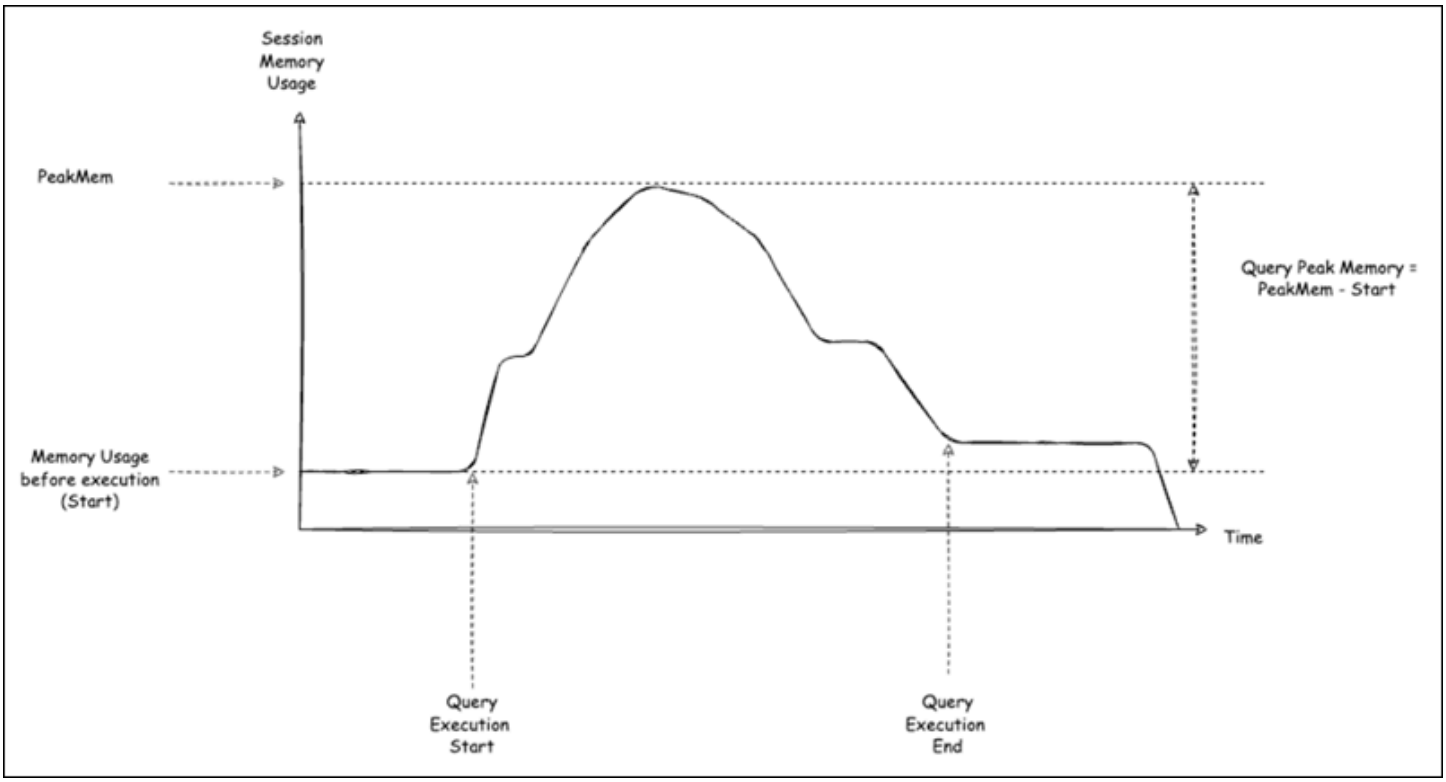
- Accès aux plans d’exécution des requêtes et aux pics de mémoire à l’aide des fonctions Aurora
- Référence des paramètres des plans d’exécution des requêtes Aurora PostgreSQL

Accès aux plans d'exécution des requêtes et aux pics de mémoire à l'aide des fonctions Aurora

Avec `aurora_compute_plan_id`, vous pouvez accéder aux plans d'exécution à l'aide des fonctions suivantes :

- aurora_stat_activity
- aurora_stat_plans

Les pics de mémoire des requêtes n'incluent pas la mémoire allouée avant le début du traitement des requêtes. L'utilisation des pics de mémoire est suivie et signalée séparément pour les phases de planification et d'exécution de chaque requête.



Vous pouvez accéder aux statistiques relatives aux pics de mémoire des requêtes à l'aide des fonctions suivantes :

- aurora_stat_statements
- aurora_stat_plans

Pour plus d'informations sur ces fonctions, consultez Référence sur les fonctions Aurora PostgreSQL.

Référence des paramètres des plans d’exécution des requêtes Aurora PostgreSQL

Vous pouvez utiliser les paramètres ci-dessous dans un groupe de paramètres de base de données pour surveiller les plans d’exécution des requêtes.

Paramètres

- aurora_compute_plan_id
- aurora_stat_plans.minutes_until_recapture
- aurora_stat_plans.calls_until_recapture
- aurora_stat_plans.with_costs
- aurora_stat_plans.with_analyze
- aurora_stat_plans.with_timing
- aurora_stat_plans.with_buffers
- aurora_stat_plans.with_wal
- aurora_stat_plans.with_triggers

> **Note**
>
> La configuration des paramètres `aurora_stat_plans.with_*` ne prend effet que pour les plans récemment capturés.

aurora_compute_plan_id

`aurora_compute_plan_id` est un paramètre de configuration qui contrôle si un identifiant de plan est attribué lors de l’exécution de la requête.

Par défaut	Valeurs autorisées	Description
off	0(désactivé)	Spécifiez `off` pour empêcher l’attribution d’un identifiant de plan.
	1(activé)	Spécifiez on pour attribuer un identifiant de plan.

aurora_stat_plans.minutes_until_recapture

Nombre de minutes qui s'écoulent avant qu'un plan ne soit capturé de nouveau. La valeur par défaut est de 0, ce qui désactive la recapture d'un plan. Lorsque le seuil `aurora_stat_plans.calls_until_recapture` est dépassé, le plan est capturé de nouveau.

Par défaut	Valeurs autorisées	Description
0	0-1073741823	Définissez le nombre de minutes qui doivent s'écouler avant qu'un plan ne soit capturé de nouveau.

aurora_stat_plans.calls_until_recapture

Nombre d'appels vers un plan avant qu'il ne soit capturé de nouveau. La valeur par défaut est de 0, ce qui désactive la recapture d'un plan après un certain nombre d'appels. Lorsque le seuil `aurora_stat_plans.minutes_until_recapture` est dépassé, le plan est capturé de nouveau.

Par défaut	Valeurs autorisées	Description
0	0-1073741823	Définissez le nombre d'appels avant qu'un plan ne soit capturé de nouveau.

aurora_stat_plans.with_costs

Capture un plan EXPLAIN avec une estimation des coûts. Les valeurs autorisées sont on et `off`. La valeur par défaut est on.

Par défaut	Valeurs autorisées	Description
on	0(désactivé)	N'affiche pas le coût estimé ni les lignes pour chaque nœud du plan.
	1(activé)	Affiche le coût estimé et les lignes pour chaque nœud du plan.

aurora_stat_plans.with_analyze

Contrôle le plan EXPLAIN avec ANALYZE. Ce mode n'est utilisé que lors de la première capture d'un plan. Les valeurs autorisées sont on et `off`. La valeur par défaut est `off`.

Par défaut	Valeurs autorisées	Description
off	0(désactivé)	N'inclut pas les statistiques d'exécution réelles du plan.
	1(activé)	Inclut les statistiques d'exécution réelles du plan.

aurora_stat_plans.with_timing

Le calendrier du plan est capturé dans EXPLAIN quand ANALYZE est utilisé. La valeur par défaut est on.

Par défaut	Valeurs autorisées	Description
on	0(désactivé)	N'inclut pas le temps de démarrage réel ni le temps passé dans chaque nœud du plan.
	1(activé)	Inclut le temps de démarrage réel et le temps passé dans chaque nœud du plan.

aurora_stat_plans.with_buffers

Les statistiques d'utilisation de la mémoire tampon du plan sont capturées dans EXPLAIN quand ANALYZE est utilisé. La valeur par défaut est `off`.

Par défaut	Valeurs autorisées	Description
off	0(désactivé)	N'inclut pas d'informations sur l'utilisation de la mémoire tampon.

Par défaut	Valeurs autorisées	Description
	1(activé)	Inclut des informations sur l'utilisation de la mémoire tampon.

aurora_stat_plans.with_wal

Les statistiques d'utilisation WAL du plan sont capturées dans EXPLAIN quand ANALYZE est utilisé. La valeur par défaut est `off`.

Par défaut	Valeurs autorisées	Description
off	0(désactivé)	N'inclut pas d'informations sur la génération d'enregistrements WAL.
	1(activé)	Inclut des informations sur la génération d'enregistrements WAL.

aurora_stat_plans.with_triggers

Les statistiques d'exécution du déclencheur du plan sont capturées dans EXPLAIN quand `ANALYZE` est utilisé. La valeur par défaut est `off`.

Par défaut	Valeurs autorisées	Description
off	0(désactivé)	N'inclut pas les statistiques d'exécution des déclencheurs.
	1(activé)	Inclut les statistiques d'exécution des déclencheurs.

Gestion des plans d’exécution de requêtes pour Aurora PostgreSQL

La gestion des plans de requêtes Aurora PostgreSQL est une fonction facultative que vous pouvez utiliser avec votre cluster de bases de données Édition compatible avec Amazon Aurora PostgreSQL. Cette fonction est packagée comme une extension `apg_plan_mgmt` que vous pouvez installer dans votre cluster de bases de données Aurora PostgreSQL. La gestion des plans de requêtes vous permet de gérer les plans d’exécution des requêtes générés par l’optimiseur pour vos applications SQL. L’extension AWS `apg_plan_mgmt` s’appuie sur la fonctionnalité native de traitement des requêtes du moteur de base de données PostgreSQL.

Vous trouverez ci-dessous des informations sur les fonctions de gestion des plans de requêtes Aurora PostgreSQL, leur configuration et leur utilisation avec votre cluster de bases de données Aurora PostgreSQL. Avant de commencer, nous vous recommandons de consulter les notes de mise à jour de la version spécifique de l’extension `apg_plan_mgmt` disponible pour votre version de PostgreSQL Aurora. Pour plus d’informations, consultez Versions de l’extension apg_plan_mgmt d’Aurora PostgreSQL dans Notes de mise à jour d’Aurora PostgreSQL.

Rubriques

Présentation de la gestion des plans de requêtes d'Aurora PostgreSQL

La gestion des plans de requêtes Aurora PostgreSQL est conçue pour garantir la stabilité du plan, quelles que soient les modifications apportées à la base de données qui peuvent entraîner une régression du plan de requête. La régression du plan de requête se produit lorsque l'optimiseur choisit un plan sous-optimal pour une instruction SQL donnée après des modifications du système ou de la base de données. Les changements en termes de statistiques, de contraintes, de paramètres d'environnement, de liaisons de paramètres de requêtes et de mises à niveau du moteur de base de données PostgreSQL peuvent tous provoquer une régression de plan.

Avec la gestion des plans de requêtes Aurora PostgreSQL, vous pouvez contrôler quand et comment les plans d'exécution de requêtes changent. Les avantages de la gestion de plans de requêtes Aurora PostgreSQL incluent ce qui suit.

- Optimisez la stabilité du plan en forçant l'optimiseur à opérer sa sélection parmi une poignée de plans de qualité connus.
- Optimisez les plans de manière centralisée, puis distribuez les meilleurs à l'échelle mondiale.
- Identifiez les index non utilisés et évaluez l'impact de la création ou de la suppression d'un index.
- Détectez automatiquement un nouveau plan à coût minimal découvert par l'optimiseur.
- Essayez les nouvelles fonctionnalités de l'optimiseur avec moins de risques, car vous pouvez choisir de n'approuver que les changements de plans qui optimisent les performances.

Vous pouvez utiliser les outils fournis par la gestion des plans de requêtes de manière proactive afin de définir le meilleur plan pour certaines requêtes. Vous pouvez également utiliser la gestion des plans de requêtes pour réagir à l'évolution des circonstances et éviter les régressions du plan. Pour plus d'informations, consultez Bonnes pratiques pour la gestion de plans de requêtes Aurora PostgreSQL.

Rubriques

- Désactivation de la gestion des plans de requêtes Aurora PostgreSQL

Instructions SQL prises en charge

La gestion des plans de requêtes prend en charge les types d'instructions SQL suivants.

- Instruction SELECT, INSERT, UPDATE ou DELETE, quelle que soit la complexité.
- Instructions préparées. Pour en savoir plus, consultez PREPARE dans la documentation PostgreSQL.
- Instructions dynamiques, y compris celles qui s'exécutent en mode immédiat. Pour plus d'informations, consultez Dynamic SQL et EXECUTE IMMEDIATE dans la documentation PostgreSQL.
- Commandes et instructions SQL intégrées. Pour en savoir plus, consultez Commandes SQL intégrées dans la documentation PostgreSQL.
- Instructions à l'intérieur de fonctions nommées. Pour plus d'informations, consultez CREATE FUNCTION dans la documentation PostgreSQL.
- Déclarations contenant des tables temporaires.
- Les instructions à l'intérieur des procédures et des blocs DO.

Vous pouvez utiliser la gestion des plans de requêtes avec `EXPLAIN` en mode manuel pour capturer un plan sans l'exécuter réellement. Pour plus d'informations, consultez Analyse du plan choisi par l'optimiseur. Pour en savoir plus sur les modes de gestion des plans de requêtes (manuel, automatique), consultez Capture des plans d'exécution d'Aurora PostgreSQL.

La gestion des plans de requêtes Aurora PostgreSQL prend en charge toutes les fonctionnalités du langage PostgreSQL, notamment les tables partitionnées, l'héritage, la sécurité au niveau des lignes et les expressions récursives de table commune. Pour en savoir plus sur ces fonctionnalités du langage PostgreSQL, consultez Partitionnement de tables, Politiques de sécurité des lignes et Requêtes WITH (expressions de table communes) et d'autres rubriques de la documentation de PostgreSQL.

Pour plus d'informations sur les différentes versions de la fonction de gestion du plan de requête d'Aurora PostgreSQL, consultez Versions de l'extension apg_plan_mgmt d'Aurora PostgreSQL dans Notes de mise à jour d'Aurora PostgreSQL.

Limites de la gestion des plans de requêtes

La version actuelle de la gestion du plan de requête d'Aurora PostgreSQL présente les limites suivantes.

- Les plans ne sont pas capturés pour les déclarations qui font référence aux relations du système : les déclarations qui font référence aux relations du système, telles que `pg_class`, ne sont pas capturées. Ceci est conçu pour empêcher la capture d'un grand nombre de plans générés par le système et utilisés en interne. Cela s'applique également aux tables système à l'intérieur des vues.
- Une classe d'instance de base de données plus importante peut être nécessaire pour votre cluster de bases de données Aurora PostgreSQL – En fonction de la charge de travail, la gestion des plans de requêtes peut nécessiter une classe d'instance de base de données présentant plus de 2 vCPU. Le nombre de `max_worker_processes` est limité par la taille de la classe d'instance de base de données. Le nombre de `max_worker_processes` fournis par une classe d'instance de base de données à 2 vCPU (db.t3.medium, par exemple) peut ne pas être suffisant pour une charge de travail donnée. Nous vous recommandons de choisir une classe d'instance de base de données avec plus de 2 vCPU pour votre cluster de bases de données Aurora PostgreSQL si vous utilisez la gestion des plans de requêtes.

  Lorsque la classe d'instance de base de données ne peut pas supporter la charge de travail, la gestion du plan de requête affiche un message d'erreur comme suit.

  ```
  WARNING: could not register plan insert background process
  HINT: You may need to increase max_worker_processes.
  ```

  Dans ce cas, vous devez augmenter la taille de votre cluster de bases de données Aurora PostgreSQL à une taille de classe d'instance de base de données avec plus de mémoire. Pour plus d'informations, consultez Moteurs de base de données pris en charge pour les classes d'instance de base de données.
- Les plans déjà stockés dans les sessions ne sont pas affectés : la gestion des plans de requêtes permet d'influencer les plans de requêtes sans modifier le code de l'application. Toutefois, lorsqu'un plan générique est déjà stocké dans une session existante et que vous souhaitez modifier son plan de requêtes, vous devez d'abord définir `plan_cache_mode` sur `force_custom_plan` dans le groupe de paramètres du cluster de bases de données.
- `queryid` dans `apg_plan_mgmt.dba_plans` et `pg_stat_statements` peuvent diverger lorsque :

- Les objets sont supprimés et recréés après leur stockage dans apg_plan_mgmt.dba_plans.
- La table `apg_plan_mgmt.plans` est importée depuis un autre cluster.

Pour plus d'informations sur les différentes versions de la fonction de gestion du plan de requête d'Aurora PostgreSQL, consultez Versions de l'extension apg_plan_mgmt d'Aurora PostgreSQL dans Notes de mise à jour d'Aurora PostgreSQL.

Terminologie de la gestion des plans de requête

Les termes suivants sont utilisés tout au long de cette rubrique.

instruction gérée

Une instruction SQL capturée par l'optimiseur dans le cadre de la gestion des plans de requêtes. Une instruction gérée possède un ou plusieurs plans d'exécution de requêtes stockés dans la vue `apg_plan_mgmt.dba_plans`.

référence du plan

Ensemble de plans approuvés pour une instruction gérée donnée. C'est-à-dire tous les plans de l'instruction gérée dont la colonne `status` de la vue `dba_plan` indique « Approuvé ».

historique du plan

Ensemble de plans capturés pour une instruction gérée donnée. L'historique du plan contient tous les plans capturés pour l'instruction, quel que soit leur statut.

régression du plan de requêtes

Le cas où l'optimiseur choisit un plan moins optimal qu'avant une modification donnée de l'environnement de base de données, telle qu'une nouvelle version de PostgreSQL ou des modifications des statistiques.

Versions de la gestion de plans de requêtes Aurora PostgreSQL

La gestion des plans de requêtes est prise en charge par toutes les versions actuellement disponibles d'Aurora PostgreSQL. Pour plus d'informations, consultez la liste des Mises à jour d'Amazon Aurora PostgreSQL dans Notes de mise à jour d'Aurora PostgreSQL.

La fonctionnalité de gestion des plans de requêtes est ajoutée à votre cluster de bases de données Aurora PostgreSQL lorsque vous installez l'extension `apg_plan_mgmt`. Différentes versions

d'Aurora PostgreSQL prennent en charge différentes versions de l'extension `apg_plan_mgmt`. Nous vous recommandons de mettre à niveau l'extension de gestion des plans de requêtes vers la dernière version de votre version d'Aurora PostgreSQL.

> **Note**
>
> Pour les notes de mise à jour relatives à chaque version d'extension `apg_plan_mgmt`, consultez Versions d'extension apg_plan_mgmt d'Aurora PostgreSQL dans Notes de mise à jour d'Aurora PostgreSQL.

Vous pouvez identifier la version exécutée sur votre cluster en vous connectant à une instance à l'aide de `psql` et de la métacommande \dx pour répertorier les extensions, comme indiqué ci-dessous.

```
labdb=> \dx
                      List of installed extensions
     Name      | Version |    Schema     |                            Description
---------------+---------+---------------
+------------------------------------------------------------------------
 apg_plan_mgmt | 1.0     | apg_plan_mgmt | Amazon Aurora with PostgreSQL compatibility
 Query Plan Management
 plpgsql       | 1.0     | pg_catalog    | PL/pgSQL procedural language
(2 rows)
```

La sortie indique que ce cluster utilise la version 1.0 de l'extension. Seules certaines versions `apg_plan_mgmt` sont disponibles pour une version d'Aurora PostgreSQL donnée. Dans certains cas, vous devrez peut-être mettre à niveau le cluster de bases de données Aurora PostgreSQL vers une nouvelle version mineure ou appliquer un correctif afin de pouvoir effectuer la mise à niveau vers la version la plus récente de la gestion des plans de requêtes. La version 1.0 d'`apg_plan_mgmt` affichée dans la sortie provient d'un cluster de bases de données Aurora PostgreSQL version 10.17, pour lequel aucune version plus récente d'`apg_plan_mgmt` n' est disponible. Dans ce cas, le cluster de bases de données Aurora PostgreSQL doit être mis à niveau vers une version plus récente de PostgreSQL.

Pour plus d'informations sur la mise à niveau de votre cluster de bases de données Aurora PostgreSQL vers une nouvelle version de PostgreSQL, consultez Mises à jour du moteur de base de données pour Amazon Aurora PostgreSQL.

Pour savoir comment mettre à niveau l'extension `apg_plan_mgmt`, consultez Mise à niveau de la gestion des plans de requête d'Aurora PostgreSQL.

Activation de la gestion de plans de requêtes Aurora PostgreSQL

La configuration de la gestion des plans de requêtes pour votre cluster de bases de données Aurora PostgreSQL implique l'installation d'une extension et la modification de plusieurs paramètres de cluster de bases de données. Vous devez disposer d'autorisations `rds_superuser` pour installer l'extension `apg_plan_mgmt` et activer la fonction pour le cluster de bases de données Aurora PostgreSQL.

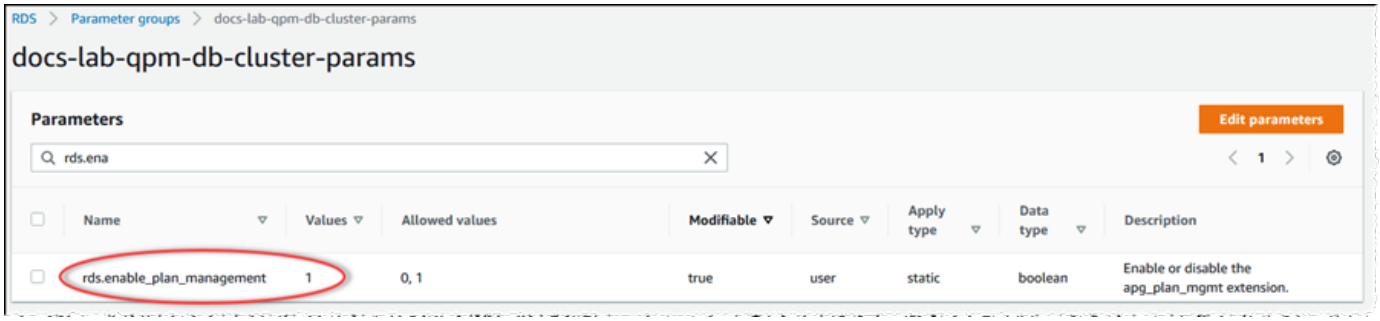
L'installation de l'extension crée un nouveau rôle, `apg_plan_mgmt`. Ce rôle permet aux utilisateurs de la base de données de consulter, de gérer et de gérer des plans de requêtes. En tant qu'administrateur doté de privilèges `rds_superuser`, veillez à attribuer le rôle `apg_plan_mgmt` aux utilisateurs de la base de données, selon leurs besoins.

Seuls les utilisateurs possédant le rôle `rds_superuser` peuvent effectuer la procédure suivante. Le rôle `rds_superuser` est nécessaire pour créer l'extension `apg_plan_mgmt` et son rôle `apg_plan_mgmt`. Les utilisateurs doivent recevoir le rôle `apg_plan_mgmt` pour administrer l'extension `apg_plan_mgmt`.

Activer la gestion des plans de requêtes pour votre cluster de bases de données Aurora PostgreSQL

Les étapes suivantes activent la gestion des plans de requêtes pour toutes les instructions SQL qui sont soumises au cluster de bases de données Aurora PostgreSQL. C'est ce que l'on appelle le mode automatique. Pour en savoir plus sur les différences entre les modes, consultez Capture des plans d'exécution d'Aurora PostgreSQL.

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Créez un groupe de paramètres de cluster de bases de données personnalisé pour le cluster de bases de données Aurora PostgreSQL. Vous devez modifier certains paramètres pour activer la gestion des plans de requêtes et définir son comportement. Pour plus d'informations, consultez Création d'un groupe de paramètres de base de données dans Amazon Aurora.
3. Ouvrez le groupe de paramètres de cluster de bases de données personnalisé et définissez le paramètre `rds.enable_plan_management` sur `1`, comme illustré dans l'image suivante.



Pour plus d'informations, consultez Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

4. Créez un groupe de paramètres de base de données personnalisé que vous pouvez utiliser pour définir les paramètres des plans de requêtes au niveau de l'instance. Pour plus d'informations, consultez Création d'un groupe de paramètres de cluster de bases de données dans Amazon Aurora.
5. Modifiez l'instance d'enregistreur du cluster de bases de données Aurora PostgreSQL pour utiliser le groupe de paramètres de base de données personnalisé. Pour plus d'informations, consultez Modification d'une instance de base de données dans un cluster de bases de données.
6. Modifiez le cluster de bases de données Aurora PostgreSQL pour utiliser le groupe de paramètres du cluster de bases de données personnalisé. Pour plus d'informations, consultez Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API.
7. Réinitialisez votre instance de base de données pour activer les paramètres des groupes de paramètres personnalisés.
8. Connectez-vous au point de terminaison de j'instance de base de données du cluster de bases de données Aurora PostgreSQL à l'aide de `psql` ou pgAdmin. L'exemple suivant utilise le compte `postgres` par défaut pour le rôle `rds_superuser`.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=my-db
```

9. Créez l'extension `apg_plan_mgmt` pour votre instance de base de données, comme indiqué ci-dessous.

```
labdb=> CREATE EXTENSION apg_plan_mgmt;
CREATE EXTENSION
```

> **Tip**
>
> Installez l'extension `apg_plan_mgmt` dans la base de données des modèles pour votre application. La base de données de modèles par défaut est nommée `template1`. Pour en savoir plus, consultez la section Bases de données modèles dans la documentation PostgreSQL.

10. Remplacez le paramètre `apg_plan_mgmt.capture_plan_baselines` par `automatic`. Ce paramètre permet à l'optimiseur de générer des plans pour chaque instruction SQL qui est soit planifiée, soit exécutée deux fois ou plus.

> **Note**
>
> La gestion des plans de requêtes dispose également d'un mode manuel que vous pouvez utiliser pour des instructions SQL spécifiques. Pour en savoir plus, consultez Capture des plans d'exécution d'Aurora PostgreSQL.

11. Remplacez la valeur du paramètre `apg_plan_mgmt.use_plan_baselines` par « on ». Ce paramètre force l'optimiseur à choisir un plan pour l'instruction à partir de sa référence de plan. Pour en savoir plus, consultez Utilisation des plans gérés Aurora PostgreSQL.

> **Note**
>
> Vous pouvez modifier la valeur de l'un ou l'autre de ces paramètres dynamiques pour la session sans avoir à redémarrer l'instance.

Lorsque la configuration de la gestion de votre plan de requêtes est terminée, veillez à attribuer le rôle `apg_plan_mgmt` à tous les utilisateurs de bases de données qui ont besoin de consulter, de gérer ou de gérer des plans de requêtes.

Mise à niveau de la gestion des plans de requête d'Aurora PostgreSQL

Nous vous recommandons de mettre à niveau l'extension de gestion des plans de requêtes vers la dernière version de votre version d'Aurora PostgreSQL.

1. Connectez-vous à l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL en tant qu'utilisateur avec des privilèges `rds_superuser`. Si vous avez conservé

le nom par défaut lors de la configuration de votre instance, vous vous connectez en tant que `postgres`. Cet exemple montre comment utiliser `psql`, mais vous pouvez également utiliser pgAdmin si vous préférez.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password
```

2. Exécutez la requête suivante pour mettre à niveau l'extension.

```
ALTER EXTENSION apg_plan_mgmt UPDATE TO '2.1';
```

3. Utilisez la fonction apg_plan_mgmt.validate_plans pour mettre à jour les hachages de tous les plans. L'optimiseur valide tous les plans approuvés, non approuvés et rejetés afin de s'assurer qu'ils sont toujours viables pour la nouvelle version de l'extension.

```
SELECT apg_plan_mgmt.validate_plans('update_plan_hash');
```

   Pour en savoir plus sur cette fonction, consultez Validation des plans.

4. Utilisez cette fonction apg_plan_mgmt.reload pour actualiser tous les plans de la mémoire partagée avec les plans validés depuis la vue dba_plans.

```
SELECT apg_plan_mgmt.reload();
```

Pour en savoir plus sur toutes les fonctions disponibles pour la gestion des plans de requêtes, consultez Référence de la fonction pour la gestion du plan de requête Aurora PostgreSQL.

Désactivation de la gestion des plans de requêtes Aurora PostgreSQL

Vous pouvez désactiver la gestion des plans de requêtes à tout moment en désactivant `apg_plan_mgmt.use_plan_baselines` et `apg_plan_mgmt.capture_plan_baselines`.

```
labdb=> SET apg_plan_mgmt.use_plan_baselines = off;

labdb=> SET apg_plan_mgmt.capture_plan_baselines = off;
```

Bonnes pratiques pour la gestion de plans de requêtes Aurora PostgreSQL

La gestion du plan de requête vous permet de contrôler comment et quand les plans d'exécution des requêtes changent. En tant qu'administrateur de base de données (DBA), les principaux objectifs de l'utilisation du QPM sont d'éviter les régressions en cas de modification de la base de données et de contrôler si l'optimiseur doit utiliser un nouveau plan. Dans ce qui suit, vous trouverez quelques bonnes pratiques recommandées pour l'utilisation de la gestion du plan de requête. Les approches proactives et réactives de la gestion des plans diffèrent quant à la manière et au moment où les nouveaux plans sont approuvés pour être utilisés.

Table des matières

Gestion proactive des plans pour empêcher toute régression des performances

Pour éviter toute régression des performances des plans, vous faites évoluer les bases de référence des plans en exécutant une procédure qui compare les performances des plans nouvellement découverts à celles de la base de référence existante des plans approuvés, puis approuve automatiquement l'ensemble de plans le plus rapide comme nouvelle base de référence. De cette façon, la base de référence des plans s'améliore au fil du temps, à mesure que des plans plus rapides sont découverts.

1. Dans un environnement de développement, identifiez les instructions SQL qui ont le plus d'impact sur les performances ou le débit du système. Capturez ensuite les plans pour ces instructions comme décrit dans les sections Capture manuelle de plans pour des instructions SQL spécifiques et Capture automatique de plans.
2. Exportez les plans capturés depuis l'environnement de développement et importez-les dans l'environnement de production. Pour plus d'informations, consultez Exportation et importation de plans gérés pour Aurora PostgreSQL.
3. En production, exécutez votre application et imposez l'utilisation des plans gérés approuvés. Pour plus d'informations, consultez Utilisation des plans gérés Aurora PostgreSQL. Tandis que l'application s'exécute, ajoutez également de nouveaux plans à mesure que l'optimiseur les découvre. Pour plus d'informations, consultez Capture automatique de plans.

4. Analysez les plans non approuvés et approuvez ceux qui affichent de bonnes performances. Pour plus d'informations, consultez Évaluation des performances des plans.
5. Tandis que votre application continue de s'exécuter, l'optimiseur commence à utiliser les nouveaux plans selon les besoins.

Assurer la stabilité du plan après une mise à niveau majeure de la version

Chaque version majeure de PostgreSQL comprend des améliorations et des modifications de l'optimiseur de requêtes destinées à améliorer les performances. Toutefois, les plans d'exécution de requêtes générés par l'optimiseur dans les versions antérieures peuvent entraîner des régressions de performances dans les nouvelles versions mises à niveau. Vous pouvez utiliser le gestionnaire de plan de requêtes pour résoudre ces problèmes de performances et garantir la stabilité du plan après une mise à niveau majeure de la version.

L'optimiseur utilise toujours un plan approuvé à coût minimal, même s'il existe plusieurs plans approuvés pour la même instruction. Après une mise à niveau, l'optimiseur peut découvrir de nouveaux plans, mais ils seront enregistrés en tant que plans non approuvés. Ces plans ne sont exécutés que s'ils sont approuvés en utilisant la gestion de plans réactive avec le paramètre unapproved_plan_execution_threshold. Vous pouvez optimiser la stabilité du plan en utilisant la gestion de plans proactive avec le paramètre evolve_plan_baselines. Elle compare les performances des nouveaux plans à celles des anciens, et approuve ou rejette les plans qui sont au moins 10 % plus rapides que le meilleur plan suivant.

Après la mise à niveau, vous pouvez utiliser la fonction `evolve_plan_baselines` pour comparer les performances du plan avant et après la mise à niveau en utilisant vos liaisons de paramètres de requête. Les étapes suivantes supposent que vous avez utilisé des plans de gestion approuvés dans votre environnement de production, comme indiqué dans Utilisation des plans gérés Aurora PostgreSQL.

1. Avant de procéder à la mise à niveau, lancez votre application avec le gestionnaire de plans de requêtes en cours d'exécution. Tandis que l'application s'exécute, ajoutez de nouveaux plans à mesure que l'optimiseur les découvre. Pour de plus amples informations, consultez Capture automatique de plans.
2. Évaluez les performances de chaque plan. Pour de plus amples informations, consultez Évaluation des performances des plans.
3. Après la mise à niveau, analysez à nouveau vos plans approuvés à l'aide de la fonction `evolve_plan_baselines`. Comparez les performances avant et après l'utilisation de vos

liaisons de paramètres de requête. Si le nouveau plan est rapide, vous pouvez l'ajouter à vos plans approuvés. S'il est plus rapide qu'un autre plan pour les mêmes liaisons de paramètres, vous pouvez alors marquer le plan le plus lent comme rejeté.

Pour de plus amples informations, consultez Approbation de plans plus performants. Pour obtenir des informations de référence sur cette fonction, consultez apg_plan_mgmt.evolve_plan_baselines.

Pour plus d'informations, consultez Ensuring consistent performance after major version upgrades with Amazon Aurora PostgreSQL-Compatible Edition Query Plan Management.

> **Note**
>
> Lorsque vous effectuez une mise à niveau d'une version majeure à l'aide d'une réplication logique ou de AWS DMS, assurez-vous de répliquer le schéma `apg_plan_mgmt` pour vous assurer que les plans existants sont copiés dans l'instance mise à niveau. Pour plus d'informations sur la réplication logique, consultez Utilisation de la réplication logique pour effectuer une mise à niveau de version majeure pour Aurora PostgreSQL.

Gestion réactive des plans pour détecter et corriger toute régression des performances

En surveillant votre application pendant son exécution, vous pouvez détecter les plans qui entraînent des régressions de performances. Lorsque vous détectez des régressions, vous devez rejeter ou corriger manuellement les plans problématiques en suivant les étapes suivantes :

1. Tandis que votre application s'exécute, imposez l'utilisation de plans gérés et ajoutez automatiquement les nouveaux plans découverts comme non approuvés. Pour de plus amples informations, veuillez consulter Utilisation des plans gérés Aurora PostgreSQL et Capture automatique de plans.
2. Surveillez l'application en cours d'exécution afin d'identifier toute régression des performances.
3. Lorsque vous détectez une régression d'un plan, définissez le statut de ce plan sur `rejected`. La prochaine fois que l'optimiseur exécutera l'instruction SQL, il ignorera automatiquement le plan rejeté et utilisera un autre plan approuvé à la place. Pour plus d'informations, consultez Rejet ou désactivation de plans plus lents.

   Dans certains cas, vous préférerez peut-être corriger un plan inapproprié plutôt que de le rejeter, de le désactiver ou de le supprimer. Utilisez l'extension `pg_hint_plan` pour tenter d'améliorer un plan. Avec `pg_hint_plan`, vous utilisez des commentaires spéciaux pour indiquer à l'optimiseur

de contourner la procédure normale de création d'un plan. Pour de plus amples informations, consultez Correction de plans à l’aide de pg_hint_plan.

Gestion des plans de requêtes Aurora PostgreSQL

Lorsque la gestion des plans de requêtes est activée pour votre cluster de bases de données Aurora PostgreSQL, l’optimiseur génère et stocke des plans d’exécution des requêtes pour toutes les instructions SQL qu’il traite plusieurs fois. L’optimiseur définit toujours le statut du premier plan généré d’une instruction gérée sur `Approved` et le stocke dans la vue `dba_plans`.

L’ensemble de plans approuvés enregistrés pour une instruction gérée est connu comme sa référence de plans. Tandis que votre application s’exécute, il est possible que l’optimiseur génère d’autres plans pour les instructions gérées. L’optimiseur attribue le statut aux plans capturés supplémentaire `Unapproved`.

Par la suite, vous pouvez décider que les plans `Unapproved` affichent de bonnes performances et les remplacer par `Approved`, `Rejected` ou `Preferred`. Pour ce faire, vous devez utiliser la fonction `apg_plan_mgmt.evolve_plan_baselines` ou la fonction `apg_plan_mgmt.set_plan_status`.

Lorsque l’optimiseur génère un plan pour une instruction SQL, la gestion des plans de requêtes enregistre le plan dans la table `apg_plan_mgmt.plans`. Les utilisateurs de base de données auxquels le rôle `apg_plan_mgmt` a été attribué peuvent voir les détails du plan en interrogeant la vue `apg_plan_mgmt.dba_plans`. Par exemple, la requête suivante répertorie les détails des plans actuellement affichés pour un cluster de bases de données Aurora PostgreSQL hors production.

- `sql_hash` – Identifiant de l’instruction SQL qui est la valeur de hachage du texte normalisé de l’instruction SQL.
- `plan_hash` – Identifiant unique pour le plan qui est une combinaison de `sql_hash` et d’un hachage du plan.
- `status` – Statut du plan. L’optimiseur peut exécuter un plan approuvé.
- `enabled` – Indique si le plan est prêt à être utilisé (true) ou non (false).
- `plan_outline` – Représentation du plan utilisé pour recréer le plan d’exécution réel. Les opérateurs de la structure arborescente correspondent aux opérateurs de la sortie EXPLAIN.

La vue `apg_plan_mgmt.dba_plans` comporte de nombreuses autres colonnes qui contiennent tous les détails du plan, tels que la date à laquelle le plan a été utilisé pour la dernière fois. Consultez

Référence pour la vue apg_plan_mgmt.dba_plans pour l'édition compatible avec Aurora PostgreSQL pour plus de détails.

Normalisation et hachage SQL

Dans la vue `apg_plan_mgmt.dba_plans`, vous pouvez identifier une instruction gérée avec une valeur de hachage SQL. Le hachage SQL est calculé sur la base d'une représentation normalisée de l'instruction SQL qui élimine certaines différences, comme les valeurs littérales.

Le processus de normalisation de chaque instruction SQL préserve l'espace et la casse, afin que vous puissiez toujours lire et comprendre l'essentiel de l'instruction SQL. La normalisation supprime ou remplace les éléments suivants.

- Principaux blocs de commentaires
- Le mot-clé EXPLAIN et les options EXPLAIN, et EXPLAIN ANALYZE
- Espaces de fin
- Tous les littéraux

Prenons par exemple l'instruction suivante.

```
/*Leading comment*/ EXPLAIN SELECT /* Query 1 */ * FROM t WHERE x > 7 AND y = 1;
```

La gestion des plans de requêtes normalise cette instruction comme suit :

```
SELECT /* Query 1 */ * FROM t WHERE x > CONST AND y = CONST;
```

La normalisation permet d'utiliser le même hachage SQL pour des instructions SQL similaires qui peuvent différer uniquement au niveau de leurs valeurs littérales ou de paramètres. En d'autres termes, plusieurs plans pour le même hachage SQL peuvent exister, avec un plan différent optimal dans différentes conditions.

> **Note**
>
> Une instruction SQL unique utilisée avec différents schémas possède des plans différents, car elle est liée au schéma spécifique au moment de l'exécution. Le planificateur utilise les statistiques pour la liaison du schéma afin de choisir le plan optimal.

Pour en savoir plus sur la façon dont l'optimiseur choisit un plan, consultez Utilisation des plans gérés Aurora PostgreSQL. Dans cette section, vous pouvez apprendre à utiliser `EXPLAIN` et `EXPLAIN ANALYZE` pour prévisualiser un plan avant qu'il ne soit réellement utilisé. Pour en savoir plus, consultez Analyse du plan choisi par l'optimiseur. Pour une image décrivant le processus de sélection d'un plan, consultez Sélection du plan à exécuter par l'optimiseur..

Capture des plans d'exécution d'Aurora PostgreSQL

La gestion des plans de requêtes Aurora PostgreSQL propose deux modes différents pour capturer les plans d'exécution des requêtes : automatique ou manuel. Vous pouvez choisir le mode en définissant la valeur du `apg_plan_mgmt.capture_plans_baselines` sur `automatic` ou `manual`. Vous pouvez capturer les plans d'exécution pour des instructions SQL spécifiques à l'aide de la capture de plan manuelle. Vous pouvez aussi capturer la totalité des plans (ou les plus lents) qui sont exécutés au moins deux fois pendant l'exécution de votre application à l'aide de la capture de plan automatique.

Lors de la capture des plans, l'optimiseur définit le statut du premier plan capturé d'une instruction gérée sur `approved`. L'optimiseur définit le statut des autres plans capturés pour une instruction gérée sur `unapproved`. Cependant, il est possible parfois d'enregistrer plusieurs plans avec le statut `approved`. Cela peut se produire lorsque plusieurs plans sont créés pour une instruction en parallèle, et avant que le premier plan pour l'instruction ne soit validé.

Pour contrôler le nombre maximal de plans pouvant être capturés et stockés dans la vue `dba_plans`, définissez le paramètre `apg_plan_mgmt.max_plans` dans votre groupe de paramètres au niveau de l'instance de base de données. Une modification du paramètre `apg_plan_mgmt.max_plans` nécessite la réinitialisation de l'instance de base de données pour que la nouvelle valeur prenne effet. Pour plus d'informations, veuillez consulter le paramètre apg_plan_mgmt.max_plans.

Capture manuelle de plans pour des instructions SQL spécifiques

Si vous disposez d'un ensemble connu d'instructions SQL à gérer, placez les instructions dans un fichier script SQL, puis capturez manuellement les plans. L'exemple suivant montre une commande psql pour la capture manuelle de plans de requêtes pour un ensemble d'instructions SQL.

```
psql> SET apg_plan_mgmt.capture_plan_baselines = manual;
psql> \i my-statements.sql
psql> SET apg_plan_mgmt.capture_plan_baselines = off;
```

Après avoir capturé un plan pour chaque instruction SQL, l'optimiseur ajoute une nouvelle ligne à la vue `apg_plan_mgmt.dba_plans`.

Il est recommandé d'utiliser les instructions EXPLAIN ou EXPLAIN EXECUTE dans le fichier de script SQL. Assurez-vous d'inclure un nombre suffisant de variantes dans les valeurs des paramètres afin de capturer tous les plans intéressants.

Si vous connaissez un meilleur plan que le plan à coût minimal de l'optimiseur, vous pouvez contraindre l'optimiseur à l'utiliser. À cette fin, spécifiez un ou plusieurs indicateurs de l'optimiseur. Pour plus d'informations, consultez Correction de plans à l’aide de pg_hint_plan. Pour comparer les performances des plans `unapproved` et `approved`, et les approuver, les rejeter ou les supprimer, veuillez consulter Évaluation des performances des plans.

Capture automatique de plans

Utilisez la capture automatique des plans pour des situations telles que la suivante :

- Vous ne connaissez pas les instructions SQL spécifiques que vous voulez gérer.
- Vous devez gérer des centaines ou des milliers d'instructions SQL.
- Votre application utilise une API client. Par exemple, JDBC utilise des instructions préparées sans nom ou des instructions en mode bloc qui ne peuvent pas être exprimées en psql.

Pour capturer des plans automatiquement

1. Activez la capture automatique des plans en définissant `apg_plan_mgmt.capture_plan_baselines` sur `automatic` dans le groupe de paramètres au niveau de l'instance de base de données. Pour de plus amples informations, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora.
2. Tandis que l'application s'exécute, l'optimiseur capture des plans pour chaque instruction SQL qui s'exécute au moins deux fois.

   Tandis que l'application s'exécute avec les paramètres par défaut de gestion des plans de requêtes, l'optimiseur capture des plans pour chaque instruction SQL qui s'exécute au moins deux fois. La capture de tous les plans à l'aide des valeurs par défaut entraîne très peu de surcharge de temps d'exécution et peut être activée en production.

Pour désactiver la capture automatique des plans

- Définissez le paramètre `apg_plan_mgmt.capture_plan_baselines` sur `off` depuis le groupe de paramètres au niveau de l'instance de base de données.

Pour mesurer les performances des plans non approuvés et les approuver, les rejeter ou les supprimer, veuillez consulter la section Évaluation des performances des plans.

Utilisation des plans gérés Aurora PostgreSQL

Pour que l'optimiseur utilise les plans capturés pour vos instructions gérées, définissez le paramètre `apg_plan_mgmt.use_plan_baselines` sur `true`. L'exemple suivant est un exemple d'instance locale.

```
SET apg_plan_mgmt.use_plan_baselines = true;
```

Pendant que l'application s'exécute, ce paramètre contraint l'optimiseur à utiliser le plan à coût minimal, préféré ou approuvé qui est valide et activé pour chaque instruction gérée.

Analyse du plan choisi par l'optimiseur

Lorsque le paramètre `apg_plan_mgmt.use_plan_baselines` est défini sur `true`, vous pouvez utiliser les instructions SQL EXPLAIN ANALYZE pour contraindre l'optimiseur à afficher le plan qu'il utiliserait s'il exécutait l'instruction. Voici un exemple.

```
EXPLAIN ANALYZE EXECUTE rangeQuery (1,10000);
```

```
                                                        QUERY PLAN
---------------------------------------------------------------------------------
 Aggregate  (cost=393.29..393.30 rows=1 width=8) (actual time=7.251..7.251 rows=1
 loops=1)
   ->  Index Only Scan using t1_pkey on t1 t  (cost=0.29..368.29 rows=10000 width=0)
 (actual time=0.061..4.859 rows=10000 loops=1)
Index Cond: ((id >= 1) AND (id <= 10000))
         Heap Fetches: 10000
 Planning time: 1.408 ms
 Execution time: 7.291 ms
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1984047223, Plan Hash: 512153379
```

La sortie montre le plan approuvé par rapport à la ligne de base qui serait exécutée. Cependant, la sortie indique également qu'elle a trouvé un plan à un coût inférieur. Dans ce cas, vous pouvez capturer ce nouveau plan à coût minimal en activant la capture automatique des plans comme décrit dans la section Capture automatique de plans.

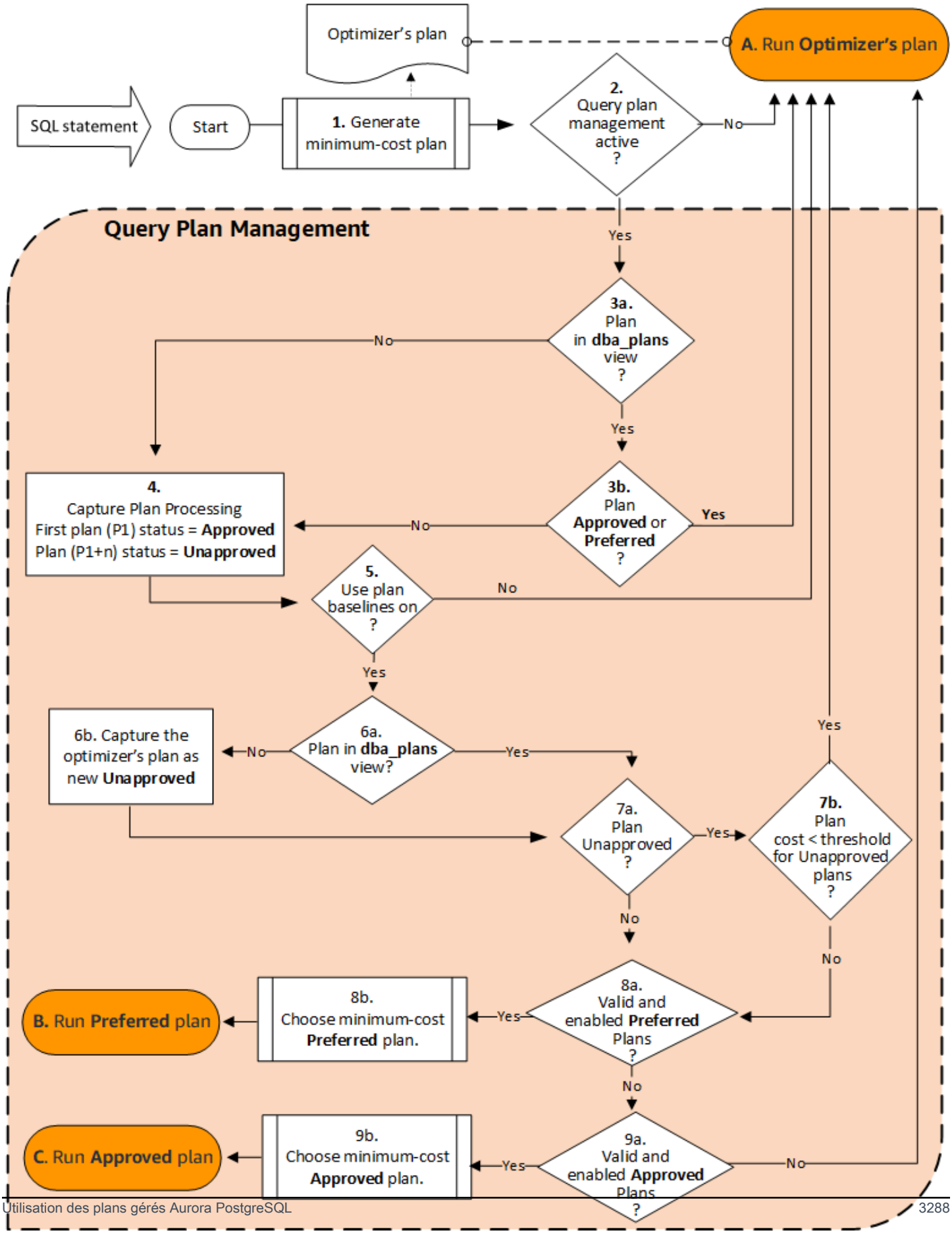
Les nouveaux plans sont toujours capturés par l'optimiseur comme `Unapproved`. Utilisez la fonction `apg_plan_mgmt.evolve_plan_baselines` pour comparer les plans et remplacer leur statut

par approuvé, rejeté ou désactivé. Pour de plus amples informations, consultez Évaluation des performances des plans.

Sélection du plan à exécuter par l'optimiseur.

Le coût d'un plan d'exécution est une estimation effectuée par l'optimiseur pour comparer différents plans. Lorsqu'il calcule le coût d'un plan, l'optimiseur inclut des facteurs tels que les opérations de CPU et d'E/S requises par ce plan. Pour en savoir plus sur l'estimation des coûts du planificateur de requêtes PostgreSQL, consultez la section Query Planning (Planification des requêtes) dans la documentation PostgreSQL.

L'image suivante montre comment un plan est choisi pour une instruction SQL donnée lorsque la gestion du plan de requête est active, et lorsqu'elle ne l'est pas.


Optimizer's plan
A. Run Optimizer's plan
SQL statement
Start
1. Generate minimum-cost plan
2. Query plan management active ?
No
Query Plan Management
Yes
3a. Plan in dba_plans view ?
No
Yes
4. Capture Plan Processing
First plan (P1) status = Approved
Plan (P1+n) status = Unapproved
3b. Plan Approved or Preferred ?
No
Yes
5. Use plan baselines on ?
No
Yes
6a. Plan in dba_plans view?
No
Yes
6b. Capture the optimizer's plan as new Unapproved
7a. Plan Unapproved ?
Yes
7b. Plan cost < threshold for Unapproved plans ?
Yes
No
No
8a. Valid and enabled Preferred Plans ?
Yes
8b. Choose minimum-cost Preferred plan.
B. Run Preferred plan
No
9a. Valid and enabled Approved Plans ?
Yes
9b. Choose minimum-cost Approved plan.
C. Run Approved plan
No

Le déroulement est le suivant :

1. L'optimiseur génère un plan à coût minimal pour l'instruction SQL.
2. Si la gestion du plan de requête n'est pas active, le plan de l'optimiseur est exécuté immédiatement (A. Exécuter le plan de l'optimiseur). La gestion du plan de requête est inactive lorsque les paramètres `apg_plan_mgmt.capture_plan_baselines` et `apg_plan_mgmt.use_plan_baselines` sont tous deux à leur valeur par défaut (« off » et « false », respectivement).

   Dans le cas contraire, la gestion du plan de requête est active. Dans ce cas, l'instruction SQL et le plan de l'optimiseur sont évalués plus en détail avant qu'un plan ne soit choisi.

   > **Tip**
   >
   > Les utilisateurs de la base de données possédant le rôle `apg_plan_mgmt` peuvent comparer les plans de manière proactive, modifier le statut des plans et forcer l'utilisation de plans spécifiques si nécessaire. Pour de plus amples informations, consultez Amélioration des plans de requêtes Aurora PostgreSQL.

3. L'instruction SQL peut déjà contenir des plans qui ont été stockés par la gestion des plans de requêtes dans le passé. Les plans sont stockés dans `apg_plan_mgmt.dba_plans`, avec des informations sur les instructions SQL qui ont été utilisées pour les créer. Les informations sur un plan comprennent son statut. Le statut d'un plan peut déterminer s'il est utilisé ou non, comme suit.
   a. Si le plan ne figure pas parmi les plans stockés pour l'instruction SQL, cela signifie que ce plan vient juste d'être généré par l'optimiseur pour l'instruction SQL donnée. Le plan est envoyé au traitement de la capture du plan (4).
   b. Si le plan figure parmi les plans stockés et que son statut est Approuvé ou Préféré, le plan est exécuté (A. Exécuter le plan de l'optimiseur).

      Si le plan figure parmi les plans stockés mais qu'il n'est ni approuvé ni préféré, il est envoyé au traitement de la capture des plans (4).
4. Lorsqu'un plan est capturé pour la première fois pour une instruction SQL donnée, le statut du plan est toujours défini comme Approuvé (P1). Si l'optimiseur génère par la suite le même plan pour la même instruction SQL, l'état de ce plan est modifié en Non-approuvé (P1+n).

   Une fois le plan capturé et son statut mis à jour, l'évaluation se poursuit à l'étape suivante (5).

5. La référence d'un plan contient l'historique de l'instruction SQL et de ses plans à différents états. La gestion du plan de requête peut prendre en compte la référence lors du choix d'un plan, selon que l'option Use plan baselines (Utiliser les références du plan) est activée ou non, comme suit.
   - L'option Use plan baselines (Utiliser les références du plan) est « désactivée » lorsque le paramètre `apg_plan_mgmt.use_plan_baselines` est défini sur sa valeur par défaut (`false`). Le plan n'est pas comparé à la référence avant son exécution (A. Exécuter le plan de l'optimiseur).
   - L'option Use plan baselines (Utiliser les références du plan) est « activée » lorsque le paramètre `apg_plan_mgmt.use_plan_baselines` est défini sur `true`. Le plan est ensuite évalué à l'aide de la référence (6).
6. Le plan est comparé à d'autres plans pour l'instruction dans la référence.
   a. Si le plan de l'optimiseur figure parmi les plans de la référence, son statut est vérifié (7a).
   b. Si le plan de l'optimiseur ne figure pas parmi les plans de la référence, il est ajouté aux plans de l'instruction en tant que nouveau plan `Unapproved`.
7. Le statut du plan est vérifié pour déterminer uniquement si son statut est Non approuvé.
   a. Si le statut du plan est Non approuvé, le coût estimé du plan est comparé au coût estimé spécifié pour le seuil du plan d'exécution non approuvé.
      - Si le coût estimé du plan est inférieur au seuil, l'optimiseur l'utilise même s'il s'agit d'un plan Non approuvé (A. Exécuter le plan de l'optimiseur). En général, l'optimiseur n'exécute pas de plan non approuvé. Toutefois, lorsque le paramètre `apg_plan_mgmt.unapproved_plan_execution_threshold` spécifie une valeur seuil de coût, l'optimiseur compare le coût du plan Non approuvé à ce seuil. Si le coût estimé est inférieur au seuil, l'optimiseur exécute le plan. Pour de plus amples informations, consultez apg_plan_mgmt.unapproved_plan_execution_threshold.
      - Si le coût estimé du plan n'est pas inférieur au seuil, les autres attributs du plan sont vérifiés (8a).
   b. Si le statut du plan est autre que Non approuvé, ses autres attributs sont vérifiés (8a).
8. L'optimiseur n'utilisera pas un plan qui est désactivé. C'est-à-dire le plan dont l'attribut `enable` est défini comme « f » (faux). L'optimiseur n'utilisera pas non plus un plan dont l'état est Rejeté.

   L'optimiseur ne peut pas utiliser de plans qui ne sont pas valides. Les plans peuvent devenir invalides au fil du temps lorsque les objets dont ils dépendent, tels que les index et les partitions de table, sont retirés ou supprimés.

a. Si l'instruction possède des plans Préférés activés et valides, l'optimiseur choisit le plan à coût minimal parmi les plans Préférés stockés pour cette instruction SQL. L'optimiseur exécute ensuite le plan Préféré à coût minimal.
b. Si l'instruction n'a pas de plans Préféré activés et valides, elle est évaluée à l'étape suivante (9).

9. Si l'instruction possède des plans Approuvés activés et valides, l'optimiseur choisit le plan à coût minimal parmi les plans Approuvés stockés pour cette instruction SQL. L'optimiseur exécute ensuite le plan Approuvé à coût minimal.

   Si l'instruction n'a pas de plan Approuvé valide et activé, l'optimiseur utilise le plan de coût minimum (A. Exécuter le plan de l'optimiseur).

Examen des plans de requête d'Aurora PostgreSQL dans la vue dba_plans

Les utilisateurs et les administrateurs de bases de données auxquels le rôle `apg_plan_mgmt` a été attribué peuvent consulter et gérer les plans stockés dans `apg_plan_mgmt.dba_plans`. L'administrateur d'un cluster de bases de données Aurora PostgreSQL (une personne disposant des autorisations `rds_superuser`) doit explicitement accorder ce rôle aux utilisateurs de bases de données qui doivent travailler avec la gestion du plan de requêtes.

La vue `apg_plan_mgmt` contient l'historique du plan pour toutes les instructions SQL gérées pour chaque base de données sur l'instance d'écriture du cluster de bases de données Aurora PostgreSQL. Cette vue vous permet d'examiner les plans, leur état, la date de leur dernière utilisation et tous les autres détails pertinents.

Comme discuté dans Normalisation et hachage SQL, chaque plan géré est identifié par la combinaison d'une valeur de hachage SQL et d'une valeur de hachage du plan. Ces identifiants vous permettent d'utiliser des outils tels que Amazon RDS Performance Insights pour suivre les performances d'un plan individuel. Pour obtenir plus d'informations sur Performance Insights, consultez Using Amazon RDS performance insights (Utilisation d'Amazon RDS Performance Insights).

Établissement d'une liste des plans gérés

Pour dresser la liste des plans gérés, utilisez l'instruction SELECT dans la vue `apg_plan_mgmt.dba_plans`. L'exemple suivant montre certaines colonnes de la vue `dba_plans` telles que `status`, qui identifie les plans approuvés et non approuvés.

```
SELECT sql_hash, plan_hash, status, enabled, stmt_name
```

```
FROM apg_plan_mgmt.dba_plans;

 sql_hash   | plan_hash |   status   | enabled | stmt_name
------------+-----------+------------+---------+-----------
 1984047223 | 512153379 | Approved   | t       | rangequery
 1984047223 | 512284451 | Unapproved | t       | rangequery
 (2 rows)
```

Pour des raisons de lisibilité, la requête et la sortie affichées ne contiennent que quelques colonnes de la vue `dba_plans`. Pour plus d'informations, consultez Référence pour la vue apg_plan_mgmt.dba_plans pour l'édition compatible avec Aurora PostgreSQL.

Amélioration des plans de requêtes Aurora PostgreSQL

Améliorez la gestion des plans de requêtes en évaluant leurs performances et en y apportant les corrections requises. Pour plus d'informations sur l'amélioration de vos plans de requêtes, consultez les rubriques suivantes.

Rubriques

- Évaluation des performances des plans
- Correction de plans à l'aide de pg_hint_plan

Évaluation des performances des plans

Une fois que l'optimiseur a capturé des plans en tant que non approuvés, utilisez la fonction `apg_plan_mgmt.evolve_plan_baselines` pour comparer les plans sur la base de leurs performances réelles. En fonction des résultats de vos analyses des performances, vous pouvez modifier le statut d'un plan de non approuvé en approuvé ou rejeté. Vous pouvez également décider d'utiliser la fonction `apg_plan_mgmt.evolve_plan_baselines` pour désactiver temporairement un plan s'il ne répond pas à vos exigences.

Approbation de plans plus performants

L'exemple suivant montre comment modifier le statut de plans gérés en approuvé à l'aide de la fonction `apg_plan_mgmt.evolve_plan_baselines`.

```
SELECT apg_plan_mgmt.evolve_plan_baselines (
   sql_hash,
   plan_hash,
```

```
   min_speedup_factor := 1.0,
   action := 'approve'
)
FROM apg_plan_mgmt.dba_plans WHERE status = 'Unapproved';
```

```
NOTICE:     rangequery (1,10000)
NOTICE:     Baseline   [ Planning time 0.761 ms, Execution time 13.261 ms]
NOTICE:     Baseline+1 [ Planning time 0.204 ms, Execution time 8.956 ms]
NOTICE:     Total time benefit: 4.862 ms, Execution time benefit: 4.305 ms
NOTICE:     Unapproved -> Approved
evolve_plan_baselines
-----------------------
0
(1 row)
```

La sortie montre un rapport de performances pour l'instruction `rangequery` avec des liaisons de paramètres de 1 et 10 000. Le nouveau plan non approuvé (`Baseline+1`) est plus performant que le meilleur plan précédent approuvé (`Baseline`). Pour confirmer que le nouveau plan est désormais `Approved`, vérifiez la vue `apg_plan_mgmt.dba_plans`.

```
SELECT sql_hash, plan_hash, status, enabled, stmt_name
FROM apg_plan_mgmt.dba_plans;
```

```
sql_hash  | plan_hash |  status  | enabled | stmt_name
------------+-----------+----------+---------+------------
1984047223 | 512153379 | Approved | t       | rangequery
1984047223 | 512284451 | Approved | t       | rangequery
(2 rows)
```

Le plan géré inclut désormais deux plans approuvés, qui constituent la référence de plans de l'instruction. Vous pouvez également appeler la fonction `apg_plan_mgmt.set_plan_status` afin de définir directement le champ de statut d'un plan sur `'Approved'`, `'Rejected'`, `'Unapproved'` ou `'Preferred'`.

Rejet ou désactivation de plans plus lents

Pour rejeter ou désactiver des plans, transférez `'reject'` ou `'disable'` en tant que paramètre d'action à la fonction `apg_plan_mgmt.evolve_plan_baselines`. Cet exemple désactive tout plan `Unapproved` capturé qui est plus lent d'au moins 10 % que le meilleur plan `Approved` pour l'instruction.

```
SELECT apg_plan_mgmt.evolve_plan_baselines(
sql_hash,  -- The managed statement ID
plan_hash, -- The plan ID
1.1,       -- number of times faster the plan must be
'disable'  -- The action to take. This sets the enabled field to false.
)
FROM apg_plan_mgmt.dba_plans
WHERE status = 'Unapproved' AND   -- plan is Unapproved
origin = 'Automatic';       -- plan was auto-captured
```

Vous pouvez également définir directement un plan sur rejeté ou désactivé. Pour définir directement le champ activé du plan sur `true` ou `false`, appelez la fonction `apg_plan_mgmt.set_plan_enabled`. Pour définir directement le champ de statut d'un plan sur `'Approved'`, `'Rejected'`, `'Unapproved'` ou `'Preferred'`, appelez la fonction `apg_plan_mgmt.set_plan_status`.

Pour supprimer des plans qui ne sont pas valides et dont vous pensez qu'ils le resteront, utilisez la fonction `apg_plan_mgmt.validate_plans`. Cette fonction vous permet de supprimer ou de désactiver des plans non valides. Pour plus d'informations, consultez Validation des plans.

Correction de plans à l'aide de pg_hint_plan

L'optimiseur de requêtes est conçu pour rechercher un plan optimal pour toutes les instructions et, dans la plupart des cas, il trouve un très bon plan. Il peut toutefois arriver que vous sachiez qu'un plan plus performant que celui généré par l'optimiseur existe. Pour amener l'optimiseur à générer le plan souhaité, deux méthodes sont recommandées : utiliser l'extension `pg_hint_plan` ou définir des variables Grand Unified Configuration (GUC) dans PostgreSQL :

- Extension `pg_hint_plan` – Spécifiez un « indicateur » pour modifier le fonctionnement du planificateur à l'aide de l'extension `pg_hint_plan` de PostgreSQL. Pour installer l'extension `pg_hint_plan` et en savoir plus sur son utilisation, consultez la documentation de pg_hint_plan.
- Variables GUC – Remplacez un ou plusieurs paramètres du modèle de coûts ou d'autres paramètres de l'optimiseur, tels que `from_collapse_limit` ou `GEQO_threshold`.

Lorsque vous utilisez une de ces techniques pour forcer l'optimiseur de requêtes à utiliser un plan, vous pouvez également utiliser la gestion des plans de requêtes pour capturer et imposer l'utilisation du nouveau plan.

Vous pouvez utiliser l'extension pg_hint_plan pour modifier l'ordre des jointures, les méthodes de jointure ou les chemins d'accès d'une instruction SQL. Utilisez un commentaire SQL avec une syntaxe pg_hint_plan spéciale pour modifier la manière dont l'optimiseur crée un plan. Par exemple, partons de l'hypothèse que l'instruction SQL possède une jointure bidirectionnelle.

```
SELECT *
FROM t1, t2
WHERE t1.id = t2.id;
```

Supposons ensuite que l'optimiseur choisisse d'utiliser l'ordre des jointures (t1, t2), alors que nous savons que l'ordre (t2, t1) est plus rapide. L'indicateur suivant oblige l'optimiseur à utiliser l'ordre des jointures plus rapide (t2, t1). Incluez EXPLAIN pour que l'optimiseur génère un plan pour l'instruction SQL mais n'exécute pas celle-ci. (Sortie non illustrée.)

```
/*+ Leading ((t2 t1)) */ EXPLAIN SELECT *
FROM t1, t2
WHERE t1.id = t2.id;
```

Les étapes suivantes montrent comment utiliser pg_hint_plan.

Pour modifier le plan généré de l'optimiseur et le capturer à l'aide de pg_hint_plan

1. Activez le mode de capture manuelle.

   ```
   SET apg_plan_mgmt.capture_plan_baselines = manual;
   ```

2. Spécifiez un indicateur pour l'instruction SQL qui vous intéresse.

   ```
   /*+ Leading ((t2 t1)) */ EXPLAIN SELECT *
   FROM t1, t2
   WHERE t1.id = t2.id;
   ```

   Après l'exécution de celle-ci, l'optimiseur capture le plan dans la vue apg_plan_mgmt.dba_plans. Le plan capturé n'inclut pas la syntaxe de commentaire pg_hint_plan spéciale, car la gestion des plans de requêtes normalise l'instruction en supprimant les commentaires de début.

3. Visualisez les plans gérés à l'aide de la vue apg_plan_mgmt.dba_plans.

   ```
   SELECT sql_hash, plan_hash, status, sql_text, plan_outline
   ```

```
FROM apg_plan_mgmt.dba_plans;
```

4. Définissez le statut du plan sur `Preferred`. En procédant ainsi, vous garantissez que l'optimiseur choisit d'exécuter ce plan au lieu d'en sélectionner un parmi l'ensemble de plans approuvés lorsque le plan à coût minimal n'a pas encore le statut `Approved` ou `Preferred`.

```
SELECT apg_plan_mgmt.set_plan_status(sql-hash, plan-hash, 'preferred' );
```

5. Désactivez la capture manuelle des plans et imposez l'utilisation de plans gérés.

```
SET apg_plan_mgmt.capture_plan_baselines = false;
SET apg_plan_mgmt.use_plan_baselines = true;
```

   Désormais, lorsque l'instruction SQL initiale s'exécutera, l'optimiseur choisira un plan `Approved` ou `Preferred`. Si le plan à coût minimal n'est ni `Approved` ni `Preferred`, l'optimiseur choisira le plan `Preferred`.

Suppression des plans de requêtes Aurora PostgreSQL

Supprimez les plans d'exécution que vous n'utilisez pas ou qui ne sont pas valides. Pour en savoir plus sur les la suppression des plans, consultez les sections suivantes.

Rubriques

- Suppression de plans
- Validation des plans

Suppression de plans

Les plans sont automatiquement supprimés s'ils n'ont pas été utilisés depuis plus d'un mois, plus précisément 32 jours. Il s'agit de la valeur par défaut du paramètre `apg_plan_mgmt.plan_retention_period`. Vous pouvez modifier la période de conservation du plan en la prolongeant ou en la raccourcissant à partir de la valeur 1. La détermination du nombre de jours depuis qu'un plan a été utilisé est calculée en soustrayant la date `last_used` de la date actuelle. La date `last_used` correspond à la date la plus récente à laquelle l'optimiseur a choisi un plan en tant que plan à coût minimal ou à laquelle le plan a été exécuté. La date est enregistrée pour le plan dans la vue `apg_plan_mgmt.dba_plans`.

Nous vous recommandons de supprimer des plans qui n'ont pas été utilisés depuis longtemps ou qui ne sont pas utiles. Chaque plan possède une date `last_used` que l'optimiseur met à jour chaque fois qu'il exécute un plan ou le choisit en tant que plan à coût minimal pour une instruction. Vérifiez les dernières dates `last_used` pour identifier les plans que vous pouvez supprimer en toute sécurité.

La requête suivante renvoie une table à trois colonnes indiquant le nombre total de plans, les plans qui n'ont pas pu être supprimés et les plans supprimés avec succès. Elle comprend une requête imbriquée qui est un exemple d'utilisation de la fonction `apg_plan_mgmt.delete_plan` pour supprimer tous les plans qui n'ont pas été choisis en tant que plan à coût minimal au cours des 31 derniers jours et son statut n'est pas `Rejected`.

```
SELECT (SELECT COUNT(*) from apg_plan_mgmt.dba_plans) total_plans,
       COUNT(*) FILTER (WHERE result = -1) failed_to_delete,
       COUNT(*) FILTER (WHERE result = 0) successfully_deleted
       FROM (
            SELECT apg_plan_mgmt.delete_plan(sql_hash, plan_hash) as result
            FROM apg_plan_mgmt.dba_plans
            WHERE last_used < (current_date - interval '31 days')
            AND status <> 'Rejected'
            ) as dba_plans ;
```

```
 total_plans | failed_to_delete | successfully_deleted
-------------+------------------+----------------------
           3 |                0 |                    2
```

Pour plus d'informations, consultez apg_plan_mgmt.delete_plan.

Pour supprimer des plans qui ne sont pas valides et dont vous pensez qu'ils le resteront, utilisez la fonction `apg_plan_mgmt.validate_plans`. Cette fonction vous permet de supprimer ou de désactiver des plans non valides. Pour plus d'informations, consultez Validation des plans.

> **Important**
>
> Si vous ne supprimez pas les plans superflus, vous risquez de tomber à court de mémoire partagée mise de côté pour la gestion des plans de requêtes. Pour contrôler la quantité de mémoire disponible pour les plans gérés, utilisez le paramètre `apg_plan_mgmt.max_plans`. Définissez ce paramètre dans votre groupe de paramètres de votre base de données personnalisés, puis réinitialisez votre instance de base de

données pour appliquer les modifications. Pour plus d'informations, consultez le paramètre apg_plan_mgmt.max_plans.

Validation des plans

Utilisez la fonction `apg_plan_mgmt.validate_plans` pour supprimer ou désactiver les plans non valides.

Des plans peuvent devenir non valides ou obsolètes en cas de suppression d'objets dont ils dépendent, tels qu'un index ou une table. Cependant, un plan peut devenir non valide de manière temporaire seulement si l'objet supprimé est recréé. Si un plan non valide est susceptible de redevenir valide plus tard, vous préférerez peut-être le désactiver ou ne rien faire plutôt que le supprimer.

Pour retrouver et supprimer tous les plans qui sont non valides et qui n'ont pas été utilisés au cours de la semaine écoulée, utilisez la fonction `apg_plan_mgmt.validate_plans` comme suit.

```
SELECT apg_plan_mgmt.validate_plans(sql_hash, plan_hash, 'delete')
FROM apg_plan_mgmt.dba_plans
WHERE last_used < (current_date - interval '7 days');
```

Pour activer ou désactiver directement un plan, utilisez la fonction `apg_plan_mgmt.set_plan_enabled`.

Exportation et importation de plans gérés pour Aurora PostgreSQL

Vous pouvez exporter vos plans gérés et les exporter dans une autre instance de base de données.

Pour exporter des plans gérés

Un utilisateur autorisé peut copier tout sous-ensemble de la table `apg_plan_mgmt.plans` dans une autre table et l'enregistrer à l'aide de la commande pg_dump. Voici un exemple de.

```
CREATE TABLE plans_copy AS SELECT *
FROM apg_plan_mgmt.plans [ WHERE predicates ] ;
```

```
% pg_dump --table apg_plan_mgmt.plans_copy -Ft mysourcedatabase > plans_copy.tar
```

```
DROP TABLE apg_plan_mgmt.plans_copy;
```

Pour importer des plans gérés

1. Copiez le fichier .tar des plans gérés exportés dans le système dans lequel vous voulez restaurer les plans.
2. Utilisez la commande `pg_restore` pour copier le fichier .tar dans une nouvelle table.

   ```
   % pg_restore --dbname mytargetdatabase -Ft plans_copy.tar
   ```

3. Fusionnez la table `plans_copy` avec la table `apg_plan_mgmt.plans`, comme montré dans l'exemple suivant.

   > **Note**
   >
   > Dans certains cas, il se peut que vous procédiez à un vidage depuis une version de l'extension `apg_plan_mgmt` et que vous la restauriez dans une autre version. Dans ces cas-là, il se peut que les colonnes de la table des plans soient différentes. Dans ce cas, nommez les colonnes explicitement au lieu d'utiliser SELECT *.

   ```
   INSERT INTO apg_plan_mgmt.plans SELECT * FROM plans_copy
    ON CONFLICT ON CONSTRAINT plans_pkey
    DO UPDATE SET
    status = EXCLUDED.status,
    enabled = EXCLUDED.enabled,
    -- Save the most recent last_used date
    --
    last_used = CASE WHEN EXCLUDED.last_used > plans.last_used
    THEN EXCLUDED.last_used ELSE plans.last_used END,
    -- Save statistics gathered by evolve_plan_baselines, if it ran:
    --
    estimated_startup_cost = EXCLUDED.estimated_startup_cost,
    estimated_total_cost = EXCLUDED.estimated_total_cost,
    planning_time_ms = EXCLUDED.planning_time_ms,
    execution_time_ms = EXCLUDED.execution_time_ms,
    total_time_benefit_ms = EXCLUDED.total_time_benefit_ms,
    execution_time_benefit_ms = EXCLUDED.execution_time_benefit_ms;
   ```

4. Rechargez les plans gérés dans la mémoire partagée et supprimez la table temporaire des plans.

```
SELECT apg_plan_mgmt.reload(); -- refresh shared memory
DROP TABLE plans_copy;
```

Référence du paramètre de gestion du plan de requête Aurora PostgreSQL

Vous pouvez définir vos préférences pour l'extension `apg_plan_mgmt` en utilisant les paramètres énumérés dans cette section. Celles-ci sont disponibles dans le paramètre personnalisé du cluster de bases de données et dans le groupe de paramètres de base de données associés à votre cluster de bases de données Aurora PostgreSQL. Ces paramètres contrôlent le comportement de la fonction de gestion du plan de requête et la façon dont elle affecte l'optimiseur. Pour plus d'informations sur la configuration de la gestion du plan de requête, consultez Activation de la gestion de plans de requêtes Aurora PostgreSQL. La modification des paramètres suivants n'a aucun effet si l'extension `apg_plan_mgmt` n'est pas configurée comme indiqué dans cette section. Pour plus d'informations sur la modification des paramètres d'instance, consultez Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora et Groupes de paramètres de base de données pour les instances de base de données Amazon Aurora.

Paramètres

- apg_plan_mgmt.capture_plan_baselines
- apg_plan_mgmt.plan_capture_threshold
- apg_plan_mgmt.explain_hashes
- apg_plan_mgmt.log_plan_enforcement_result
- apg_plan_mgmt.max_databases
- apg_plan_mgmt.max_plans
- apg_plan_mgmt.plan_hash_version
- apg_plan_mgmt.plan_retention_period
- apg_plan_mgmt.unapproved_plan_execution_threshold
- apg_plan_mgmt.use_plan_baselines
- auto_explain.hashes

apg_plan_mgmt.capture_plan_baselines

Capture les plans d'exécution des requêtes générés par l'optimiseur pour chaque instruction SQL et les stocke dans la vue `dba_plans`. Par défaut, le nombre maximal de plans pouvant être stockés est de 10 000, tel que spécifié par le paramètre `apg_plan_mgmt.max_plans`. Pour obtenir des informations de référence, consultez apg_plan_mgmt.max_plans.

Vous pouvez définir ce paramètre dans le groupe de paramètres du cluster de bases de données personnalisé ou dans le groupe de paramètres de base de données personnalisé. La modification de la valeur de ce paramètre ne nécessite pas de redémarrage.

Par défaut	Valeurs autorisées	Description
off	automatique	Appliquez ce paramètre au niveau de la session ou dans un groupe de paramètres pour capturer les plans utilisés deux fois ou plus.
	manuelle	Appliquez ce paramètre au niveau de la session ou dans un groupe de paramètres pour capturer les plans utilisés une fois ou plus.
	off	Désactive la capture de plan.

Pour plus d'informations, consultez Capture des plans d'exécution d'Aurora PostgreSQL.

apg_plan_mgmt.plan_capture_threshold

Spécifie un seuil de sorte que si le coût total du plan d'exécution de la requête est inférieur à celui-ci, le plan n'est pas capturé dans la vue `apg_plan_mgmt.dba_plans`.

La modification de la valeur de ce paramètre ne nécessite pas de redémarrage.

Par défaut	Valeurs autorisées	Description
0	0 - 1.79769e+308	Définit le seuil du coût total d'exécution du plan de la requête `apg_plan_mgmt` pour la capture des plans.

Pour plus d'informations, consultez Examen des plans de requête d'Aurora PostgreSQL dans la vue dba_plans.

apg_plan_mgmt.explain_hashes

Spécifie si `EXPLAIN [ANALYZE]` affiche `sql_hash` et `plan_hash` à la fin de sa sortie. La modification de la valeur de ce paramètre ne nécessite pas de redémarrage.

Par défaut	Valeurs autorisées	Description
0	0 (désactivé)	EXPLAIN n'affiche pas sql_hash et plan_hash sans l'option true de hachage.
	1 (activé)	EXPLAIN affiche sql_hash et plan_hash sans l'option true de hachage.

apg_plan_mgmt.log_plan_enforcement_result

Spécifie si les résultats doivent être enregistrés pour voir si les plans gérés par la gestion QPM sont utilisés correctement. Lorsqu'un plan générique stocké est utilisé, aucun enregistrement n'est écrit dans les fichiers journaux. La modification de la valeur de ce paramètre ne nécessite pas de redémarrage.

Par défaut	Valeurs autorisées	Description
none	none	N'affiche aucun résultat d'application de plan dans les fichiers journaux.
	on_error	Affiche uniquement le résultat d'application de plan dans les fichiers journaux quand la gestion QPM n'utilise pas les plans gérés.
	Tout	Affiche tous les résultats d'application de plan dans les fichiers journaux, y compris les succès et les échecs.

apg_plan_mgmt.max_databases

Spécifie le nombre maximum de bases de données sur votre instance en écriture du cluster de bases de données Aurora PostgreSQL qui peuvent utiliser la gestion du plan de requête. Par défaut, jusqu'à dix bases de données peuvent utiliser la gestion du plan de requête. Si vous avez plus de dix bases de données sur l'instance, vous pouvez modifier la valeur de ce paramètre. Pour savoir combien de bases de données se trouvent sur une instance donnée, connectez-vous à l'instance en utilisant `psql`. Ensuite, utilisez la méta-commande psql, `\l`, pour répertorier les bases de données.

Pour modifier la valeur de ce paramètre, vous devez redémarrer l'instance pour que le réglage prenne effet.

Par défaut	Valeurs autorisées	Description
10	10-2147483647	Nombre maximum de bases de données qui peuvent utiliser la gestion du plan de requête sur l'instance.

Vous pouvez définir ce paramètre dans le groupe de paramètres du cluster de bases de données personnalisé ou dans le groupe de paramètres de base de données personnalisé.

apg_plan_mgmt.max_plans

Définit le nombre maximal d'instructions SQL que le gestionnaire de plans de requêtes peut conserver dans la vue `apg_plan_mgmt.dba_plans`. Nous vous recommandons de définir ce paramètre sur `10000` ou plus pour toutes les versions Aurora PostgreSQL.

Vous pouvez définir ce paramètre dans le groupe de paramètres du cluster de bases de données personnalisé ou dans le groupe de paramètres de base de données personnalisé. Pour modifier la valeur de ce paramètre, vous devez redémarrer l'instance pour que le réglage prenne effet.

Par défaut	Valeurs autorisées	Description
10 000	10-2147483647	Nombre maximum de plans qui peuvent être stockés dans la vue `apg_plan_mgmt.dba_plans` .

Par défaut	Valeurs autorisées	Description
		La valeur par défaut pour Aurora PostgreSQL version 10 et les versions plus anciennes est 1 000.

Pour plus d'informations, consultez Examen des plans de requête d'Aurora PostgreSQL dans la vue dba_plans.

apg_plan_mgmt.plan_hash_version

Spécifie les cas d'utilisation que le calcul plan_hash est conçu pour couvrir. Une version supérieure de `apg_plan_mgmt.plan_hash_version` couvre toutes les fonctionnalités de la version inférieure. Par exemple, la version 3 couvre les cas d'utilisation pris en charge par la version 2.

La modification de la valeur de ce paramètre doit être suivie d'un appel à `apg_plan_mgmt.validate_plans('update_plan_hash')`. Elle met à jour les valeurs plan_hash dans chaque base de données avec apg_plan_mgmt installé et les entrées dans la table des plans. Pour plus d'informations, consultez Validation des plans.

Par défaut	Valeurs autorisées	Description
1	1	Calcul plan_hash par défaut.
	2	Le calcul plan_hash a été modifié pour la prise en charge de plusieurs schémas.
	3	Le calcul plan_hash a été modifié pour la prise en charge de plusieurs schémas et la prise en charge des tables partition nées.
	4	Le calcul plan_hash a été modifié pour les opérateurs parallèles et pour prendre charge les nœuds matérialisés.

apg_plan_mgmt.plan_retention_period

Spécifie le nombre de jours pour conserver les plans dans la vue `apg_plan_mgmt.dba_plans`, après quoi ils sont automatiquement supprimés. Par défaut, un plan est supprimé lorsque 32 jours se sont écoulés depuis la dernière utilisation du plan (la colonne `last_used` dans la vue `apg_plan_mgmt.dba_plans`). Vous pouvez remplacer la valeur de ce paramètre par n'importe quel nombre, 1 et plus.

Pour modifier la valeur de ce paramètre, vous devez redémarrer l'instance pour que le réglage prenne effet.

Par défaut	Valeurs autorisées	Description
32	1-2147483647	Nombre maximum de jours depuis la dernière utilisation d'un plan avant qu'il ne soit supprimé.

Pour plus d'informations, consultez Examen des plans de requête d'Aurora PostgreSQL dans la vue dba_plans.

apg_plan_mgmt.unapproved_plan_execution_threshold

Spécifie un seuil en dessous duquel un plan non approuvé peut être utilisé par l'optimiseur. Par défaut, le seuil est de 0, de sorte que l'optimiseur n'exécute pas les plans non approuvés. La définition de ce paramètre sur un seuil de coût extrêmement bas, tel que 100, permet d'éviter les frais d'exécution. Vous pouvez également définir ce paramètre sur une valeur extrêmement élevée, telle que 10000000, en utilisant la gestion de plans réactive. Cela permet à l'optimiseur d'utiliser tous les plans choisis sans frais d'exécution. Lorsqu'un mauvais plan est découvert, vous pouvez le marquer manuellement comme « rejeté » afin qu'il ne soit pas utilisé la prochaine fois.

La valeur de ce paramètre représente une estimation du coût d'exécution d'un plan donné. Si un plan non approuvé est inférieur à ce coût estimé, l'optimiseur l'utilise pour l'instruction SQL. Vous pouvez afficher les plans capturés et leur statut (Approuvé, Non approuvé) dans la vue `dba_plans`. Pour en savoir plus, consultez Examen des plans de requête d'Aurora PostgreSQL dans la vue dba_plans.

La modification de la valeur de ce paramètre ne nécessite pas de redémarrage.

Par défaut	Valeurs autorisées	Description
0	0-2147483647	Coût estimé du plan en dessous duquel un plan non approuvé est utilisé.

Pour plus d'informations, consultez Utilisation des plans gérés Aurora PostgreSQL.

apg_plan_mgmt.use_plan_baselines

Spécifie que l'optimiseur doit utiliser l'un des plans approuvés capturés et stockés dans la vue `apg_plan_mgmt.dba_plans`. Par défaut, ce paramètre est désactivé (false), ce qui amène l'optimiseur à utiliser le plan présentant le coût le plus faible qu'il génère sans autre évaluation. En activant ce paramètre (en lui attribuant la valeur true), l'optimiseur choisit un plan d'exécution de requête pour la déclaration à partir de sa référence de plan. Pour plus d'informations, consultez Utilisation des plans gérés Aurora PostgreSQL. Pour trouver une image détaillant ce processus, consultez Sélection du plan à exécuter par l'optimiseur..

Vous pouvez définir ce paramètre dans le groupe de paramètres du cluster de bases de données personnalisé ou dans le groupe de paramètres de base de données personnalisé. La modification de la valeur de ce paramètre ne nécessite pas de redémarrage.

Par défaut	Valeurs autorisées	Description
false	true	Utilisez un plan approuvé, préféré ou non approuvé à partir de la liste `apg_plan_mgmt.dba_plans` . Si aucun de ces plans ne répond à tous les critères d'évaluation de l'optimiseur, celui-ci peut alors utiliser le plan présentant le coût le plus faible qu'il a lui-même généré. Pour plus d'informations, consultez Sélection du plan à exécuter par l'optimiseur..
	false	Utilisez le plan de coût minimum généré par l'optimiseur.

Vous pouvez évaluer les temps de réponse des différents plans capturés et modifier le statut du plan, si nécessaire. Pour plus d'informations, consultez Amélioration des plans de requêtes Aurora PostgreSQL.

auto_explain.hashes

Spécifie si la sortie auto_explain affiche sql_hash et plan_hash. La modification de la valeur de ce paramètre ne nécessite pas de redémarrage.

Par défaut	Valeurs autorisées	Description
0(désactivé)	0(désactivé)	Le résultat de `auto_explain` n'affiche pas `sql_hash` et `plan_hash` .
	1(activé)	Le résultat de `auto_explain` montre `sql_hash` et `plan_hash` .

Référence de la fonction pour la gestion du plan de requête Aurora PostgreSQL

L'extension `apg_plan_mgmt` fournit les fonctions suivantes.

Fonctions

- apg_plan_mgmt.copy_outline
- apg_plan_mgmt.delete_plan
- apg_plan_mgmt.evolve_plan_baselines
- apg_plan_mgmt.get_explain_plan
- apg_plan_mgmt.plan_last_used
- apg_plan_mgmt.reload
- apg_plan_mgmt.set_plan_enabled
- apg_plan_mgmt.set_plan_status
- apg_plan_mgmt.update_plans_last_used
- apg_plan_mgmt.validate_plans

apg_plan_mgmt.copy_outline

Copiez un hachage et un contour de plan SQL donnés vers un hachage et un contour de plan SQL cible, écrasant ainsi le hachage et le contour de plan de la cible. Cette fonction est disponible dans `apg_plan_mgmt` versions 2.3 et ultérieures.

Syntaxe

```
apg_plan_mgmt.copy_outline(
    source_sql_hash,
    source_plan_hash,
    target_sql_hash,
    target_plan_hash,
    force_update_target_plan_hash
)
```

Valeur renvoyée

Renvoie 0 lorsque la copie est réussie. Déclenche des exceptions pour les entrées non valides.

Paramètres

Paramètre	Description
`source_sql_hash`	L'ID `sql_hash` associé à la valeur `plan_hash` à copier vers la requête cible.
`source_plan_hash`	L'ID `plan_hash` à copier vers la requête cible.
`target_sql_hash`	L'ID `sql_hash` de la requête à mettre à jour avec le hachage et le plan source.
`target_plan_hash`	L'ID `plan_hash` de la requête à mettre à jour avec le hachage et le plan source.
`force_update_target_plan_hash`	(Facultatif) L'ID `target_plan_hash` de la requête est mis à jour même si le plan source n'est pas reproductible pour `target_sql_hash`. Lorsqu'elle est définie sur true, cette

Paramètre	Description
	fonction peut être utilisée pour copier des plans dans des schémas où les noms des relations et les colonnes sont cohérents.

Notes d'utilisation

Cette fonction vous permet de copier un hachage de plan et un contour de plan qui utilise des astuces dans d'autres instructions similaires, ce qui vous évite d'avoir à utiliser des instructions d'astuces en ligne à chaque occurrence dans les instructions cibles. Si la requête cible mise à jour résulte en un plan invalide, cette fonction soulève une erreur et annule la tentative de mise à jour.

apg_plan_mgmt.delete_plan

Supprimez un plan géré.

Syntaxe

```
apg_plan_mgmt.delete_plan(
    sql_hash,
    plan_hash
)
```

Valeur renvoyée

Renvoie 0 si la suppression a réussi ou -1 si elle a échoué.

Paramètres

Paramètre	Description
`sql_hash`	ID `sql_hash` de l'instruction SQL gérée du plan.
`plan_hash`	ID `plan_hash` du plan géré.

apg_plan_mgmt.evolve_plan_baselines

Vérifie si un plan déjà approuvé est plus rapide ou si un plan identifié par l'optimiseur de requêtes en tant que plan à coût minimal est plus rapide.

Syntaxe

```
apg_plan_mgmt.evolve_plan_baselines(
    sql_hash,
    plan_hash,
    min_speedup_factor,
    action
)
```

Valeur renvoyée

Nombre de plans qui n'étaient pas plus rapides que le meilleur plan approuvé.

Paramètres

Paramètre	Description
`sql_hash`	ID `sql_hash` de l'instruction SQL gérée du plan.
`plan_hash`	ID `plan_hash` du plan géré. Utilisez NULL pour indiquer que tous les plans ont la même valeur d'ID `sql_hash`.
`min_speed up_factor`	Le facteur de montée en charge minimum est le nombre de fois qu'un plan doit être plus rapide que le meilleur des plans déjà approuvés pour être approuvé. Sinon, ce facteur peut être le nombre de fois qu'un plan doit être plus lent pour être rejeté ou désactivé. Il s'agit d'une valeur flottante positive.
`action`	Action que la fonction doit exécuter. Les valeurs valides sont notamment les suivantes. La casse n'a pas d'importance. • `'disable'` – Désactivez tout plan correspondant qui ne respecte pas le facteur de montée en charge minimum. • `'approve'` – Activez tout plan correspondant qui respecte le facteur de montée en charge minimum et définissez son statut sur `approved`.

Paramètre	Description
	• `'reject'` – Pour chaque plan correspondant qui ne respecte pas le facteur de montée en charge minimum, définissez son statut sur `rejected`. • NULL – La fonction renvoie simplement le nombre de plans qui ne présentent aucun avantage en termes de performances, car ils ne respectent pas le facteur de montée en charge minimum.

Notes d'utilisation

Définissez les plans spécifiés sur approuvé, rejeté ou désactivé selon que la durée de planification et d'exécution est plus rapide que celle du meilleur plan approuvé selon un facteur que vous pouvez définir. Le paramètre d'action peut être défini sur `'approve'` ou `'reject'` pour approuver ou rejeter automatiquement un plan qui respecte les critères de performance. Vous pouvez également le définir sur '' (chaîne vide) pour tester les performances et produire un rapport, mais sans effectuer d'action.

Vous pouvez inutilement éviter d'exécuter à nouveau la fonction `apg_plan_mgmt.evolve_plan_baselines` pour un plan sur lequel elle a été exécutée récemment. À cette fin, limitez les plans aux plans non approuvés créés récemment. Vous pouvez également éviter d'exécuter la fonction `apg_plan_mgmt.evolve_plan_baselines` sur un plan approuvé qui a reçu un horodatage `last_verified` récent.

Effectuez un test des performances pour comparer la durée de planification et d'exécution de chaque plan par rapport à d'autres plans de la référence. Dans certains cas, il n'existe qu'un seul plan pour une instruction et ce plan est approuvé. Dans ce cas, comparez la durée de planification et la durée d'exécution du plan à ce que seraient ces durées si aucun plan n'était utilisé.

L'avantage (ou le désavantage) incrémentiel de chaque plan est enregistré dans la vue `apg_plan_mgmt.dba_plans` de la colonne `total_time_benefit_ms`. Lorsque cette valeur est positive, il y a un avantage mesurable en termes de performances à inclure ce plan dans la référence.

En plus de consigner la durée de planification et d'exécution de chaque plan candidat, la colonne `last_verified` de la vue `apg_plan_mgmt.dba_plans` est mise à jour avec `current_timestamp`. L'horodatage `last_verified` peut être utilisé pour éviter d'exécuter à nouveau cette fonction sur un plan dont les performances ont récemment été vérifiées.

apg_plan_mgmt.get_explain_plan

Génère le texte d'une instruction EXPLAIN pour l'instruction SQL spécifiée.

Syntaxe

```
apg_plan_mgmt.get_explain_plan(
    sql_hash,
    plan_hash,
    [explainOptionList]
)
```

Valeur renvoyée

Renvoie des statistiques d'exécution pour les instructions SQL spécifiées. Utiliser sans explainOptionList pour renvoyer un plan EXPLAIN simple.

Paramètres

Paramètre	Description
sql_hash	ID sql_hash de l'instruction SQL gérée du plan.
plan_hash	ID plan_hash du plan géré.
explainOptionList	Liste séparée par des virgules d'options d'explication. Les valeurs valides incluent 'analyze' , 'verbose' , 'buffers' , 'hashes' et 'format json'. Si la liste des explainOptionList est NULL ou une chaîne vide (''), cette fonction génère une instruction EXPLAIN, sans aucune statistique.

Notes d'utilisation

Pour le explainOptionList, vous pouvez utiliser l'une des mêmes options que vous utiliseriez avec une instruction EXPLAIN. L'optimiseur Aurora PostgreSQL concatène la liste des options que vous fournissez à l'instruction EXPLAIN.

apg_plan_mgmt.plan_last_used

Renvoie la date `last_used` du plan spécifié depuis la mémoire partagée.

> **Note**
>
> La valeur de la mémoire partagée est toujours à jour sur l'instance de base de données principale du cluster de bases de données. La valeur est uniquement vidée vers périodiquement dans la colonne `last_used` de la vue `apg_plan_mgmt.dba_plans`.

Syntaxe

```
apg_plan_mgmt.plan_last_used(
    sql_hash,
    plan_hash
)
```

Valeur renvoyée

Renvoie la date `last_used`.

Paramètres

Paramètre	Description
`sql_hash`	ID `sql_hash` de l'instruction SQL gérée du plan.
`plan_hash`	ID `plan_hash` du plan géré.

apg_plan_mgmt.reload

Recharge les plans dans la mémoire partagée depuis la vue `apg_plan_mgmt.dba_plans`.

Syntaxe

```
apg_plan_mgmt.reload()
```

Valeur renvoyée

Aucun.

Paramètres

Aucune.

Notes d’utilisation

Appelez `reload` dans les cas suivants :

- Utilisez-le pour rafraîchir immédiatement la mémoire partagée d’un réplica en lecture seule, plutôt que d’attendre que les nouveaux plans se propagent au réplica.
- Utilisez-le après l’importation de plans gérés.

apg_plan_mgmt.set_plan_enabled

Activez ou désactivez un plan géré.

Syntaxe

```
apg_plan_mgmt.set_plan_enabled(
    sql_hash,
    plan_hash,
    [true | false]
)
```

Valeur renvoyée

Renvoie 0 si le paramétrage a réussi ou -1 s’il a échoué.

Paramètres

Paramètre	Description
`sql_hash`	ID `sql_hash` de l’instruction SQL gérée du plan.
`plan_hash`	ID `plan_hash` du plan géré.

Paramètre	Description
`enabled`	Valeur booléenne true ou false : • La valeur `true` active le plan. • La valeur `false` désactive le plan.

apg_plan_mgmt.set_plan_status

Définissez le statut d'un plan géré sur `Approved`, `Unapproved`, `Rejected` ou `Preferred`.

Syntaxe

```
apg_plan_mgmt.set_plan_status(
    sql_hash,
    plan_hash,
    status
)
```

Valeur renvoyée

Renvoie 0 si le paramétrage a réussi ou -1 s'il a échoué.

Paramètres

Paramètre	Description
`sql_hash`	ID `sql_hash` de l'instruction SQL gérée du plan.
`plan_hash`	ID `plan_hash` du plan géré.
`status`	Chaîne dotée de l'une des valeurs suivantes : • `'Approved'` • `'Unapproved'` • `'Rejected'` • `'Preferred'`

Paramètre	Description
	La casse que vous utilisez n'a pas d'importance, mais la valeur du statut est définie avec des capitales initiales dans la vue `apg_plan_mgmt.dba_plans` . Pour plus d'informations sur ces valeurs, consultez la rubrique `status` de la section Référence pour la vue apg_plan_mgmt.dba_plans pour l'édition compatible avec Aurora PostgreSQL.

apg_plan_mgmt.update_plans_last_used

Met immédiatement à jour le tableau des plans avec la date `last_used` stockée dans la mémoire partagée.

Syntaxe

```
apg_plan_mgmt.update_plans_last_used()
```

Valeur renvoyée

Aucun.

Paramètres

Aucune.

Notes d'utilisation

Appelez `update_plans_last_used` pour veiller à ce que les requêtes de la colonne `dba_plans.last_used` utilise les informations les plus récentes. Si la date `last_used` n'est pas immédiatement mise à jour, un processus en arrière-plan met à jour la table des plans avec la date `last_used` une fois par heure (par défaut).

Par exemple, si une instruction avec un certain `sql_hash` commence à s'exécuter lentement, vous pouvez déterminer quels plans de cette instruction ont été exécutés depuis le début de la régression des performances. Pour ce faire, commencez par vider les données de la mémoire partagée sur le disque afin que les dates `last_used` soient à jour, puis interrogez tous les plans du `sql_hash` de l'instruction présentant la régression des performances. Dans la requête, assurez-vous que le date `last_used` est supérieure ou égale à la date à laquelle la régression des performances a commencé. La requête identifie le plan ou l'ensemble de plans susceptibles d'être responsables

de la régression des performances. Vous pouvez utiliser `apg_plan_mgmt.get_explain_plan` avec `explainOptionList` défini sur `verbose, hashes`. Vous pouvez également utiliser `apg_plan_mgmt.evolve_plan_baselines` pour analyser le plan et tous les autres plans susceptibles de s'avérer plus performants.

La fonction `update_plans_last_used` affecte uniquement l'instance de base de données principale du cluster de bases de données.

apg_plan_mgmt.validate_plans

Validez le fait que l'optimiseur peut toujours recréer des plans. L'optimiseur valide les plans `Approved`, `Unapproved` et `Preferred`, que les plans soient activés ou désactivés. Les plans `Rejected` ne sont pas validés. Vous pouvez également utiliser la fonction `apg_plan_mgmt.validate_plans` pour supprimer ou désactiver des plans non valides.

Syntaxe

```
apg_plan_mgmt.validate_plans(
    sql_hash,
    plan_hash,
    action)

apg_plan_mgmt.validate_plans(
    action)
```

Valeur renvoyée

Nombre de plans non valides.

Paramètres

Paramètre	Description
`sql_hash`	ID `sql_hash` de l'instruction SQL gérée du plan.
`plan_hash`	ID `plan_hash` du plan géré. Utilisez NULL pour inclure tous les plans ayant la même valeur d'ID `sql_hash`.
`action`	Action que la fonction doit exécuter pour les plans non valides. Les valeurs de chaînes valides sont notamment les suivantes. La casse n'a pas d'importance.

Paramètre	Description
	• `'disable'` – Chaque plan non valide est désactivé. • `'delete'` – Chaque plan non valide est supprimé. • `'update_plan_hash'` – Met à jour l'ID `plan_hash` des plans qui ne peuvent pas être reproduits exactement. Il vous permet aussi de corriger un plan en réécrivant le code SQL. Vous pouvez ensuite enregistrer le bon plan comme plan `Approved` pour le SQL original. • NULL – La fonction renvoie simplement le nombre de plans non valides. Aucune autre action n'est exécutée. • '' – Une chaîne vide génère un message indiquant le nombre de plans valides et non valides. Toute autre valeur est traitée comme une chaîne vide.

Notes d'utilisation

Utilisez le formulaire `validate_plans(action)` pour valider tous les plans gérés pour toutes les instructions gérées dans la vue `apg_plan_mgmt.dba_plans` complète.

Utilisez le formulaire `validate_plans(sql_hash, plan_hash, action)` pour valider un plan géré spécifié avec `plan_hash` pour une instruction gérée spécifiée avec `sql_hash`.

Utilisez le formulaire `validate_plans(sql_hash, NULL, action)` pour valider tous les plans gérés pour l'instruction gérée spécifiée avec `sql_hash`.

Référence pour la vue apg_plan_mgmt.dba_plans pour l'édition compatible avec Aurora PostgreSQL

Les colonnes des informations des plans de la vue `apg_plan_mgmt.dba_plans` sont notamment les suivantes.

Colonne dba_plans	Description
`cardinality_error`	Mesure de l'erreur entre la cardinalité estimée et la cardinalité réelle. La cardinalité désigne le nombre de lignes de table que le plan doit traiter. Si l'erreur de cardinalité est importante,

Colonne dba_plans	Description
	la probabilité que le plan ne soit pas optimal augmente. Cette colonne est complétée par la fonction apg_plan_mgmt.evolve_plan_baselines.
`compatibility_level`	Ce paramètre indique quand un plan de requête a été validé pour la dernière fois. Dans les versions 12.19, 13.15, 14.12, 15.7, 16.3 et ultérieures d'Aurora PostgreSQL, il affiche le numéro de version d'Aurora. Pour les versions antérieures, il affiche un numéro de version spécifique à la fonctionnalité. **Note** Conservez la valeur par défaut de ce paramètre. Aurora PostgreSQL définit et met à jour automatiquement cette valeur.
`created_by`	Utilisateur authentifié (`session_user`) qui a créé le plan.
`enabled`	Indicateur montrant si le plan est activé ou désactivé. Par défaut, tous les plans sont activés. Vous pouvez désactiver des plans pour empêcher l'optimiseur de les utiliser. Pour modifier cette valeur, utilisez la fonction apg_plan_mgmt.set_plan_enabled.
`environment_variables`	Paramètres et valeurs PostgreSQL Grand Unified Configuration (GUC) que l'optimiseur a remplacés au moment de la capture du plan.
`estimated_startup_cost`	Coût estimé de la configuration de l'optimiseur avant que celui-ci fournisse des lignes d'une table.
`estimated_total_cost`	Coût estimé de l'optimiseur pour la fourniture de la ligne finale du tableau.
`execution_time_benefit_ms`	Avantage de l'activation du plan en termes de temps d'exécution, en millisecondes. Cette colonne est complétée par la fonction apg_plan_mgmt.evolve_plan_baselines.

Colonne dba_plans	Description
`execution_time_ms`	Durée d'exécution du plan estimée en millisecondes. Cette colonne est complétée par la fonction apg_plan_mgmt.evolve_plan_baselines.
`has_side_effects`	Valeur indiquant que l'instruction SQL est une instruction de langage de manipulation de données (DML) ou une instruction SELECT contenant une fonction VOLATILE.
`last_used`	Cette valeur est mise à jour à la date du jour chaque fois que le plan est exécuté ou lorsque le plan est le plan à coût minimal de l'optimiseur de requêtes. Cette valeur est stockée dans la mémoire partagée et vidée périodiquement sur le disque. Pour obtenir la valeur la plus récente, lisez la date dans la mémoire partagée en appelant la fonction `apg_plan_mgmt.plan_last_used(sql_hash, plan_hash)` au lieu de lire la valeur `last_used` . Pour plus d'informations, consultez le paramètre apg_plan_mgmt.plan_retention_period.
`last_validated`	Date et heure les plus récentes auxquelles l'application a vérifié que le plan pouvait être recréé à l'aide de la fonction apg_plan_mgmt.validate_plans ou apg_plan_mgmt.evolve_plan_baselines.
`last_verified`	Date et heure les plus récentes auxquelles la fonction apg_plan_mgmt.evolve_plan_baselines a vérifié qu'un plan était le plus performant pour les paramètres spécifiés.
`origin`	Indique la façon dont le plan a été capturé avec le paramètre apg_plan_mgmt.capture_plan_baselines. Les valeurs valides sont notamment les suivantes : `M` – Le plan a été capturé au moyen de la capture manuelle. `A` – Le plan a été capturé au moyen de la capture automatique.
`param_list`	Valeurs des paramètres qui ont été transférées à l'instruction si celle-ci est une instruction préparée.

Colonne dba_plans	Description
`plan_created`	Date et heure de création du plan.
`plan_hash`	Identifiant du plan. La combinaison de `plan_hash` et `sql_hash` identifie de manière unique un plan spécifique.
`plan_outline`	Représentation du plan utilisé pour recréer le plan d'exécution réel, indépendamment de la base de données. Les opérateurs de l'arborescence correspondent aux opérateurs qui apparaiss ent dans la sortie EXPLAIN.
`planning_time_ms`	Durée réelle d'exécution du planificateur en millisecondes. Cette colonne est complétée par la fonction apg_plan_mgmt.evol ve_plan_baselines.
`queryId`	Hachage d'instruction, tel que calculé par l'extension `pg_stat_statements` . Il ne s'agit pas d'un identifiant stable ou indépendant de la base de données dans la mesure où il dépend d'identifiants d'objet (OID). La valeur sera `0` si `compute_query_id` a pour valeur `off` lors de la capture du plan de requête.
`sql_hash`	Valeur de hachage du texte de l'instruction SQL, normalisée avec les littéraux supprimés.
`sql_text`	Texte complet de l'instruction SQL.

Colonne dba_plans	Description
`status`	Statut du plan, qui détermine la manière dont l'optimiseur utilise un plan. Les valeurs valides sont notamment les suivantes. • `Approved` – Plan utilisable que l'optimiseur peut choisir d'exécuter. L'optimiseur exécute le plan à coût minimal à partir d'un ensemble de plans approuvés d'une instruction gérée (référence). Pour réinitialiser un plan sur le statut approuvé, utilisez la fonction apg_plan_mgmt.evolve_plan_baselines. • `Unapproved` – Plan capturé que vous n'avez pas vérifié en vue de son utilisation. Pour plus d'informations, consultez Évaluation des performances des plans. • `Rejected` – Plan que l'optimiseur n'utilisera pas. Pour plus d'informations, consultez Rejet ou désactivation de plans plus lents. • `Preferred` – Plan que vous avez identifié comme plan préféré à utiliser pour une instruction gérée. Si le plan à coût minimal de l'optimiseur n'est pas un plan approuvé ou un plan préféré, vous pouvez réduire le traitemen t lié à l'application du plan. À cette fin, définissez comme un sous-ensemble des plans approuvé `Preferred` . Si le plan à coût minimal de l'optimiseur n'est pas un plan `Approved`, un plan `Preferred` sera choisi avant un plan `Approved`. Pour réinitialiser un plan sur `Preferred` , utilisez la fonction apg_plan_mgmt.set_plan_status.
`stmt_name`	Nom de l'instruction SQL au sein d'une instruction PREPARE. Cette valeur est une chaîne vide dans le cas d'une instruction préparée sans nom. Cette valeur est NULL dans le cas d'une instruction non préparée.

Colonne dba_plans	Description
`total_time_benefit_ms`	Avantage de l’activation de ce plan en termes de durée totale, en millisecondes. Cette valeur prend en compte la durée de planification et la durée d’exécution. Si cette valeur est négative, cela signifie que l’activation de ce plan est désavantageuse. Cette colonne est complétée par la fonction apg_plan_mgmt.evolve_plan_baselines.

Fonctionnalités avancées de Query Plan Management

Vous trouverez ci-dessous des informations sur les fonctionnalités Query Plan Management (QPM) avancées d’Aurora PostgreSQL :

Rubriques

- Capture de plans d’exécution Aurora PostgreSQL dans des réplicas
- Prise en charge de la partition de table

Capture de plans d’exécution Aurora PostgreSQL dans des réplicas

QPM (Query Plan Management) vous permet de capturer les plans de requête générés par des réplicas Aurora et de les stocker sur l’instance de base de données principale du cluster de bases de données Aurora. Vous pouvez collecter les plans de requête de tous les réplicas Aurora et conserver les plans optimaux dans une table persistante centrale sur l’instance principale. Vous pouvez ensuite appliquer ces plans à d’autres réplicas si nécessaire. Cela vous permet de préserver la stabilité des plans d’exécution et d’améliorer les performances des requêtes sur l’ensemble des clusters de bases de données et des versions du moteur.

Rubriques

- Prérequis
- Gestion de la capture de plans dans des réplicas Aurora
- Résolution des problèmes

Prérequis

Activez **`capture_plan_baselines parameter`** dans le réplica Aurora : définissez le paramètre `capture_plan_baselines` sur automatique ou manuel pour capturer des plans dans des réplicas Aurora. Pour plus d'informations, consultez apg_plan_mgmt.capture_plan_baselines.

Installez l'extension postgres_fdw : vous devez installer l'extension de l'encapsuleur de données externes `postgres_fdw` pour capturer des plans dans des réplicas Aurora. Pour installer l'extension, exécutez la commande suivante dans chaque base de données.

```
postgres=> CREATE EXTENSION IF NOT EXISTS postgres_fdw;
```

Gestion de la capture de plans dans des réplicas Aurora

Activation de la capture de plans dans des réplicas Aurora

Vous devez disposer de privilèges `rds_superuser` pour créer ou supprimer la capture de plans dans des réplicas Aurora. Pour plus d'informations sur les rôles utilisateur et les autorisations, consultez Comprendre les rôles et les autorisations PostgreSQL.

Pour capturer des plans, appelez la fonction apg_plan_mgmt.create_replica_plan_capture dans l'instance de base de données d'enregistreur, comme indiqué ci-dessous :

```
postgres=> CALL apg_plan_mgmt.create_replica_plan_capture('endpoint', 'password');
```

- point de terminaison : le point de terminaison de l'enregistreur ou cluster_endpoint prend en charge le basculement pour la capture de plans dans des réplicas Aurora.

  Pour plus d'informations sur le point de terminaison d'enregistreur Aurora Global Database, consultez Affichage des points de terminaison d'une base de données Amazon Aurora Global Database.

  Pour plus d'informations sur les points de terminaison de cluster, consultez Points de terminaison de cluster pour Amazon Aurora.
- password : nous vous recommandons de suivre les instructions ci-dessous lors de la création du mot de passe afin de renforcer sa sécurité :
  - au moins 8 caractères ;
  - au moins une lettre majuscule, une lettre minuscule et un chiffre ;
  - au moins un caractère spécial (?, !, #, <, >, *, etc.).

> **Note**
>
> Si vous modifiez le point de terminaison, le mot de passe ou le numéro de port, vous devez exécuter à nouveau `apg_plan_mgmt.create_replica_plan_capture()` avec le point de terminaison et le mot de passe pour réinitialiser la capture de plans. Dans le cas contraire, la capture de plans dans des réplicas Aurora échouera.

Désactivation de la capture de plans dans des réplicas Aurora

Vous pouvez désactiver le paramètre `capture_plan_baselines` dans le réplica Aurora en définissant sa valeur sur `off` dans le groupe Paramètre.

Suppression de la capture de plans dans des réplicas Aurora

Vous pouvez supprimer complètement la capture de plans dans des réplicas Aurora. Pour supprimer la capture de plans, appelez `apg_plan_mgmt.remove_replica_plan_capture` comme indiqué :

```
postgres=> CALL apg_plan_mgmt.remove_replica_plan_capture();
```

Vous devez appeler à nouveau apg_plan_mgmt.create_replica_plan_capture() pour activer la capture de plans dans des réplicas Aurora avec le point de terminaison et le mot de passe.

Résolution des problèmes

Vous trouverez ci-dessous des suggestions de résolution des problèmes et des solutions de contournement si le plan n'est pas capturé dans des réplicas Aurora comme prévu.

- Définition des paramètres : vérifiez si le paramètre `capture_plan_baselines` est défini sur la bonne valeur pour activer la capture de plans.
- Installation de l'extension **`postgres_fdw`** : utilisez la requête suivante pour vérifier si `postgres_fdw` est installée.

  ```
  postgres=> SELECT * FROM pg_extension WHERE extname = 'postgres_fdw'
  ```

- Appel de create_replica_plan_capture () : utilisez la commande suivante pour vérifier si le mappage utilisateur existe. Sinon, appelez `create_replica_plan_capture()` pour initialiser la fonctionnalité.

```
postgres=> SELECT * FROM pg_foreign_server WHERE srvname =
 'apg_plan_mgmt_writer_foreign_server';
```

- Point de terminaison et numéro de port : vérifiez si le point de terminaison et le numéro de port sont exacts. Aucun message d'erreur ne s'affiche si ces valeurs sont incorrectes.

  Utilisez la commande suivante pour vérifier si le point de terminaison est utilisé dans create() et dans quelle base de données il réside :

```
postgres=> SELECT srvoptions FROM pg_foreign_server WHERE srvname =
 'apg_plan_mgmt_writer_foreign_server';
```

- reload() : vous devez appeler apg_plan_mgmt.reload() après avoir appelé apg_plan_mgmt.delete_plan() dans les réplicas Aurora pour que la suppression soit effective. Cela permet de s'assurer que le changement a été mis en œuvre avec succès.
- Mot de passe : vous devez saisir le mot de passe dans create_replica_plan_capture() conformément aux instructions mentionnées. Sinon, vous recevrez un message d'erreur. Pour plus d'informations, consultez Gestion de la capture de plans dans des réplicas Aurora. Utilisez un autre mot de passe conforme aux instructions.
- Connexion interrégionales : la capture de plans dans des réplicas Aurora est également prise en charge dans la base de données globale Aurora, où l'instance d'enregistreur et les réplicas Aurora peuvent se trouver dans des régions différentes. Assurez-vous d'utiliser le point de terminaison d'enregistreur Aurora Global Database pour maintenir la connectivité après des événements de basculement ou de bascule. Pour plus d'informations sur les points de terminaison Aurora Global Database, consultez Affichage des points de terminaison d'une base de données Amazon Aurora Global Database. L'instance d'enregistreur et le réplica interrégional doivent être en mesure de communiquer à l'aide d'un appairage de VPC. Pour en savoir plus, consultez Appairage de VPC. En cas de basculement vers une autre région, vous devez reconfigurer le point de terminaison du cluster de bases de données principal.

> **Note**
>
> Lorsque vous utilisez un point de terminaison de cluster au lieu d'un point de terminaison d'enregistreur Aurora Global Database, vous devez mettre à jour le point de terminaison du cluster après avoir effectué une opération de basculement ou de bascule globale.

Prise en charge de la partition de table

La gestion des plans de requêtes Aurora PostgreSQL, aussi appelée QPM, prend en charge le partitionnement déclaratif des tables dans les versions suivantes :

- 15.3 et versions 15 ultérieures
- 14.8 et versions 14 ultérieures
- 13.11 et versions 13 ultérieures

Pour plus d'informations, consultez Partitionnement de table.

Rubriques

- Configuration d'une partition de table
- Capture de plans pour une partition de table
- Application d'un plan de partition de tables
- Convention de nommage

Configuration d'une partition de table

Pour configurer une partition de table dans la gestion de plans de requêtes Aurora PostgreSQL, procédez comme suit :

1. Définissez `apg_plan_mgmt.plan_hash_version` sur 3 ou plus dans le groupe de paramètres du cluster de bases de données.
2. Accédez à une base de données qui utilise la gestion de plans de requêtes et qui a des entrées dans la vue `apg_plan_mgmt.dba_plans`.
3. Appelez `apg_plan_mgmt.validate_plans('update_plan_hash')` pour mettre à jour la valeur `plan_hash` dans la table des plans.
4. Répétez les étapes 2 et 3 pour toutes les bases de données dont la gestion de plans de requêtes est activée et qui a des entrées dans la vue `apg_plan_mgmt.dba_plans`.

Pour plus d'informations sur ces paramètres, consultez Référence du paramètre de gestion du plan de requête Aurora PostgreSQL.

Capture de plans pour une partition de table

Dans la gestion de plans de requêtes, des plans différents se distinguent par leur valeur de `plan_hash`. Pour comprendre comment le paramètre `plan_hash` change, vous devez d'abord comprendre des types de plans similaires.

La combinaison des méthodes d'accès, des noms d'index et des noms de partition sans chiffres, cumulée au niveau du nœud Append doit être constante pour que les plans soient considérés comme identiques. Les partitions spécifiques accessibles dans les plans ne sont pas significatives. Dans l'exemple suivant, une table `tbl_a` est créée avec 4 partitions.

```
postgres=>create table tbl_a(i int, j int, k int, l int, m int) partition by range(i);
CREATE TABLE
postgres=>create table tbl_a1 partition of tbl_a for values from (0) to (1000);
CREATE TABLE
postgres=>create table tbl_a2 partition of tbl_a for values from (1001) to (2000);
CREATE TABLE
postgres=>create table tbl_a3 partition of tbl_a for values from (2001) to (3000);
CREATE TABLE
postgres=>create table tbl_a4 partition of tbl_a for values from (3001) to (4000);
CREATE TABLE
postgres=>create index t_i on tbl_a using btree (i);
CREATE INDEX
postgres=>create index t_j on tbl_a using btree (j);
CREATE INDEX
postgres=>create index t_k on tbl_a using btree (k);
CREATE INDEX
```

Les plans suivants sont considérés comme identiques, car une seule méthode d'analyse est utilisée pour scanner `tbl_a` quel que soit le nombre de partitions interrogées par la requête.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 999 and j < 9910 and k > 50;

                        QUERY PLAN
-----------------------------------------------------------------------
Seq Scan on tbl_a1 tbl_a
    Filter: ((i >= 990) AND (i <= 999) AND (j < 9910) AND (k > 50))
SQL Hash: 1553185667, Plan Hash: -694232056
(3 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                        QUERY PLAN
------------------------------------------------------------------
Append
    ->  Seq Scan on tbl_a1 tbl_a_1
            Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
    ->  Seq Scan on tbl_a2 tbl_a_2
            Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
    SQL Hash: 1553185667, Plan Hash: -694232056
    (6 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a3 tbl_a_3
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -694232056
(8 rows)
```

Les 3 plans suivants sont également considérés comme identiques car, au niveau parent, les méthodes d'accès, les noms d'index et les noms de partition sans chiffres sont `SeqScan tbl_a`, `IndexScan (i_idx) tbl_a`.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a2_i_idx on tbl_a2 tbl_a_2
         Index Cond: ((i >= 990) AND (i <= 1100))
```

```
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(7 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
--------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(10 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
--------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a3 tbl_a_3
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a4_i_idx on tbl_a4 tbl_a_4
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(11 rows)
```

Quels que soient l'ordre et le nombre d'occurrences dans les partitions enfants, les méthodes d'accès, les noms d'index sans chiffres et les noms de partition sans chiffres sont constants au niveau parent pour chacun des plans ci-dessus.

Toutefois, les plans sont considérés comme différents si l'une des conditions suivantes est remplie :

- Toutes les méthodes d'accès supplémentaires sont utilisées dans le plan.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Bitmap Heap Scan on tbl_a3 tbl_a_3
         Recheck Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a3_i_idx
               Index Cond: ((i >= 990) AND (i <= 2100))
 SQL Hash: 1553185667, Plan Hash: 1134525070
(11 rows)
```

- Une ou plusieurs méthodes d'accès dans le plan ne sont plus utilisées.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 1100 and j < 9910 and k > 50;

                               QUERY PLAN
-------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -694232056
(6 rows)
```

- L'index associé à une méthode d'indexation est modifié.

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between
 990 and 1100 and j < 9910 and k > 50;

                               QUERY PLAN
```

```
------------------------------------------------------------------------
 Append
   ->  Seq Scan on tbl_a1 tbl_a_1
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a2_j_idx on tbl_a2 tbl_a_2
         Index Cond: (j < 9910)
         Filter: ((i >= 990) AND (i <= 1100) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993343726
(7 rows)
```

Application d'un plan de partition de tables

Les plans approuvés pour les tables partitionnées sont appliqués avec une correspondance de position. Les plans ne sont pas spécifiques aux partitions et peuvent être appliqués à des partitions autres que les plans référencés dans la requête d'origine. Les plans peuvent également être appliqués pour des requêtes accédant à un nombre différent de partitions de celui du plan approuvé d'origine.

Par exemple, si le plan approuvé se rapporte au plan suivant :

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
 SQL Hash: 1553185667, Plan Hash: -993736942
(10 rows)
```

Ensuite, ce plan peut également être appliqué aux requêtes SQL faisant référence à 2 ou 4 partitions, ou encore plus. Les plans possibles qui pourraient découler de ces scénarios pour l'accès à 2 et 4 partitions sont les suivants :

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 1100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1100) AND (j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(8 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a4 tbl_a_4
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(12 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 3100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
```

```
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 3100) AND (j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a3_i_idx on tbl_a3 tbl_a_3
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Index Scan using tbl_a4_i_idx on tbl_a4 tbl_a_4
         Index Cond: ((i >= 990) AND (i <= 3100))
         Filter: ((j < 9910) AND (k > 50))
 Note: An Approved plan was used instead of the minimum cost plan.
 SQL Hash: 1553185667, Plan Hash: -993736942, Minimum Cost Plan Hash: -1873216041
(14 rows)
```

Envisagez un autre plan approuvé avec des méthodes d'accès différentes pour chaque partition :

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 2100 and j < 9910 and k > 50;

                                QUERY PLAN
---------------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 2100) AND (j < 9910) AND (k > 50))
   ->  Bitmap Heap Scan on tbl_a3 tbl_a_3
         Recheck Cond: ((i >= 990) AND (i <= 2100))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a3_i_idx
               Index Cond: ((i >= 990) AND (i <= 2100))
 SQL Hash: 1553185667, Plan Hash: 2032136998
(12 rows)
```

Dans ce cas, tout plan qui lit à partir de deux partitions ne serait pas appliqué. À moins que toutes les combinaisons (méthode d'accès, nom d'index) du plan approuvé soient utilisables, le plan ne peut pas être appliqué. Par exemple, les plans suivants ont des hachages différents et le plan approuvé ne peut pas être appliqué dans les cas suivants :

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1900 and j < 9910 and k > 50;
```

```
                              QUERY PLAN
-----------------------------------------------------------------------
 Append
   ->  Bitmap Heap Scan on tbl_a1 tbl_a_1
         Recheck Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a1_i_idx
               Index Cond: ((i >= 990) AND (i <= 1900))
   ->  Bitmap Heap Scan on tbl_a2 tbl_a_2
         Recheck Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
         ->  Bitmap Index Scan on tbl_a2_i_idx
               Index Cond: ((i >= 990) AND (i <= 1900))
  Note: This is not an Approved plan.  No usable Approved plan was found.
  SQL Hash: 1553185667, Plan Hash: -568647260
(13 rows)
```

```
postgres=>explain (hashes true, costs false) select j, k from tbl_a where i between 990
 and 1900 and j < 9910 and k > 50;

                              QUERY PLAN
-----------------------------------------------------------------------
 Append
   ->  Index Scan using tbl_a1_i_idx on tbl_a1 tbl_a_1
         Index Cond: ((i >= 990) AND (i <= 1900))
         Filter: ((j < 9910) AND (k > 50))
   ->  Seq Scan on tbl_a2 tbl_a_2
         Filter: ((i >= 990) AND (i <= 1900) AND (j < 9910) AND (k > 50))
 Note: This is not an Approved plan.  No usable Approved plan was found.
 SQL Hash: 1553185667, Plan Hash: -496793743
(8 rows)
```

Convention de nommage

Pour que QPM applique un plan avec des tables partitionnées déclaratives, vous devez suivre des règles d’attribution de nom spécifiques pour les tables parents, les partitions de tables et les index :

- Noms des tables parents : ces noms doivent différer par des alphabets ou des caractères spéciaux, et pas uniquement par des chiffres. Par exemple, tA, tB et tC sont des noms acceptables pour des tables parentes distinctes, alors que t1, t2 et t3 ne le sont pas.

- Noms des tables de partition individuelles : les partitions d'un même parent ne doivent différer les unes des autres que par des chiffres. Par exemple, les noms de partition acceptables pour tA peuvent être tA1, tA2 ou t1A, t2A ou même plusieurs chiffres.

  Toute autre différence (lettres, caractères spéciaux) ne garantit pas l'application du plan.
- Nom des index : dans la hiérarchie des tables de partition, assurez-vous que tous les index portent des noms uniques. Les sections non numériques des noms doivent donc être différentes. Par exemple, si vous avez une table partitionnée nommée `tA` avec un index nommé `tA_col1_idx1`, aucun autre index ne peut être appelé `tA_col1_idx2`. Toutefois, vous pouvez avoir un index nommé `tA_a_col1_idx2`, car la partie non numérique du nom est unique. Cette règle s'applique aux index créés à la fois sur la table parent et sur les tables de partition individuelles.

Le non-respect des conventions de nommage ci-dessus peut entraîner l'échec de l'application des plans approuvés. L'exemple suivant illustre un tel échec d'application :

```
postgres=>create table t1(i int, j int, k int, l int, m int) partition by range(i);
CREATE TABLE
postgres=>create table t1a partition of t1 for values from (0) to (1000);
CREATE TABLE
postgres=>create table t1b partition of t1 for values from (1001) to (2000);
CREATE TABLE
postgres=>SET apg_plan_mgmt.capture_plan_baselines TO 'manual';
SET
postgres=>explain (hashes true, costs false) select count(*) from t1 where i > 0;

                           QUERY PLAN
-----------------------------------------------------------------------------
 Aggregate
   ->  Append
         ->  Seq Scan on t1a t1_1
               Filter: (i > 0)
         ->  Seq Scan on t1b t1_2
               Filter: (i > 0)
 SQL Hash: -1720232281, Plan Hash: -1010664377
(7 rows)
```

```
postgres=>SET apg_plan_mgmt.use_plan_baselines TO 'on';
SET
postgres=>explain (hashes true, costs false) select count(*) from t1 where i > 1000;
```

```
                            QUERY PLAN
------------------------------------------------------------------------
 Aggregate
   ->  Seq Scan on t1b t1
         Filter: (i > 1000)
 Note: This is not an Approved plan. No usable Approved plan was found.
 SQL Hash: -1720232281, Plan Hash: 335531806
(5 rows)
```

Même si les deux plans peuvent sembler identiques, leurs valeurs `Plan Hash` sont différentes en raison du nom des tables enfants. Le nom des tables varie par leurs caractères alphabétiques au lieu de simples chiffres, ce qui entraîne un échec de l'application des plans correspondants.

Utilisation d'extensions avec encapsuleurs de données externes

Pour prolonger la fonctionnalités à votre cluster de bases de données Édition compatible avec Aurora PostgreSQL, vous pouvez installer et utiliser diverses extensions PostgreSQL. Par exemple, si votre cas d'utilisation nécessite une saisie intensive de données dans de très grandes tables, vous pouvez installer l'extension `pg_partman` pour partitionner vos données et ainsi répartir la charge de travail.

> **Note**
>
> Depuis Aurora PostgreSQL 14.5, Aurora PostgreSQL prend en charge le kit Trusted Language Extensions pour PostgreSQL. Cette fonction est mise en œuvre sous forme d'extension `pg_tle`, que vous pouvez ajouter à votre instance Aurora PostgreSQL. En utilisant cette extension, les développeurs peuvent créer leurs propres extensions PostgreSQL dans un environnement sûr qui simplifie les exigences d'installation et de configuration, ainsi qu'une grande partie des tests préliminaires pour les nouvelles extensions. Pour de plus amples informations, consultez Utilisation de Trusted Language Extensions pour PostgreSQL.

Dans certains cas, plutôt que d'installer une extension, vous pouvez ajouter un module spécifique à la liste de `shared_preload_libraries` dans votre groupe de paramètres de cluster de bases de données personnalisé de votre cluster de bases de données Aurora PostgreSQL. Généralement, le groupe de paramètres du cluster de bases de données par défaut charge uniquement le `pg_stat_statements`, mais plusieurs autres modules peuvent être ajoutés à la liste. Par exemple, vous pouvez ajouter une fonctionnalité de planification en ajoutant le module `pg_cron`, comme indiqué dans Planification de la maintenance avec l'extension PostgreSQL pg_cron. Autre exemple, vous pouvez enregistrer les plans d'exécution des requêtes en chargeant le module `auto_explain`. Pour en savoir plus, consultez Consigner les plans d'exécution de requêtes dans le centre de connaissances AWS.

Une extension qui donne accès à des données externes est plus spécifiquement appelée foreign data wrapper (encapsuleur de données externes) (FDW). Par exemple, l'extension `oracle_fdw` permet à votre cluster de bases de données Aurora PostgreSQL de fonctionner avec des bases de données Oracle.

Vous pouvez également spécifier précisément quelles extensions peuvent être installées sur votre instance de base de données Aurora PostgreSQL, en les répertoriant dans le paramètre

`rds.allowed_extensions`. Pour plus d'informations, consultez Restriction de l'installation des extensions PostgreSQL.

Vous trouverez ci-dessous des informations sur la configuration et l'utilisation de certaines des extensions, des modules et des FDW disponibles pour Aurora PostgreSQL. Par souci de simplicité, elles sont toutes appelées « extensions ». Pour obtenir la liste des extensions que vous pouvez utiliser avec les versions d'Aurora PostgreSQL actuellement disponibles, consultez Versions d'extension pour Amazon Aurora PostgreSQL dans les Notes de mise à jour pour Aurora PostgreSQL.

- Gestion des objets volumineux avec le module lo
- Gestion des données spatiales avec l'extension PostGIS
- Gestion des partitions PostgreSQL avec l'extension pg_partman
- Planification de la maintenance avec l'extension PostgreSQL pg_cron
- Utilisation de pgAudit pour journaliser l'activité de la base de données
- Utilisation de pglogical pour synchroniser les données entre les instances
- Utilisation des bases de données Oracle avec l'extension oracle_fdw
- Utilisation de bases de données SQL Server avec l'extension tds_fdw

Utilisation de la prise en charge de la délégation des extensions Amazon Aurora pour PostgreSQL

Grâce à la prise en charge de la délégation des extensions d'Amazon Aurora pour PostgreSQL, vous pouvez déléguer la gestion des extensions à un utilisateur qui n'a pas besoin d'être un `rds_superuser`. Avec cette prise en charge de la délégation des extensions, un rôle appelé `rds_extension` est créé. Vous devrez l'attribuer à un utilisateur pour qu'il puisse gérer les autres extensions. Ce rôle peut créer, mettre à jour et supprimer des extensions.

Vous pouvez spécifier les extensions qui peuvent être installées sur votre instance de base de données Aurora PostgreSQL, en les répertoriant dans le paramètre `rds.allowed_extensions`. Pour en savoir plus, consultez Utilisation des extensions PostgreSQL avec Amazon RDS pour PostgreSQL.

Vous pouvez restreindre la liste des extensions disponibles qui peuvent être gérées par l'utilisateur ayant le rôle `rds_extension` à l'aide du paramètre `rds.allowed_delegated_extensions`.

La prise en charge de la délégation des extensions est disponible dans les versions suivantes :

- Toutes les versions supérieures
- 15.5 et versions 15 ultérieures
- 14.10 et versions 14 ultérieures
- 13.13 et versions 13 ultérieures
- 12.17 et versions 12 ultérieures

Rubriques

Activation de la prise en charge de la délégation des extensions pour un utilisateur

Vous devez effectuer les opérations suivantes pour activer la prise en charge de la délégation des extensions pour un utilisateur :

1. Accorder un rôle **`rds_extension`** à un utilisateur : connectez-vous à la base de données en tant que `rds_superuser` et exécutez la commande suivante.

   ```
   Postgres => grant rds_extension to user_name;
   ```

2. Définir la liste des extensions que les utilisateurs délégués pourront gérer : `rds.allowed_delegated_extensions` permet de spécifier un sous-ensemble des extensions disponibles à l'aide de `rds.allowed_extensions` dans le paramètre de cluster de bases de données. Vous pouvez effectuer cette opération à l'un des niveaux suivants :

- Dans le cluster ou le groupe de paramètres d'instance, via l'API AWS Management Console or. Pour de plus amples informations, veuillez consulter Groupes de paramètres pour Amazon Aurora.
- Utilisez la commande suivante au niveau de la base de données :

```
alter database database_name set rds.allowed_delegated_extensions =
 'extension_name_1,
                   extension_name_2,...extension_name_n';
```

- Utilisez la commande suivante au niveau de l'utilisateur :

```
alter user user_name set rds.allowed_delegated_extensions = 'extension_name_1,
                   extension_name_2,...extension_name_n';
```

> **Note**
>
> Il n'est pas nécessaire de redémarrer la base de données après avoir modifié le paramètre dynamique `rds.allowed_delegated_extensions`.

3. Autoriser l'utilisateur délégué à accéder aux objets créés lors du processus de création des extensions : certaines extensions créent des objets qui nécessitent l'octroi d'autorisations supplémentaires avant que l'utilisateur ayant le rôle `rds_extension` puisse y accéder. `rds_superuser` doit accorder à l'utilisateur délégué l'accès à ces objets. L'une des options consiste à utiliser un déclencheur d'événement pour accorder automatiquement l'autorisation à l'utilisateur délégué.

   Exemple de déclencheur d'événement

   Si vous souhaitez autoriser un utilisateur délégué ayant le rôle `rds_extension` à utiliser des extensions qui nécessitent de définir des autorisations sur ses objets créés lors de la création des extensions, vous pouvez personnaliser l'exemple de déclencheur d'événement ci-dessous et ajouter uniquement les extensions pour lesquelles vous souhaitez que les utilisateurs délégués aient accès à toutes les fonctionnalités. Ce déclencheur d'événement peut être créé sur template1 (modèle par défaut). Par conséquent, toutes les bases de données créées à partir de template1 auront ce déclencheur d'événement. Lorsqu'un utilisateur délégué installe l'extension, ce déclencheur octroie automatiquement la propriété des objets créés par l'extension.

```
CREATE OR REPLACE FUNCTION create_ext()

  RETURNS event_trigger AS $$

DECLARE

  schemaname TEXT;
  databaseowner TEXT;

  r RECORD;

BEGIN

  IF tg_tag = 'CREATE EXTENSION' and current_user != 'rds_superuser' THEN
    RAISE NOTICE 'SECURITY INVOKER';
    RAISE NOTICE 'user: %', current_user;
    FOR r IN SELECT * FROM pg_catalog.pg_event_trigger_ddl_commands()
    LOOP
        CONTINUE WHEN r.command_tag != 'CREATE EXTENSION' OR r.object_type !=
 'extension';

        schemaname = (
            SELECT n.nspname
            FROM pg_catalog.pg_extension AS e
            INNER JOIN pg_catalog.pg_namespace AS n
            ON e.extnamespace = n.oid
            WHERE e.oid = r.objid
        );

        databaseowner = (
            SELECT pg_catalog.pg_get_userbyid(d.datdba)
            FROM pg_catalog.pg_database d
            WHERE d.datname = current_database()
        );
        RAISE NOTICE 'Record for event trigger %, objid: %,tag: %, current_user: %,
 schema: %, database_owenr: %', r.object_identity, r.objid, tg_tag, current_user,
 schemaname, databaseowner;
        IF r.object_identity = 'address_standardizer_data_us' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_gaz TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_lex TO %I WITH GRANT OPTION;', schemaname, databaseowner);
```

```
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.us_rules TO %I WITH GRANT OPTION;', schemaname, databaseowner);
        ELSIF r.object_identity = 'dict_int' THEN
            EXECUTE pg_catalog.format('ALTER TEXT SEARCH DICTIONARY %I.intdict
 OWNER TO %I;', schemaname, databaseowner);
        ELSIF r.object_identity = 'pg_partman' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.part_config TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON
 TABLE %I.part_config_sub TO %I WITH GRANT OPTION;', schemaname, databaseowner);
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE
 ON TABLE %I.custom_time_partitions TO %I WITH GRANT OPTION;', schemaname,
 databaseowner);
        ELSIF r.object_identity = 'postgis_topology' THEN
            EXECUTE pg_catalog.format('GRANT SELECT, UPDATE, INSERT, DELETE ON ALL
 TABLES IN SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT USAGE, SELECT ON ALL SEQUENCES IN
 SCHEMA topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA
 topology TO %I WITH GRANT OPTION;', databaseowner);
            EXECUTE pg_catalog.format('GRANT USAGE ON SCHEMA topology TO %I WITH
 GRANT OPTION;', databaseowner);
        END IF;
    END LOOP;
  END IF;
END;
$$ LANGUAGE plpgsql SECURITY DEFINER;

CREATE EVENT TRIGGER log_create_ext ON ddl_command_end EXECUTE PROCEDURE
 create_ext();
```

Configuration utilisée dans la prise en charge de la délégation des extensions Aurora pour PostgreSQL

Nom de la configuration	Description	Valeur par défaut	Remarques	Qui peut modifier ou accorder l'autorisation
`rds.allowed_delegated_extensions`	Ce paramètre limite les extensions qu'un rôle rds_extension peut gérer dans une base de données. Il doit s'agir d'un sous-ensemble de rds.allowed_extensions.	chaîne vide	• Par défaut, ce paramètre est une chaîne vide, ce qui signifie qu'aucune extension n'a été déléguée aux utilisateurs ayant le rôle `rds_extension` . • Toute extension peut être ajoutée si l'utilisateur dispose des autorisations appropriées. Pour ce faire, définissez le paramètre `rds.allowed_delegated_extensions` sur une chaîne de noms d'extension séparés par des virgules. En ajoutant une liste	rds_superuser

Nom de la configuration	Description	Valeur par défaut	Remarques	Qui peut modifier ou accorder l'autorisation
			d'extensions à ce paramètre, vous identifiez explicitement les extensions que l'utilisateur ayant le rôle `rds_extension` peut installer. • Lorsqu'il est défini sur `*`, cela signifie que toutes les extensions répertoriées dans `rds_allowed_extensions` sont déléguées aux utilisateurs ayant le rôle `rds_extension`. Pour en savoir plus sur la configuration de ce paramètre, consultez Activation de la prise en charge de la délégation des	

Nom de la configuration	Description	Valeur par défaut	Remarques	Qui peut modifier ou accorder l'autorisation
			extensions pour un utilisateur.	
rds.allowed_extensions	Ce paramètre permet au client de limiter les extensions qui peuvent être installées dans l'instance de base de données RDS pour PostgreSQL. Pour plus d'informations, consultez Restriction de l'installation des extensions PostgreSQL.	"*"	Par défaut, ce paramètre est défini sur *, ce qui signifie que toutes les extensions prises en charge par RDS pour PostgreSQL et Aurora PostgreSQL peuvent être créées par les utilisateurs disposant des privilèges nécessaires. Une chaîne vide signifie qu'aucune extension ne peut être installée dans l'instance de base de données Aurora PostgreSQL.	administrateur

Nom de la configuration	Description	Valeur par défaut	Remarques	Qui peut modifier ou accorder l'autorisation
`rds.delegated_extension_allow_drascad`	Ce paramètre contrôle la capacité de l'utilisateur ayant le rôle `rds_extension` à supprimer l'extension à l'aide de l'option en cascade.	off	Par défaut, la propriété `rds.delegated_extension_allow_drop_cascade` a la valeur `off`. Cela signifie que les utilisateurs ayant le rôle `rds_extension` ne sont pas autorisés à supprimer une extension à l'aide de l'option en cascade. Pour qu'ils puissent effectuer cette action, le paramètre `rds.delegated_extension_allow_drop_cascade` doit être défini sur on.	rds_superuser

Désactiver la prise en charge de la délégation des extensions

Désactivation partielle

Les utilisateurs délégués ne peuvent pas créer d'extensions, mais peuvent toujours mettre à jour les extensions existantes.

- Réinitialisez `rds.allowed_delegated_extensions` dans le groupe de paramètres de cluster de bases de données.
- Utilisez la commande suivante au niveau de la base de données :

```
alter database database_name reset rds.allowed_delegated_extensions;
```

- Utilisez la commande suivante au niveau de l'utilisateur :

```
alter user user_name reset rds.allowed_delegated_extensions;
```

Désactivation complète

La révocation du rôle `rds_extension` d'un utilisateur rétablit ses autorisations standard. L'utilisateur ne pourra plus créer, mettre à jour ni supprimer des extensions.

```
postgres => revoke rds_extension from user_name;
```

Avantages liés à l'utilisation de la prise en charge de la délégation des extensions Amazon Aurora

Grâce à la prise en charge de la délégation des extensions d'Amazon Aurora pour PostgreSQL, vous déléguez en toute sécurité la gestion des extensions aux utilisateurs qui n'ont pas le rôle `rds_superuser`. Cette fonctionnalité présente les avantages suivants :

- Vous pouvez facilement déléguer la gestion des extensions aux utilisateurs de votre choix.
- Cela ne nécessite pas le rôle `rds_superuser`.
- Permet de prendre en charge différents ensembles d'extensions pour différentes bases de données dans le même cluster de bases de données.

Limitation de la prise en charge de la délégation des extensions Aurora pour PostgreSQL

- Les objets créés pendant le processus de création d'une extension peuvent nécessiter des privilèges supplémentaires pour que l'extension fonctionne correctement.

Autorisations requises pour certaines extensions

Pour créer, utiliser ou mettre à jour les extensions suivantes, l'utilisateur délégué doit disposer des privilèges nécessaires sur les fonctions, tables et schémas suivants.

Exter s nécen nt des droits de propr ou des autor ions	Fonction	Tables	Schema	Dictionnaire de recherche de texte	Comment
addre tanda er_da s		us_gaz, us_lex, us_lex, l.us_rules			
amch	bt_index_check, bt_index_ parent_check				
dict_i				intdict	
pg_p n		custom_ti me_partitions, part_config, part_config_sub			
pg_s tatem					
Post(	st_tileenvelope	spatial_ref_sys			

Exter s nécer nt des droits de propr ou des autor ions	Fonction	Tables	Schema	Dictionnaire de recherche de texte	Comment
postg aster					
postg opolc		topologie, couche	topologie		l'utilisateur délégué doit être le propriéta ire de la base de données
log_f	create_fo reign_tab le_for_log_file				
rds_t	role_pass word_encr yption_type				
postg iger_ oder		geocode_s ettings_default, geocode_s ettings	tiger		
pg_fr acem	pg_freespace				

Exten s nécen nt des droits de propr ou des autor ions	Fonction	Tables	Schema	Dictionnaire de recherche de texte	Comment
pg_vi lity	pg_visibility				

Considérations de sécurité

N’oubliez pas qu’un utilisateur doté d’un rôle `rds_extension` peut gérer les extensions sur toutes les bases de données sur lesquelles il dispose du privilège de connexion. Si l’intention est de permettre à l’utilisateur délégué de ne gérer une extension que sur une seule base de données, il est recommandé de révoquer tous les privilèges publics sur chaque base de données, puis d’accorder explicitement le privilège de connexion pour cette base de données spécifique à l’utilisateur délégué.

Plusieurs extensions peuvent permettre à un utilisateur d’accéder aux informations de plusieurs bases de données. Assurez-vous que les utilisateurs auxquels vous accordez `rds_extension` disposent de fonctionnalités entre les bases de données avant d’ajouter ces extensions à `rds.allowed_delegated_extensions`. Par exemple, `postgres_fdw` et `dblink` fournissent des fonctionnalités permettant d’interroger les bases de données sur la même instance ou sur des instances distantes. `log_fdw` lit les fichiers journaux du moteur Postgres, qui concernent toutes les bases de données de l’instance et peuvent contenir des requêtes lentes ou des messages d’erreur provenant de plusieurs bases de données. `pg_cron` permet d’exécuter des tâches d’arrière-plan planifiées sur l’instance de base de données et peut configurer des tâches pour qu’elles s’exécutent dans une autre base de données.

Désactivation de la suppression d’extension en cascade

La possibilité de supprimer l’extension avec l’option en cascade par un utilisateur ayant le rôle `rds_extension` est contrôlée par le paramètre `rds.delegated_extension_allow_drop_cascade`. Par défaut, la propriété `rds-delegated_extension_allow_drop_cascade` a la valeur `off`. Cela signifie que les utilisateurs dotés du rôle `rds_extension` ne sont pas autorisés à supprimer une extension à l’aide de l’option en cascade, comme indiqué dans la requête ci-dessous.

```
DROP EXTENSION CASCADE;
```

Car cela supprime automatiquement les objets qui dépendent de l’extension, puis tous les objets qui dépendent de ces objets. Toute tentative d’utilisation de l’option en cascade entraînera une erreur.

Pour qu’ils puissent effectuer cette action, le paramètre `rds.delegated_extension_allow_drop_cascade` doit être défini sur `on`.

La modification du paramètre dynamique `rds.delegated_extension_allow_drop_cascade` ne nécessite pas de redémarrage de la base de données. Vous pouvez procéder à l’un des niveaux suivants :

- Dans le cluster ou le groupe de paramètres d'instance, via l'API AWS Management Console or.
- À l’aide de la commande suivante au niveau de la base de données :

  ```
  alter database database_name set rds.delegated_extension_allow_drop_cascade = 'on';
  ```

- À l’aide de la commande suivante au niveau de l’utilisateur :

  ```
  alter role tenant_user set rds.delegated_extension_allow_drop_cascade = 'on';
  ```

Exemples d’extensions pouvant être ajoutées à l’aide de la prise en charge de la délégation des extensions

- `rds_tools`

  ```
  extension_test_db=> create extension rds_tools;
  CREATE EXTENSION
  extension_test_db=> SELECT * from rds_tools.role_password_encryption_type() where
   rolname = 'pg_read_server_files';
  ```

```
ERROR: permission denied for function role_password_encryption_type
```

- amcheck

```
extension_test_db=> CREATE TABLE amcheck_test (id int);
CREATE TABLE
extension_test_db=> INSERT INTO amcheck_test VALUES (generate_series(1,100000));
INSERT 0 100000
extension_test_db=> CREATE INDEX amcheck_test_btree_idx ON amcheck_test USING btree
 (id);
CREATE INDEX
extension_test_db=> create extension amcheck;
CREATE EXTENSION
extension_test_db=> SELECT bt_index_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_check
extension_test_db=> SELECT bt_index_parent_check('amcheck_test_btree_idx'::regclass);
ERROR: permission denied for function bt_index_parent_check
```

- pg_freespacemap

```
extension_test_db=> create extension pg_freespacemap;
CREATE EXTENSION
extension_test_db=> SELECT * FROM pg_freespace('pg_authid');
ERROR: permission denied for function pg_freespace
extension_test_db=> SELECT * FROM pg_freespace('pg_authid',0);
ERROR: permission denied for function pg_freespace
```

- pg_visibility

```
extension_test_db=> create extension pg_visibility;
CREATE EXTENSION
extension_test_db=> select * from pg_visibility('pg_database'::regclass);
ERROR: permission denied for function pg_visibility
```

- postgres_fdw

```
extension_test_db=> create extension postgres_fdw;
CREATE EXTENSION
extension_test_db=> create server myserver foreign data wrapper postgres_fdw options
 (host 'foo', dbname 'foodb', port '5432');
ERROR: permission denied for foreign-data wrapper postgres_fdw
```

Gestion des objets volumineux avec le module lo

Le module lo (extension) est destiné aux utilisateurs et aux développeurs de base de données qui travaillent avec des bases de données PostgreSQL via des pilotes JDBC ou ODBC. JDBC et ODBC s'attendent à ce que la base de données gère la suppression des objets volumineux lorsque les références à ces derniers changent. Cependant, PostgreSQL ne fonctionne pas de cette façon. PostgreSQL ne suppose pas qu'un objet doit être supprimé lorsque sa référence change. Les objets restent donc sur le disque, sans référence. L'extension lo inclut une fonction qui se déclenche en cas de changement de référence afin de supprimer des objets si nécessaire.

> **Tip**
>
> Pour déterminer si votre base de données peut bénéficier de l'extension lo, utilisez l'utilitaire `vacuumlo` pour vérifier la présence d'objets volumineux orphelins. Pour obtenir le nombre d'objets volumineux orphelins sans effectuer aucune action, exécutez l'utilitaire avec l'option `-n` (no-op). Pour savoir comment procéder, consultez vacuumlo utility.

Le module lo est disponible pour Aurora PostgreSQL 13.7, 12.11, 11.16, 10.21 et versions mineures ultérieures.

Pour installer le module (extension), vous avez besoin de privilèges `rds_superuser`. L'installation de l'extension lo ajoute ce qui suit à votre base de données :

- `lo` : type de données d'objet volumineux (lo) que vous pouvez utiliser pour les objets volumineux binaires (BLOB) et d'autres objets volumineux. Le type de données `lo` est un domaine du type de données `oid`. En d'autres termes, il s'agit d'un identifiant d'objet avec des contraintes facultatives. Pour en savoir plus, consultez Identifiants d'objet dans la documentation PostgreSQL. En termes simples, vous pouvez utiliser le type de données `lo` pour distinguer les colonnes de votre base de données contenant des références d'objets volumineux des autres identifiants d'objet (OID).
- `lo_manage` : fonction que vous pouvez utiliser dans les déclencheurs sur les colonnes de la table contenant des références d'objets volumineux. Lorsque vous supprimez ou modifiez une valeur qui fait référence à un objet volumineux, le déclencheur dissocie l'objet (`lo_unlink`) de sa référence. Utilisez le déclencheur sur une colonne uniquement si la colonne est la seule référence de base de données à l'objet volumineux.

Pour en savoir plus sur le module d'objets volumineux, consultez lo dans la documentation PostgreSQL.

Installation de l'extension lo

Avant d'installer l'extension lo, assurez-vous que vous disposez de privilèges `rds_superuser`.

Pour installer l'extension lo

1. Utilisez `psql` pour vous connecter à l'instance de base de données principale de votre cluster de bases de données Aurora PostgreSQL.

   ```
   psql --host=your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password
   ```

   Lorsque vous y êtes invité, saisissez votre mot de passe. Le client `psql` se connecte à la base de données de connexions administratives par défaut, `postgres=>`, et l'affiche sous forme d'invite.
2. Installez l'extension comme suit.

   ```
   postgres=> CREATE EXTENSION lo;
   CREATE EXTENSION
   ```

Vous pouvez désormais utiliser le type de données `lo` pour définir des colonnes dans vos tables. Vous pouvez, par exemple, créer une table (`images`) qui contient des données d'image tramée. Vous pouvez utiliser le type de données `lo` pour une colonne `raster`, comme illustré dans l'exemple suivant, qui crée une table.

```
postgres=> CREATE TABLE images (image_name text, raster lo);
```

Utilisation de la fonction de déclenchement lo_manage pour supprimer des objets

Vous pouvez utiliser la fonction `lo_manage` dans un déclencheur sur une colonne `lo` ou d'autres colonnes d'objets volumineux pour les nettoyer (et éviter les objets orphelins) lorsque `lo` est mis à jour ou supprimé.

Pour configurer des déclencheurs sur les colonnes qui font référence à des objets volumineux

- Effectuez l'une des actions suivantes :

- Créez un déclencheur BEFORE UPDATE OR DELETE sur chaque colonne de sorte qu'il contienne des références uniques à des objets volumineux, en utilisant le nom de colonne comme argument.

```
postgres=> CREATE TRIGGER t_raster BEFORE UPDATE OR DELETE ON images
    FOR EACH ROW EXECUTE FUNCTION lo_manage(raster);
```

- Appliquez un déclencheur uniquement lorsque la colonne est en cours de mise à jour.

```
postgres=> CREATE TRIGGER t_raster BEFORE UPDATE OF images
    FOR EACH ROW EXECUTE FUNCTION lo_manage(raster);
```

La fonction de déclenchement `lo_manage` fonctionne uniquement dans le contexte de l'insertion ou de la suppression de données de colonne, en fonction de la façon dont vous définissez le déclencheur. Elle n'a aucun effet lorsque vous effectuez une opération `DROP` ou `TRUNCATE` sur une base de données. Cela signifie que vous devriez supprimer les colonnes d'objets de n'importe quelle table avant de procéder à la suppression de la base de données, pour éviter la création d'objets orphelins.

Par exemple, supposons que vous souhaitiez supprimer la base de données contenant la table `images`. Vous supprimez la colonne comme suit.

```
postgres=> DELETE FROM images COLUMN raster
```

En supposant que la fonction `lo_manage` est définie sur cette colonne pour gérer les suppressions, vous pouvez maintenant supprimer la table en toute sécurité.

Suppression d'objets volumineux orphelins à l'aide de **`vacuumlo`**

L'utilitaire `vacuumlo` identifie les objets volumineux orphelins et peut les supprimer des bases de données. Cet utilitaire est disponible depuis PostgreSQL 9.1.24. Si les utilisateurs de base de données travaillent régulièrement avec des objets volumineux, nous vous recommandons d'exécuter `vacuumlo` occasionnellement pour nettoyer les objets volumineux orphelins.

Avant d'installer l'extension lo, vous pouvez utiliser `vacuumlo` pour déterminer si votre cluster de bases de données Aurora PostgreSQL peut en bénéficier. Pour ce faire, exécutez `vacuumlo` avec l'option `-n` (no-op) pour afficher les objets qui seraient supprimés, comme illustré dans l'exemple suivant :

```
$ vacuumlo -v -n -h your-cluster-instance-1.666666666666.aws-region.rds.amazonaws.com -
p 5433 -U postgres docs-lab-spatial-db
Password:*****
Connected to database "docs-lab-spatial-db"
Test run: no large objects will be removed!
Would remove 0 large objects from database "docs-lab-spatial-db".
```

Comme indiqué dans la sortie, les objets volumineux orphelins ne posent aucun problème pour cette base de données.

Pour plus d'informations sur cet utilitaire, consultez `vacuumlo` dans la documentation PostgreSQL.

Comprendre le fonctionnement d'**`vacuumlo`**

La commande `vacuumlo` supprime les objets volumineux orphelins (LO) de votre base de données PostgreSQL sans affecter ni entrer en conflit avec vos tables utilisateur.

La commande fonctionne comme suit :

1. `vacuumlo` commence par créer une table temporaire contenant tous les ID d'objet (OID) des objets volumineux de votre base de données.
2. `vacuumlo` parcourt ensuite chaque colonne de la base de données qui utilise les types de données `oid` ou `lo`. Si `vacuumlo` identifie un OID correspondant dans ces colonnes, il le supprime de la table temporaire. `vacuumlo` ne vérifie que les colonnes portant spécifiquement le nom `oid` ou `lo`, et non les domaines définis à partir de ces types.
3. Les autres entrées de la table temporaire représentent des LO orphelins, que `vacuumlo` peut alors supprimer sans risque.

Amélioration des performances de **`vacuumlo`**

Vous pouvez potentiellement améliorer les performances de `vacuumlo` en augmentant la taille du lot à l'aide de l'option `-l`. `vacuumlo` peut ainsi traiter plus de LO à la fois.

Si la mémoire disponible sur votre système est suffisante pour maintenir la table temporaire en mémoire, augmenter le paramètre `temp_buffers` au niveau de la base de données peut contribuer à de meilleures performances. La table peut ainsi être conservée entièrement en mémoire, ce qui améliore les performances globales.

La requête suivante estime la taille de la table temporaire :

```
SELECT
    pg_size_pretty(SUM(pg_column_size(oid))) estimated_lo_temp_table_size
FROM
    pg_largeobject_metadata;
```

Considérations relatives aux objets volumineux

Voici quelques points importants à garder à l'esprit lors de l'utilisation d'objets volumineux.

- `Vacuumlo` est la seule solution, car il n'existe actuellement aucune autre méthode pour supprimer les LO orphelins.
- Des outils tels que pglogical, la réplication logique native et AWS DMS qui utilisent des technologies de réplication ne prennent pas en charge la réplication d'objets volumineux.
- Lorsque vous concevez le schéma de votre base de données, évitez d'utiliser des objets volumineux dans la mesure du possible et envisagez d'utiliser d'autres types de données, comme `bytea`.
- Exécutez `vacuumlo` régulièrement, au moins une fois par semaine, pour éviter les problèmes liés aux LO orphelins.
- Utilisez un déclencheur avec la fonction `lo_manage` sur les tables qui stockent des objets volumineux pour empêcher la création de LO orphelins.

Gestion des données spatiales avec l'extension PostGIS

PostGIS est une extension de PostgreSQL pour le stockage et la gestion des informations spatiales. Pour en savoir plus sur PostGIS, consultez PostGIS.net.

À partir de la version 10.5, PostgreSQL prend en charge la bibliothèque libprotobuf 1.3.0 utilisée par PostGIS pour travailler avec les données des tuiles vectorielles des boîtes de cartes.

La configuration de l'extension PostGIS nécessite des privilèges `rds_superuser`. Nous vous recommandons de créer un utilisateur (rôle) pour gérer l'extension PostGIS et vos données spatiales. L'extension PostGIS et ses composants associés ajoutent des milliers de fonctions à PostgreSQL. Pensez à créer l'extension PostGIS dans son propre schéma si cela est logique pour votre cas d'utilisation. L'exemple suivant montre comment installer l'extension dans sa propre base de données, mais cela n'est pas nécessaire.

Rubriques

Étape 1 : créer un utilisateur (rôle) pour gérer l'extension PostGIS

Tout d'abord, connectez-vous à votre instance de base de données RDS pour PostgreSQL en tant qu'utilisateur disposant de privilèges `rds_superuser`. Si vous avez conservé le nom par défaut lors de la configuration de votre instance, vous vous connectez en tant que `postgres`.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --username=postgres
 --password
```

Créez un rôle (utilisateur) distinct pour administrer l'extension PostGIS.

```
postgres=>  CREATE ROLE gis_admin LOGIN PASSWORD 'change_me';
CREATE ROLE
```

Accordez des privilèges `rds_superuser` à ce rôle, pour lui permettre d'installer l'extension.

```
postgres=> GRANT rds_superuser TO gis_admin;
GRANT
```

Créez une base de données à utiliser pour vos artefacts PostGIS. Cette étape est facultative. Vous pouvez également créer un schéma dans votre base de données utilisateur pour les extensions PostGIS, mais cela n'est pas non plus nécessaire.

```
postgres=> CREATE DATABASE lab_gis;
CREATE DATABASE
```

Accordez à `gis_admin` tous les privilèges sur la base de données `lab_gis`.

```
postgres=> GRANT ALL PRIVILEGES ON DATABASE lab_gis TO gis_admin;
GRANT
```

Quittez la session et reconnectez-vous à votre instance de base de données RDS pour PostgreSQL en tant que `gis_admin`.

```
postgres=> psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=gis_admin --password --dbname=lab_gis
Password for user gis_admin:...
lab_gis=>
```

Continuez à configurer l'extension comme indiqué dans les étapes suivantes.

Étape 2 : Chargez les extensions PostGIS

L'extension PostGIS comprend plusieurs extensions connexes qui fonctionnent ensemble pour fournir des fonctionnalités géospatiales. En fonction de votre cas d'utilisation, vous n'aurez peut-être pas besoin de toutes les extensions créées dans cette étape.

Utilisez les instructions `CREATE EXTENSION` pour charger les extensions PostGIS.

```
CREATE EXTENSION postgis;
CREATE EXTENSION
CREATE EXTENSION postgis_raster;
CREATE EXTENSION
CREATE EXTENSION fuzzystrmatch;
CREATE EXTENSION
CREATE EXTENSION postgis_tiger_geocoder;
CREATE EXTENSION
CREATE EXTENSION postgis_topology;
CREATE EXTENSION
CREATE EXTENSION address_standardizer_data_us;
CREATE EXTENSION
```

Vous pouvez vérifier les résultats en exécutant la requête SQL présentée dans cet exemple, qui répertorie les extensions et leurs propriétaires.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
```

```
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;
List of schemas
     Name     |   Owner
--------------+-----------
 public       | postgres
 tiger        | rdsadmin
 tiger_data   | rdsadmin
 topology     | rdsadmin
(4 rows)
```

Étape 3 : Transfert de la propriété des schémas d'extension

Utilisez les instructions ALTER SCHEMA pour transférer la propriété des schémas au rôle `gis_admin`.

```
ALTER SCHEMA tiger OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA tiger_data OWNER TO gis_admin;
ALTER SCHEMA
ALTER SCHEMA topology OWNER TO gis_admin;
ALTER SCHEMA
```

Vous pouvez confirmer le changement de propriétaire en exécutant la requête SQL suivante. Vous pouvez également utiliser la méta-commande \dn à partir de la ligne de commande psql.

```
SELECT n.nspname AS "Name",
  pg_catalog.pg_get_userbyid(n.nspowner) AS "Owner"
  FROM pg_catalog.pg_namespace n
  WHERE n.nspname !~ '^pg_' AND n.nspname <> 'information_schema'
  ORDER BY 1;

      List of schemas
     Name     |     Owner
--------------+---------------
 public       | postgres
 tiger        | gis_admin
 tiger_data   | gis_admin
 topology     | gis_admin
(4 rows)
```

Étape 4 : Transfert de la propriété des tables PostGIS

> **Note**
>
> Ne changez pas la propriété des fonctions PostGIS. Le bon fonctionnement et les futures mises à niveau de PostGIS nécessitent que ces fonctions conservent leur propriété d'origine. Pour plus d'informations sur les autorisations PostGIS, consultez Sécurité de PostgreSQL.

Utilisez la fonction suivante pour transférer la propriété des tables PostGIS au rôle `gis_admin`. Exécutez l'instruction suivante à partir de l'invite psql pour créer la fonction.

```
CREATE FUNCTION exec(text) returns text language plpgsql volatile AS $f$ BEGIN EXECUTE
 $1; RETURN $1; END; $f$;
CREATE FUNCTION
```

Ensuite, exécutez cette requête pour exécuter la fonction `exec` qui à son tour exécute les instructions et modifie les autorisations.

```
SELECT exec('ALTER TABLE ' || quote_ident(s.nspname) || '.' || quote_ident(s.relname)
 || ' OWNER TO gis_admin;')
  FROM (
    SELECT nspname, relname
    FROM pg_class c JOIN pg_namespace n ON (c.relnamespace = n.oid)
    WHERE nspname in ('tiger','topology') AND
    relkind IN ('r','S','v') ORDER BY relkind = 'S')
s;
```

Étape 5 : Testez les extensions

Pour éviter d'avoir à spécifier le nom du schéma, ajoutez le schéma `tiger` à votre chemin de recherche en utilisant la commande suivante.

```
SET search_path=public,tiger;
SET
```

Testez le schéma `tiger` à l'aide de l'instruction SELECT suivante.

```
SELECT address, streetname, streettypeabbrev, zip
```

```
 FROM normalize_address('1 Devonshire Place, Boston, MA 02109') AS na;
address | streetname | streettypeabbrev |  zip
---------+------------+------------------+-------
       1 | Devonshire | Pl               | 02109
(1 row)
```

Pour en savoir plus sur cette extension, consultez Tiger Geocoder dans la documentation de PostGIS.

Testez l'accès au schéma `topology` en utilisant l'instruction `SELECT` suivante. Cela appelle la fonction `createtopology` qui enregistre un nouvel objet topologique (my_new_topo) avec l'identifiant de référence spatiale spécifié (26986) et la tolérance par défaut (0.5). Pour en savoir plus, consultez la rubrique CreateTopology (Créer une topologie) dans la documentation de PostGIS.

```
SELECT topology.createtopology('my_new_topo',26986,0.5);
 createtopology
----------------
              1
(1 row)
```

Étape 6 : Mettre à niveau l'extension PostGIS

Chaque nouvelle version de PostgreSQL prend en charge une ou plusieurs versions de l'extension PostGIS compatibles avec cette version. La mise à niveau du moteur PostgreSQL vers une nouvelle version ne met pas automatiquement à niveau l'extension PostGIS. Avant de mettre à niveau le moteur PostgreSQL, vous mettez généralement à niveau PostGIS vers la version la plus récente disponible pour la version actuelle de PostgreSQL. Pour en savoir plus, consultez Versions de l'extension PostGIS.

Après la mise à niveau du moteur PostgreSQL, vous mettez à nouveau à niveau l'extension PostGIS, vers la version prise en charge par la nouvelle version du moteur PostgreSQL. Pour obtenir plus d'informations sur la mise à niveau du moteur PostgreSQL, consultez Test d'une mise à niveau de votre cluster de bases de données de production vers une nouvelle version majeure.

Vous pouvez vérifier à tout moment si des mises à jour de l'extension PostGIS sont disponibles sur votre cluster de bases de données Aurora PostgreSQL . Pour ce faire, exécutez la commande suivante. Cette fonction est disponible avec PostGIS 2.5.0 et les versions ultérieures.

```
SELECT postGIS_extensions_upgrade();
```

Si votre application ne prend pas en charge la dernière version de PostGIS, vous pouvez installer une version plus ancienne de PostGIS qui est disponible dans votre version majeure comme suit.

```
CREATE EXTENSION postgis VERSION "2.5.5";
```

Si vous souhaitez effectuer une mise à niveau vers une version PostGIS spécifique à partir d'une version antérieure, vous pouvez également utiliser la commande suivante.

```
ALTER EXTENSION postgis UPDATE TO "2.5.5";
```

Selon la version à partir de laquelle vous effectuez la mise à niveau, vous devrez peut-être utiliser à nouveau cette fonction. Le résultat de la première exécution de la fonction détermine si une mise à niveau supplémentaire est nécessaire. C'est le cas, par exemple, pour la mise à niveau de PostGIS 2 vers PostGIS 3. Pour plus d'informations, consultez Mise à niveau de PostGIS 2 vers PostGIS 3.

Si vous avez mis à niveau cette extension pour vous préparer à une mise à niveau de version majeure du moteur PostgreSQL, vous pouvez continuer avec d'autres tâches préliminaires. Pour plus d'informations, consultez Test d'une mise à niveau de votre cluster de bases de données de production vers une nouvelle version majeure.

Versions de l'extension PostGIS

Nous vous recommandons d'installer les versions de toutes les extensions, telles que PostGIS, telles qu'elles sont répertoriées dans Versions des extensions pour l'Édition compatible avec PostgreSQL dans Notes de mise à jour d'Aurora PostgreSQL. Pour obtenir une liste des versions qui sont disponibles dans votre version, utilisez la commande suivante.

```
SELECT * FROM pg_available_extension_versions WHERE name='postgis';
```

Vous pouvez trouver des informations sur les versions dans les sections suivantes des Notes de mise à jour d'Aurora PostgreSQL :

- Versions d'extension pour Aurora PostgreSQL 14
- Versions d'extension pour Aurora Édition compatible avec PostgreSQL 13
- Versions d'extension pour Aurora Édition compatible avec PostgreSQL 12
- Versions d'extension pour Aurora Édition compatible avec PostgreSQL 11
- Versions d'extension pour Aurora Édition compatible avec PostgreSQL 10
- Versions d'extension pour Aurora Édition compatible avec PostgreSQL 9.6

Mise à niveau de PostGIS 2 vers PostGIS 3

Depuis la version 3.0, la fonctionnalité matricielle de PostGIS est désormais une extension distincte, `postgis_raster`. Cette extension dispose de son propre chemin d'installation et de mise à niveau. Cela supprime des dizaines de fonctions, de types de données et d'autres artefacts nécessaires au traitement des images matricielles de l'extension `postgis` de base. Cela signifie que si votre cas d'utilisation ne nécessite pas de traitement matriciel, vous n'avez pas besoin d'installer l'extension `postgis_raster`.

Dans l'exemple de mise à niveau suivant, la première commande de mise à niveau extrait la fonctionnalité raster dans l'extension `postgis_raster`. Une deuxième commande de mise à niveau est alors nécessaire pour mettre à niveau `postgis_raster` vers la nouvelle version.

Pour effectuer une mise à niveau de PostGIS 2 vers PostGIS 3

1. Identifiez la version par défaut de PostGIS qui est disponible pour la version de PostgreSQL sur votre cluster de bases de données Aurora PostgreSQL. Pour ce faire, exécutez la requête suivante.

```
SELECT * FROM pg_available_extensions
    WHERE default_version > installed_version;
  name   | default_version | installed_version |                          comment
---------+-----------------+-------------------
+------------------------------------------------------------
 postgis | 3.1.4           | 2.3.7             | PostGIS geometry and geography
 spatial types and functions
(1 row)
```

2. Identifiez les versions de PostGIS installées dans chaque base de données sur l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL. En d'autres termes, interrogez chaque base de données utilisateur comme suit.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
```

```
    AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
  Name   | Version | Schema |                            Description
---------+---------+--------
+------------------------------------------------------------------------
 postgis | 2.3.7   | public | PostGIS geometry, geography, and raster spatial types
 and functions
(1 row)
```

Ce décalage entre la version par défaut (PostGIS 3.1.4) et la version installée (PostGIS 2.3.7) signifie que vous devez mettre à niveau l'extension PostGIS.

```
ALTER EXTENSION postgis UPDATE;
ALTER EXTENSION
WARNING: unpackaging raster
WARNING: PostGIS Raster functionality has been unpackaged
```

3. Exécutez la requête suivante pour vérifier que la fonctionnalité raster est maintenant dans son propre package.

```
SELECT
    probin,
    count(*)
FROM
    pg_proc
WHERE
    probin LIKE '%postgis%'
GROUP BY
    probin;
          probin          | count
--------------------------+-------
 $libdir/rtpostgis-2.3    | 107
 $libdir/postgis-3        | 487
(2 rows)
```

Le résultat montre qu'il y a toujours une différence entre les versions. Les fonctions PostGIS sont en version 3 (postgis-3), tandis que les fonctions raster (rtpostgis) sont en version 2

(rtpostgis-2.3). Pour terminer la mise à niveau, vous exécutez à nouveau la commande de mise à niveau, comme suit.

```
postgres=> SELECT postgis_extensions_upgrade();
```

Vous pouvez ignorer les messages d'avertissement, il n'y a aucun risque. Exécutez à nouveau la requête suivante pour vérifier que la mise à niveau est terminée. La mise à niveau est terminée lorsque PostGIS et toutes les extensions associées ne sont plus marquées comme nécessitant une mise à niveau.

```
SELECT postgis_full_version();
```

4. Utilisez la requête suivante pour voir le processus de mise à niveau terminé et les extensions packagées séparément, et vérifiez que leurs versions correspondent.

```
SELECT
    e.extname AS "Name",
    e.extversion AS "Version",
    n.nspname AS "Schema",
    c.description AS "Description"
FROM
    pg_catalog.pg_extension e
    LEFT JOIN pg_catalog.pg_namespace n ON n.oid = e.extnamespace
    LEFT JOIN pg_catalog.pg_description c ON c.objoid = e.oid
        AND c.classoid = 'pg_catalog.pg_extension'::pg_catalog.regclass
WHERE
    e.extname LIKE '%postgis%'
ORDER BY
    1;
      Name      | Version | Schema |                            Description
----------------+---------+--------
+------------------------------------------------------------------------
 postgis        | 3.1.5   | public | PostGIS geometry, geography, and raster
 spatial types and functions
 postgis_raster | 3.1.5   | public | PostGIS raster types and functions
(2 rows)
```

Le résultat montre que l'extension PostGIS 2 a été mise à niveau vers PostGIS 3, et que `postgis` et l'extension `postgis_raster` maintenant séparée sont en version 3.1.5.

Une fois cette mise à niveau terminée, si vous ne prévoyez pas d'utiliser la fonctionnalité raster, vous pouvez abandonner l'extension comme suit.

```
DROP EXTENSION postgis_raster;
```

Gestion des partitions PostgreSQL avec l'extension pg_partman

Le partitionnement de table PostgreSQL fournit un cadre à des fins de traitement hautes performances des entrées de données et des rapports. Utilisez le partitionnement pour les bases de données nécessitant une saisie très rapide de grandes quantités de données. Le partitionnement permet également d'interroger plus rapidement les tables volumineuses. Le partitionnement permet de maintenir les données sans affecter l'instance de base de données, car il nécessite moins de ressources d'I/O.

Le partitionnement vous permet de diviser les données en morceaux de taille personnalisée à des fins de traitement. Par exemple, vous pouvez choisir de partitionner des données chronologiques pour des plages telles que les plages horaires, quotidiennes, hebdomadaires, mensuelles, trimestrielles, annuelles, personnalisées ou toute autre combinaison de celles-ci. Pour un exemple de données chronologiques, si vous partitionnez la table par heure, chaque partition contiendra une heure de données. Si vous partitionnez la table chronologique par jour, chaque partition contiendra un jour de données, etc. La clé de partition contrôle la taille d'une partition.

Lorsque vous utilisez une commande SQL `INSERT` ou `UPDATE` sur une table partitionnée, le moteur de base de données achemine les données vers la partition qui convient. Les partitions de table PostgreSQL qui stockent les données sont des tables enfants de la table principale.

Lors de la lecture d'une requête de base de données, l'optimiseur PostgreSQL examine la clause `WHERE` de la requête et, si possible, dirige l'analyse de base de données vers les seules partitions pertinentes.

À partir de la version 10, PostgreSQL utilise le partitionnement déclaratif pour implémenter le partitionnement de table. Cette technique est également connue sous le nom de partitionnement PostgreSQL natif. Avant PostgreSQL version 10, il fallait utiliser des déclencheurs pour implémenter des partitions.

Le partitionnement de table PostgreSQL offre les fonctionnalités suivantes :

- Création de nouvelles partitions à tout moment.

- Plages de partitions variables.
- Partitions détachables et ré-attachables à l’aide d’instructions DDL (data definition language).

  Par exemple, les partitions détachables sont utiles pour supprimer les données historiques de la partition principale, tout en les conservant à des fins d’analyse.
- Les nouvelles partitions héritent des propriétés de la table de base de données parent, et notamment :
  - Index
  - Clés primaires devant inclure la colonne de clé de partition
  - Clés étrangères
  - Contraintes de validation
  - Références
- Création d’index pour la table complète ou chaque partition spécifique.

Vous ne pouvez pas modifier le schéma d’une partition individuelle. Vous pouvez cependant modifier la table parent (par exemple, ajouter une nouvelle colonne), qui se propage aux partitions.

Rubriques

Présentation de l’extension PostgreSQL pg_partman

Vous pouvez utiliser l’extension `pg_partman` PostgreSQL pour automatiser la création et la maintenance des partitions de table. Pour plus d’informations générales, consultez PG Partition Manager dans la documentation `pg_partman`.

> **Note**
>
> L’extension `pg_partman` est prise en charge sur Aurora PostgreSQL versions 12.6 et ultérieures.

Plutôt que de créer manuellement chaque partition, vous configurez pg_partman avec les paramètres suivants :

- Table à partitionner
- Type de partition
- Clé de partition
- Granularité de partition
- Options de pré-création et de gestion des partitions

Après avoir créé une table partitionnée PostgreSQL, vous l'enregistrez auprès de pg_partman en appelant la fonction create_parent. Cela crée les partitions nécessaires en fonction des paramètres passés dans la fonction.

L'extension pg_partman propose également la fonction run_maintenance_proc que vous pouvez appeler sur une base planifiée pour gérer automatiquement les partitions. Programmez cette fonction de manière à ce qu'elle s'exécute périodiquement (par exemple, toutes les heures) pour vous assurer que les partitions appropriées sont créées, si besoin. Vous pouvez également vous assurer que les partitions sont automatiquement supprimées.

Activation de l'extension pg_partman

En présence de plusieurs bases de données au sein de la même instance de base de données pour laquelle vous souhaitez gérer les partitions, activez l'extension pg_partman séparément pour chaque base de données. Pour activer l'extension pg_partman pour une base de données spécifique, créez le schéma de maintenance de partition, puis créez l'extension pg_partman comme suit.

```
CREATE SCHEMA partman;
CREATE EXTENSION pg_partman WITH SCHEMA partman;
```

> **Note**
>
> Pour créer l'extension pg_partman, assurez-vous que vous disposez des privilèges rds_superuser.

Si vous recevez une erreur similaire à la suivante, accordez les privilèges `rds_superuser` au compte ou utilisez votre compte de super-utilisateur.

```
ERROR: permission denied to create extension "pg_partman"
HINT: Must be superuser to create this extension.
```

Pour accorder des privilèges `rds_superuser`, connectez-vous avec votre compte de super-utilisateur et exécutez la commande suivante :

```
GRANT rds_superuser TO user-or-role;
```

Pour les exemples illustrant l'utilisation de l'extension pg_partman, nous utilisons l'exemple de table et de partition de base de données ci-dessous. Cette base de données utilise une table partitionnée basée sur un horodatage. Un schéma `data_mart` contient une table nommée `events` avec une colonne nommée `created_at`. Les paramètres suivants sont inclus dans la table `events` :

- Clés primaires `event_id` et `created_at`, qui doivent utiliser la colonne pour guider la partition.
- Contrainte de vérification `ck_valid_operation` pour appliquer des valeurs pour une colonne de table `operation`.
- Deux clés étrangères, l'une (`fk_orga_membership`)) pointant vers la table externe `organization` et l'autre (`fk_parent_event_id`) correspondant à un clé étrangère auto-référencée.
- Deux index, l'un (`idx_org_id`) correspondant à la clé étrangère et l'autre (`idx_event_type`) au type d'événement.

Les instructions DDL suivantes créent ces objets, qui sont automatiquement inclus dans chaque partition.

```
CREATE SCHEMA data_mart;
CREATE TABLE data_mart.organization ( org_id BIGSERIAL,
        org_name TEXT,
        CONSTRAINT pk_organization PRIMARY KEY (org_id)
    );

CREATE TABLE data_mart.events(
        event_id        BIGSERIAL,
        operation       CHAR(1),
        value           FLOAT(24),
```

```
        parent_event_id BIGINT,
        event_type      VARCHAR(25),
        org_id          BIGSERIAL,
        created_at      timestamp,
        CONSTRAINT pk_data_mart_event PRIMARY KEY (event_id, created_at),
        CONSTRAINT ck_valid_operation CHECK (operation = 'C' OR operation = 'D'),
        CONSTRAINT fk_orga_membership
            FOREIGN KEY(org_id)
            REFERENCES data_mart.organization (org_id),
        CONSTRAINT fk_parent_event_id
            FOREIGN KEY(parent_event_id, created_at)
            REFERENCES data_mart.events (event_id,created_at)
    ) PARTITION BY RANGE (created_at);

CREATE INDEX idx_org_id     ON  data_mart.events(org_id);
CREATE INDEX idx_event_type ON  data_mart.events(event_type);
```

Configuration des partitions à l'aide de la fonction create_parent

Après avoir activé l'extension `pg_partman`, utilisez la fonction `create_parent` pour configurer les partitions dans le schéma de maintenance des partitions. L'exemple suivant utilise l'exemple de table `events` créé dans Activation de l'extension pg_partman. Appelez la fonction `create_parent` comme suit.

```
SELECT partman.create_parent(
 p_parent_table => 'data_mart.events',
 p_control      => 'created_at',
 p_type         => 'range',
 p_interval     => '1 day',
 p_premake      => 30);
```

Les paramètres sont les suivants :

- `p_parent_table` – Table parent partitionnée. Cette table doit être présente et pleinement qualifiée, y compris le schéma.
- `p_control` – Colonne sur laquelle le partitionnement doit être basé. Le type de données doit être un entier ou une valeur basée sur le temps.
- `p_type` – Le type est `'range'` ou `'list'`.
- `p_interval` – Intervalle de temps ou plage d'entiers pour chaque partition. Par exemple : `1 day`, `1 hour` et d'autres valeurs similaires.

- p_premake – Nombre de partitions à créer à l'avance pour prendre en charge les nouvelles insertions.

Pour une description complète de la fonction create_parent, consultez Fonctions de création dans la documentation pg_partman.

Configuration de la maintenance des partitions à l'aide de la fonction run_maintenance_proc

Vous pouvez exécuter des opérations de maintenance des partitions pour créer automatiquement de nouvelles partitions, détacher des partitions ou supprimer d'anciennes partitions. La maintenance des partitions repose sur la fonction run_maintenance_proc de l'extension pg_partman, et l'extension pg_cron, qui lance un planificateur interne. Le planificateur pg_cron exécute automatiquement les instructions SQL, fonctions et procédures définies dans vos bases de données.

L'exemple suivant utilise l'exemple de table events créé dans Activation de l'extension pg_partman pour définir l'exécution automatique des opérations de maintenance des partitions. Au préalable, ajoutez pg_cron au paramètre shared_preload_libraries dans le groupe de paramètres de l'instance de base de données.

```
CREATE EXTENSION pg_cron;

UPDATE partman.part_config
SET infinite_time_partitions = true,
    retention = '3 months',
    retention_keep_table=true
WHERE parent_table = 'data_mart.events';
SELECT cron.schedule('@hourly', $$CALL partman.run_maintenance_proc()$$);
```

Vous trouverez ci-dessous une explication étape par étape de l'exemple précédent :

1. Modifiez le groupe de paramètres associé à votre instance de base de données et ajoutez pg_cron à la valeur du paramètre shared_preload_libraries. Pour prendre effet, cette modification implique un redémarrage de l'instance de base de données. Pour plus d'informations, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora.
2. Exécutez la commande CREATE EXTENSION pg_cron; à l'aide d'un compte disposant des autorisations rds_superuser. Cela permet d'activer l'extension pg_cron. Pour plus d'informations, consultez Planification de la maintenance avec l'extension PostgreSQL pg_cron.

3. Exécutez la commande `UPDATE partman.part_config` pour ajuster les paramètres `pg_partman` de la table `data_mart.events`.
4. Exécutez la commande `SET . . .` pour configurer la table `data_mart.events` avec les clauses suivantes :
   a. `infinite_time_partitions = true,` – Configure la table pour créer automatiquement de nouvelles partitions sans aucune limite.
   b. `retention = '3 months',` – Configure la table pour présenter une rétention maximale de trois mois.
   c. `retention_keep_table=true` – configure la table de telle sorte qu'au terme de la période de rétention, la table ne soit pas supprimée automatiquement. Les partitions antérieures à la période de rétention sont uniquement détachées de la table parent.
5. Exécutez la commande `SELECT cron.schedule . . .` pour faire un appel de fonction `pg_cron`. Cet appel définit la fréquence à laquelle le planificateur exécute la procédure de maintenance `pg_partman`, `partman.run_maintenance_proc`. Pour cet exemple, la procédure s'exécute toutes les heures.

Pour une description complète de la fonction `run_maintenance_proc`, consultez Fonctions de maintenance dans la documentation `pg_partman`.

Planification de la maintenance avec l'extension PostgreSQL pg_cron

Vous pouvez utiliser l'extension `pg_cron` PostgreSQL pour planifier des commandes de maintenance dans une base de données PostgreSQL. Pour plus d'informations concernant l'extension, consultez What is pg_cron? (Qu'est-ce que pg_cron ?) dans la documentation pg_cron.

L'extension `pg_cron` est prise en charge par le moteur Aurora PostgreSQL à partir de la version 12.6.

Pour en savoir plus sur l'utilisation de `pg_cron`, consultez Schedule jobs with pg_cron on your RDS pour PostgreSQL or your Aurora PostgreSQL-Compatible Edition databases (Planifier des tâches avec pg_cron sur votre RDS pour PostgreSQL ou sur vos bases de données Aurora Édition compatible avec PostgreSQL).

> **Note**
>
> La version d'extension `pg_cron` est affichée sous forme de version à deux chiffres, par exemple 1.6, dans la vue pg_available_extensions. Bien que des versions à trois chiffres,

comme 1.6.4 ou 1.6.5, puissent apparaître dans certains contextes, il est nécessaire d'indiquer la version à deux chiffres lors de la mise à niveau d'une extension.

Rubriques

Configuration de l'extension pg_cron

Configurez l'extension `pg_cron` comme suit :

1. Modifiez le groupe de paramètres personnalisé employé avec votre instance de base de données PostgreSQL en ajoutant `pg_cron` à la valeur du paramètre `shared_preload_libraries`.

   Redémarrez l'instance de la base de données PostgreSQL pour que les modifications du groupe de paramètres prennent effet. Pour en savoir plus sur l'utilisation des groupes de paramètres, consultez Paramètres Amazon Aurora PostgreSQL..

2. Après le redémarrage de l'instance de base de données PostgreSQL, exécutez la commande suivante à l'aide d'un compte disposant d'autorisations `rds_superuser`. Par exemple, si vous avez utilisé les paramètres par défaut lors de la création de votre cluster de bases de données Aurora PostgreSQL, connectez-vous en tant qu'utilisateur `postgres` et créez l'extension.

   ```
   CREATE EXTENSION pg_cron;
   ```

   Le planificateur `pg_cron` est défini dans la base de données PostgreSQL par défaut nommée `postgres`. Les objets `pg_cron` sont créés dans cette base de données `postgres` et toutes les actions de planification s'y exécutent.

3. Vous pouvez utiliser les paramètres par défaut ou planifier des tâches à exécuter dans d'autres bases de données de votre instance de base de données PostgreSQL. Pour planifier des tâches dans d'autres bases de données de votre instance de base de données PostgreSQL, consultez l'exemple disponible dans Planification d'une tâche cron pour une base de données autre que la base de données par défaut.

Octroi d'autorisations utilisateurs de la base de données pour l'utilisation de pg_cron

L'installation de l'extension `pg_cron` requiert les privilèges `rds_superuser`. Toutefois, les autorisations d'utiliser le `pg_cron` peuvent être accordées (par un membre du groupe/rôle `rds_superuser`) à d'autres utilisateurs de la base de données, afin qu'ils puissent planifier leurs propres tâches. Nous vous recommandons de n'accorder des autorisations au schéma `cron` qu'en cas de besoin, si cela améliore les opérations dans votre environnement de production.

Pour accorder à un utilisateur de base de données des autorisations dans le schéma `cron`, exécutez la commande suivante :

```
postgres=> GRANT USAGE ON SCHEMA cron TO db-user;
```

Cela donne *db-user* l'autorisation d'accéder au `cron` schéma pour planifier des tâches cron pour les objets auxquels ils sont autorisés à accéder. Si l'utilisateur de la base de données ne dispose pas des autorisations nécessaires, la tâche échoue après avoir validé le message d'erreur dans le fichier `postgresql.log`, comme indiqué ci-dessous :

```
2020-12-08 16:41:00 UTC::@:[30647]:ERROR: permission denied for table table-name
2020-12-08 16:41:00 UTC::@:[27071]:LOG: background worker "pg_cron" (PID 30647) exited
 with exit code 1
```

En d'autres termes, assurez-vous que les utilisateurs de la base de données disposant d'autorisations sur le schéma `cron` ont également des autorisations sur les objets (tables, schémas, etc.) qu'ils prévoient de planifier.

Les détails de la tâche cron et sa réussite ou son échec sont également saisis dans la table `cron.job_run_details`. Pour plus d'informations, consultez Tableaux pour planifier les tâches et capturer leur statut .

Planification des tâches pg_cron

Les sections suivantes montrent comment vous pouvez planifier diverses tâches de gestion à l'aide de tâches `pg_cron`.

> **Note**
>
> Lorsque vous créez des tâches `pg_cron`, vérifiez que le paramètre `max_worker_processes` est supérieur au nombre de `cron.max_running_jobs`. Une

tâche pg_cron échoue si elle manque de processus de travail en arrière-plan. Le nombre de tâches pg_cron par défaut est de 5. Pour plus d'informations, consultez Paramètres de gestion de l'extension pg_cron.

Rubriques

- Vidage d'une table
- Purge de la table Historique pg_cron
- Journalisation des erreurs dans le fichier postgresql.log uniquement
- Planification d'une tâche cron pour une base de données autre que la base de données par défaut

Vidage d'une table

Autovacuum gère la maintenance dans la plupart des cas. Toutefois, vous pouvez vider une table spécifique quand bon vous semble.

L'exemple suivant montre comment utiliser la fonction `cron.schedule` pour configurer une tâche de manière à ce qu'elle utilise `VACUUM FREEZE` sur une table spécifique tous les jours à 22:00 (GMT).

```
SELECT cron.schedule('manual vacuum', '0 22 * * *', 'VACUUM FREEZE pgbench_accounts');
 schedule
----------
1
(1 row)
```

Une fois l'exemple précédent exécuté, vous pouvez vérifier l'historique dans la table `cron.job_run_details` comme suit.

```
postgres=> SELECT * FROM cron.job_run_details;
jobid  | runid | job_pid | database | username | command                          |
 status    | return_message | start_time                    | end_time
-------+-------+---------+----------+----------+-------------------------------
+-----------+----------------+-------------------------------
+-------------------------------
 1     | 1     | 3395    | postgres | adminuser| vacuum freeze pgbench_accounts
 | succeeded | VACUUM         | 2020-12-04 21:10:00.050386+00 | 2020-12-04
 21:10:00.072028+00
(1 row)
```

Voici une requête de la table `cron.job_run_details` permettant d'afficher les tâches qui ont échoué.

```
postgres=> SELECT * FROM cron.job_run_details WHERE status = 'failed';
jobid | runid | job_pid | database | username | command                        | status
 | return_message                                | start_time                    |
 end_time
------+-------+---------+----------+----------+--------------------------------+--------
+-----------------------------------------------+-------------------------------
+-------------------------------
 5    | 4     | 30339   | postgres | adminuser| vacuum freeze pgbench_account | failed
 | ERROR: relation "pgbench_account" does not exist | 2020-12-04 21:48:00.015145+00 |
 2020-12-04 21:48:00.029567+00
(1 row)
```

Pour plus d'informations, consultez Tableaux pour planifier les tâches et capturer leur statut .

Purge de la table Historique pg_cron

La table `cron.job_run_details` contient un historique des tâches cron et celui-ci peut considérablement s'étoffer au fil du temps. Nous vous recommandons de planifier une tâche afin de purger cette table. Par exemple, conserver les entrées d'une semaine peut s'avérer suffisant à des fins de dépannage.

L'exemple suivant utilise la fonction cron.schedule pour planifier une tâche qui s'exécute tous les jours à minuit afin de purger la table `cron.job_run_details`. Cette tâche ne conserve que les entrées des sept derniers jours. Utilisez votre compte `rds_superuser` pour planifier la tâche comme suit :

```
SELECT cron.schedule('0 0 * * *', $$DELETE
    FROM cron.job_run_details
    WHERE end_time < now() - interval '7 days'$$);
```

Pour plus d'informations, consultez Tableaux pour planifier les tâches et capturer leur statut .

Journalisation des erreurs dans le fichier postgresql.log uniquement

Pour empêcher les écritures dans la table `cron.job_run_details`, modifiez le groupe de paramètres associé à l'instance de base de données PostgreSQL et désactivez le paramètre `cron.log_run`. L'extension `pg_cron` n'écrit plus dans la table et consigne uniquement des erreurs

dans le fichier `postgresql.log`. Pour plus d'informations, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora.

Utilisez la commande suivante pour vérifier la valeur du paramètre `cron.log_run`.

```
postgres=> SHOW cron.log_run;
```

Pour plus d'informations, consultez Paramètres de gestion de l'extension pg_cron.

Planification d'une tâche cron pour une base de données autre que la base de données par défaut

Toutes les métadonnées de `pg_cron` sont conservées dans la base de données par défaut PostgreSQL nommée `postgres`. Des exécutants étant utilisés en arrière-plan pour exécuter les tâches de maintenance cron, vous pouvez planifier une tâche dans n'importe quelle base de données de l'instance de base de données PostgreSQL.

> **Note**
>
> Seuls les utilisateurs disposant du rôle `rds_superuser` ou de privilèges `rds_superuser` peuvent répertorier toutes les tâches cron de la base de données. Les autres utilisateurs peuvent uniquement consulter leurs propres tâches dans la table `cron.job`.

1. Dans la base de données cron, planifiez la tâche comme vous le faites normalement à l'aide de la fonction cron.schedule.

   ```
   postgres=> SELECT cron.schedule('database1 manual vacuum', '29 03 * * *', 'vacuum
    freeze test_table');
   ```

2. En tant qu'utilisateur ayant le rôle `rds_superuser`, mettez à jour la colonne de base de données correspondant à la tâche que vous venez de créer de manière à l'exécuter dans une autre base de données de votre instance de base de données PostgreSQL.

   ```
   postgres=> UPDATE cron.job SET database = 'database1' WHERE jobid = 106;
   ```

3. Procédez à une vérification en interrogeant la table `cron.job`.

   ```
   postgres=> SELECT * FROM cron.job;
   jobid | schedule    | command                          | nodename  | nodeport |
    database | username  | active | jobname
   ```

```
------+-------------+--------------------------------+-----------+----------
+----------+-----------+--------+------------------------
106   | 29 03 * * * | vacuum freeze test_table       | localhost | 8192     |
 database1| adminuser | t      | database1 manual vacuum
  1   | 59 23 * * * | vacuum freeze pgbench_accounts | localhost | 8192     |
 postgres | adminuser | t      | manual vacuum
(2 rows)
```

> **Note**
>
> Dans certains cas, vous pouvez ajouter une tâche cron que vous avez l'intention d'exécuter sur une base de données différente. Dans de tels cas, le tâche peut essayer de s'exécuter dans la base de données par défaut (`postgres`) avant la mise à jour de la colonne de base de données correcte. Si le nom d'utilisateur dispose d'autorisations, la tâche s'exécute correctement dans la base de données par défaut.

Référence pour l'extension pg_cron

Vous pouvez utiliser les paramètres, fonctions et tables suivants avec l'extension `pg_cron`. Pour plus d'informations, consultez Qu'est-ce que pg_cron dans la documentation pg_cron.

Rubriques

- Paramètres de gestion de l'extension pg_cron
- Référence de fonction : cron.schedule
- Référence de fonction : cron.unschedule
- Tableaux pour planifier les tâches et capturer leur statut

Paramètres de gestion de l'extension pg_cron

La liste ci-dessous répertorie les paramètres permettant de contrôler le comportement de l'extension `pg_cron`.

Paramètre	Description
cron.database_name	Base de données dans laquelle les métadonnées `pg_cron` sont conservées.

Paramètre	Description
`cron.host`	Nom d'hôte permettant de se connecter à PostgreSQL. Vous ne pouvez pas modifier cette valeur.
`cron.log_run`	Enregistrez chaque tâche qui s'exécute dans la table `job_run_details` . Les valeurs sont on ou `off`. Pour plus d'informations, consultez Tableaux pour planifier les tâches et capturer leur statut .
`cron.log_statement`	Enregistre toutes les instructions cron avant leur exécution. Les valeurs sont on ou `off`.
`cron.max_running_jobs`	Nombre maximal de tâches pouvant être exécutées simultanément.
`cron.use_background_workers`	Utilisez des exécutants en arrière-plan plutôt que des sessions client. Vous ne pouvez pas modifier cette valeur.

Utilisez la commande SQL suivante pour afficher ces paramètres et leurs valeurs.

```
postgres=> SELECT name, setting, short_desc FROM pg_settings WHERE name LIKE 'cron.%'
 ORDER BY name;
```

Référence de fonction : cron.schedule

Cette fonction planifie une tâche cron. Cette tâche est initialement planifiée dans la base de données `postgres` par défaut. La fonction renvoie une valeur `bigint` correspondant à l'identifiant de la tâche. Pour planifier l'exécution de tâches dans d'autres bases de données de votre instance de base de données PostgreSQL, consultez l'exemple disponible dans Planification d'une tâche cron pour une base de données autre que la base de données par défaut.

La fonction présente deux formats de syntaxe.

Syntaxe

```
cron.schedule (job_name,
    schedule,
    command
);

cron.schedule (schedule,
    command
);
```

Paramètres

Paramètre	Description
`job_name`	Nom de la tâche cron.
`schedule`	Texte indiquant la planification de la tâche cron. Le format correspond au format cron standard.
`command`	Texte de la commande à exécuter.

Exemples

```
postgres=> SELECT cron.schedule ('test','0 10 * * *', 'VACUUM pgbench_history');
 schedule
----------
      145
(1 row)

postgres=> SELECT cron.schedule ('0 15 * * *', 'VACUUM pgbench_accounts');
 schedule
----------
      146
(1 row)
```

Référence de fonction : cron.unschedule

Cette fonction supprime une tâche cron. Vous pouvez spécifier soit le `job_name` ou le `job_id`. Une politique assure que vous soyez le propriétaire pouvant supprimer la planification de la tâche. La fonction renvoie une valeur booléenne indiquant la réussite ou l'échec.

La fonction a les formats de syntaxe suivants.

Syntaxe

```
cron.unschedule (job_id);

cron.unschedule (job_name);
```

Paramètres

Paramètre	Description
`job_id`	Identifiant de tâche renvoyé par la fonction `cron.schedule` lors de la planification de la tâche cron.
`job_name`	Nom d'une tâche cron planifiée avec la fonction `cron.schedule` .

Exemples

```
postgres=> SELECT cron.unschedule(108);
 unschedule
------------
 t
(1 row)

postgres=> SELECT cron.unschedule('test');
 unschedule
------------
 t
(1 row)
```

Tableaux pour planifier les tâches et capturer leur statut

Les tables suivantes sont utilisées pour planifier les tâches cron et enregistrer la façon dont elles ont été accomplies.

Tableau	Description
`cron.job`	Contient les métadonnées relatives à chaque tâche planifiée. La plupart des interactions avec cette table doivent être effectuées à l'aide des fonctions `cron.schedule` et `cron.unschedule`. **Important** Nous vous recommandons de ne pas accorder de privilèges de mise à jour ou d'insertion directement à cette table. Ce faisant, l'utilisateur pourrait mettre à jour la colonne `username` à exécuter en tant que `rds-superuser`.
`cron.job_run_details`	Contient des informations historiques sur l'exécution de tâches planifiées antérieures. Ces informations sont utiles pour examiner l'état, les messages renvoyés et les heures de début et de fin d'exécution de la tâche. **Note** Pour éviter que cette table évolue indéfiniment, purgez-la de manière régulière. Pour obtenir un exemple, consultez Purge de la table Historique pg_cron.

Utilisation de pgAudit pour journaliser l'activité de la base de données

Les institutions financières, les agences gouvernementales et de nombreux secteurs doivent tenir des journaux d'audit pour se conformer aux exigences réglementaires. En utilisant l'extension d'audit PostgreSQL (pgAudit) avec votre cluster de bases de données Aurora PostgreSQL, vous pouvez

capturer les enregistrements détaillés généralement utiles aux auditeurs ou pour répondre aux exigences réglementaires. Par exemple, vous pouvez configurer l'extension pgAudit pour suivre les modifications apportées à des bases de données et à des tables spécifiques, pour enregistrer l'utilisateur qui a effectué la modification et de nombreux autres détails.

L'extension pgAudit s'appuie sur les fonctionnalités de l'infrastructure de journalisation PostgreSQL native en étendant les messages de journal de manière plus détaillée. En d'autres termes, vous utilisez la même approche pour consulter votre journal d'audit que pour consulter les messages du journal. Pour plus d'informations sur la journalisation PostgreSQL, consultez Fichiers journaux de base de données Aurora PostgreSQL.

L'extension pgAudit supprime les données sensibles, telles que les mots de passe en texte clair, des journaux. Si votre cluster de bases de données Aurora PostgreSQL est configuré(e) pour enregistrer les instructions du langage de manipulation de données (DML) comme indiqué dans Activer la journalisation des requêtes pour votre cluster de bases de données Aurora PostgreSQL., vous pouvez éviter le problème de mot de passe en texte clair en utilisant l'extension PostgreSQL Audit.

Vous pouvez configurer l'audit sur vos instances de base de données avec une grande précision. Vous pouvez auditer toutes les bases de données et tous les utilisateurs. Vous pouvez également choisir de n'auditer que certaines bases de données, certains utilisateurs et d'autres objets. Vous pouvez également exclure explicitement certains utilisateurs et certaines bases de données de l'audit. Pour de plus amples informations, consultez Exclusion d'utilisateurs ou de bases de données de la journalisation d'audit.

Compte tenu de la quantité de détails qui peuvent être capturés, nous vous recommandons, si vous utilisez pgAudit, de surveiller votre consommation de stockage.

L'extension pgAudit est prise en charge sur toutes les versions d'Aurora PostgreSQL disponibles. Pour une liste des versions de pgAudit prises en charge par la version d'Aurora PostgreSQL, consultez Versions d'extension pour Amazon Aurora PostgreSQL dans les Notes de mise à jour d'Aurora PostgreSQL.

Rubriques

Configuration de l’extension pgAudit

Pour configurer l’extension pgAudit sur votre cluster de bases de données Aurora PostgreSQL, vous devez d’abord ajouter pgAudit aux bibliothèques partagées sur le groupe de paramètres de cluster de bases de données personnalisé pour votre cluster de bases de données Aurora PostgreSQL. Pour plus d’informations sur la création d’un groupe de paramètres de cluster de bases de données, consultez Groupes de paramètres pour Amazon Aurora. Ensuite, vous installez l’extension pgAudit. Enfin, vous spécifiez les bases de données et les objets que vous souhaitez auditer. Les procédures de cette section vous guident. Pour ce faire, vous pouvez utiliser la AWS Management Console ou la AWS CLI.

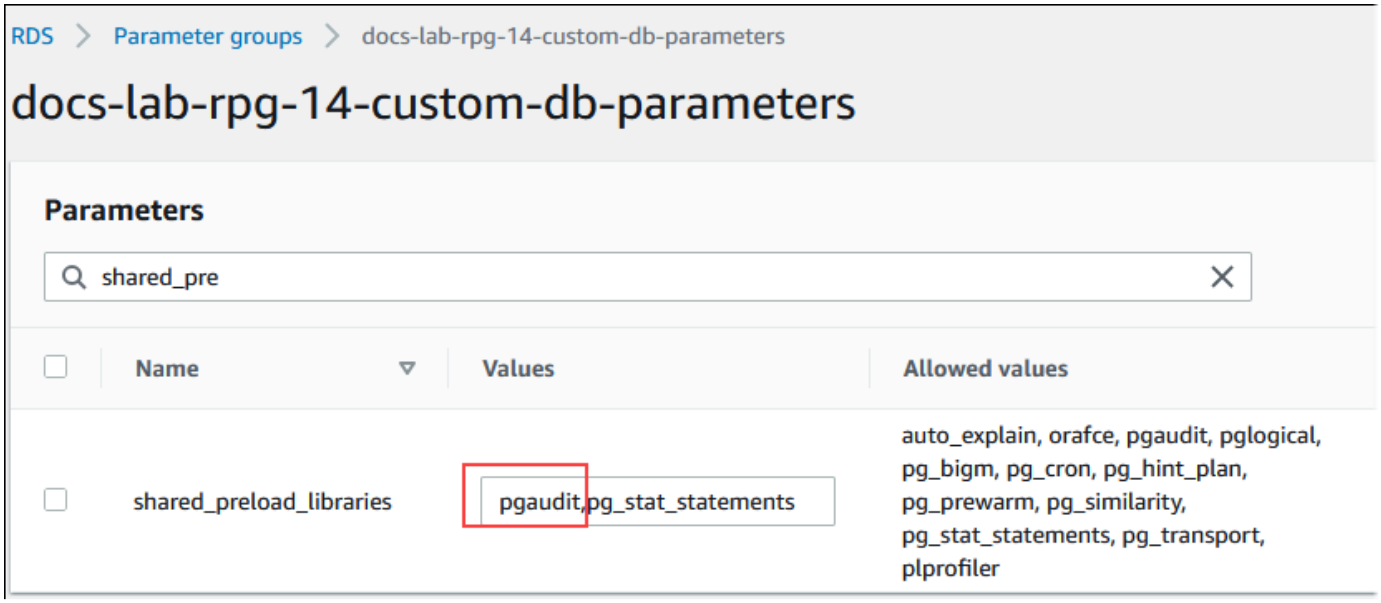
Vous devez disposer d’autorisations en tant que rôle `rds_superuser` pour effectuer toutes ces tâches.

Les étapes suivantes supposent que votre cluster de bases de données Aurora PostgreSQL est associé(e) à un groupe de paramètres de cluster de bases de données personnalisé.

Console

Configurer l’extension pgAudit

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez l’instance d’enregistreur de votre cluster de bases de données Aurora PostgreSQL .
3. Ouvrez l’onglet Configuration de l’instance d’enregistreur de votre cluster de bases de données Aurora PostgreSQL. Parmi les détails de l’instance, trouvez le lien Groupe de paramètres.
4. Cliquez sur le lien pour ouvrir les paramètres personnalisés associés à votre cluster de bases de données Aurora PostgreSQL.
5. Dans le champ de recherche Parameters (Paramètres), tapez `shared_pre` pour trouver le paramètre `shared_preload_libraries`.
6. Choisissez Edit parameters (Modifier les paramètres) pour accéder aux valeurs des propriétés.
7. Ajoutez `pgaudit` à la liste dans le champ Values (Valeurs). Utilisez une virgule pour séparer les éléments de la liste de valeurs.



8. Redémarrez l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL afin que vos modifications du paramètre `shared_preload_libraries` prennent effet.
9. Lorsque l'instance est disponible, vérifiez que pgAudit a été initialisé. Utilisez `psql` pour vous connecter à l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL, puis exécutez la commande suivante.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

10. Une fois pgAudit initialisé, vous pouvez maintenant créer l'extension. Vous devez créer l'extension après avoir initialisé la bibliothèque, car l'extension `pgaudit` installe des déclencheurs d'événements pour auditer les instructions du langage de définition des données (DDL).

```
CREATE EXTENSION pgaudit;
```

11. Fermez la session `psql`.

```
labdb=> \q
```

12. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

13. Trouvez le paramètre `pgaudit.log` dans la liste et définissez la valeur appropriée pour votre cas d'utilisation. Par exemple, la définition du paramètre `pgaudit.log` en `write` comme indiqué dans l'image suivante permet de capturer des insertions, des mises à jour, des suppressions et d'autres types de modifications dans le journal.

Vous pouvez également choisir l'une des valeurs suivantes pour le paramètre `pgaudit.log`.

- none – La valeur par défaut. Aucune modification de base de données n'est journalisée.
- all – Journalise tout (lecture, écriture, fonction, rôle, ddl, divers).
- ddl – Journalise toutes les instructions en langage de définition de données (DDL) qui ne sont pas incluses dans la classe `ROLE`.
- function – Journalise les appels de fonction et les blocs `DO`.
- misc – Journalise diverses commandes, telles que `DISCARD`, `FETCH`, `CHECKPOINT`, `VACUUM` et `SET`.
- read – Journalise `SELECT` et `COPY` lorsque la source est une relation (comme une table) ou une requête.
- role – Journalise les instructions relatives aux rôles et privilèges, telles que `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` et `DROP ROLE`.
- write – Journalise `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` et `COPY` lorsque la destination est une relation (table).

14. Sélectionnez Enregistrer les modifications.
15. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

16. Choisissez l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL dans la liste des bases de données.

AWS CLI

Configurer pgAudit

Pour configurer pgAudit à l'aide de l'AWS CLI, vous devez appeler l'opération modify-db-parameter-group afin de modifier les paramètres du journal d'audit dans votre groupe de paramètres personnalisé, comme indiqué dans la procédure suivante.

1. Utilisez la commande AWS CLI suivante pour ajouter `pgaudit` au paramètre `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pgaudit,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilisez la commande AWS CLI suivante pour redémarrer l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL afin que la bibliothèque pgAudit soit initialisée.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Lorsque l'instance est disponible, vous pouvez vérifier que `pgaudit` a été initialisé. Utilisez `psql` pour vous connecter à l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL, puis exécutez la commande suivante.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pgaudit
(1 row)
```

   Une fois pgAudit initialisé, vous pouvez maintenant créer l'extension.

```
CREATE EXTENSION pgaudit;
```

4. Fermez la session `psql` afin de pouvoir utiliser l'AWS CLI.

```
labdb=> \q
```

5. Utilisez la commande AWS CLI suivante pour spécifier les classes d'instructions qui doivent être journalisées par journalisation des audits de session. L'exemple définit le paramètre `pgaudit.log` sur `write`, qui capture les insertions, les mises à jour et les suppressions dans le journal.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.log,ParameterValue=write,ApplyMethod=pending-reboot" \
   --region aws-region
```

Vous pouvez également choisir l'une des valeurs suivantes pour le paramètre `pgaudit.log`.

- none – La valeur par défaut. Aucune modification de base de données n'est journalisée.
- all – Journalise tout (lecture, écriture, fonction, rôle, ddl, divers).
- ddl – Journalise toutes les instructions en langage de définition de données (DDL) qui ne sont pas incluses dans la classe `ROLE`.
- function – Journalise les appels de fonction et les blocs `DO`.
- misc – Journalise diverses commandes, telles que `DISCARD`, `FETCH`, `CHECKPOINT`, `VACUUM` et `SET`.
- read – Journalise `SELECT` et `COPY` lorsque la source est une relation (comme une table) ou une requête.
- role – Journalise les instructions relatives aux rôles et privilèges, telles que `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` et `DROP ROLE`.
- write – Journalise `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` et `COPY` lorsque la destination est une relation (table).

Redémarrez l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL, à l'aide de la commande AWS CLI suivante.

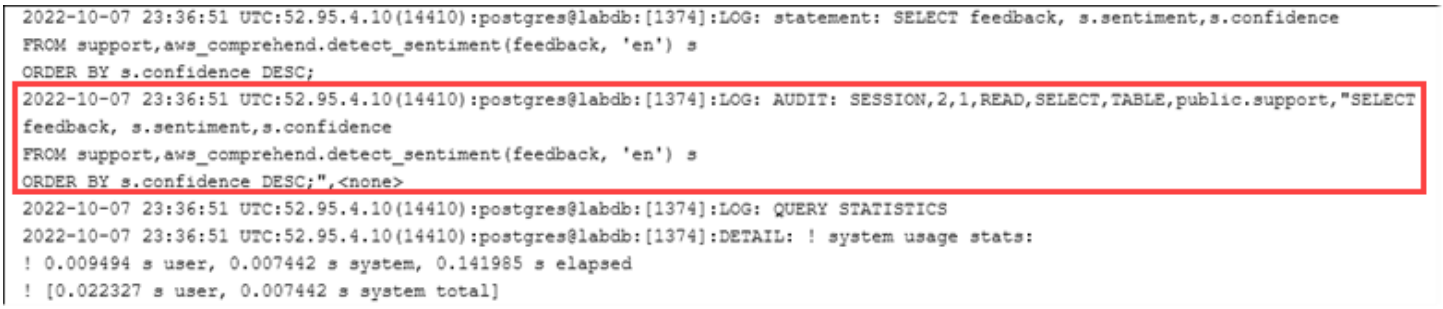
```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

Audit d'objets de base de données

Une fois que pgAudit est défini sur votre cluster de bases de données Aurora PostgreSQL et qu'il est configuré en fonction de vos besoins, des informations plus détaillées sont capturées dans le journal PostgreSQL. Par exemple, alors que la configuration de journalisation PostgreSQL par défaut identifie la date et l'heure auxquelles une modification a été apportée à une table de base de données, avec l'extension pgAudit, l'entrée du journal peut inclure le schéma, l'utilisateur qui a effectué la modification et d'autres détails en fonction de la manière dont les paramètres de l'extension sont configurés. Vous pouvez configurer l'audit pour suivre les modifications de différentes manières.

- Pour chaque session, par utilisateur. Au niveau de la session, vous pouvez capturer le texte de commande complet.
- Pour chaque objet, par utilisateur et par base de données.

La fonctionnalité d'audit des objets est activée lorsque vous créez le rôle `rds_pgaudit` sur votre système, puis que vous ajoutez ce rôle au paramètre `pgaudit.role` dans votre groupe de paramètres personnalisé. Par défaut, le paramètre `pgaudit.role` n'est pas défini et la seule valeur autorisée est `rds_pgaudit`. Les étapes suivantes supposent que `pgaudit` a été initialisé et que vous avez créé l'extension `pgaudit` en suivant la procédure décrite dans Configuration de l'extension pgAudit.



Comme le montre cet exemple, la ligne « LOG: AUDIT: SESSION » fournit des informations sur la table et son schéma, entre autres détails.

Configurer l'audit d'objets

1. Utilisez `psql` pour vous connecter à l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL..

```
psql --host=your-instance-name.aws-region.rds.amazonaws.com --port=5432 --
username=postgrespostgres --password --dbname=labdb
```

2. Créez un rôle de base de données appelé `rds_pgaudit` à l'aide de la commande suivante.

```
labdb=> CREATE ROLE rds_pgaudit;
CREATE ROLE
labdb=>
```

3. Fermez la session `psql`.

```
labdb=> \q
```

   Dans les étapes suivantes, utilisez l'AWS CLI pour modifier les paramètres du journal d'audit dans votre groupe de paramètres personnalisé.

4. Utilisez la commande AWS CLI suivante pour définir le paramètre `pgaudit.role` à `rds_pgaudit`. Par défaut, ce paramètre est vide et `rds_pgaudit` est la seule valeur autorisée.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=pgaudit.role,ParameterValue=rds_pgaudit,ApplyMethod=pending-reboot"
 \
   --region aws-region
```

5. Utilisez la commande AWS CLI suivante pour redémarrer l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL afin que les modifications apportées aux paramètres prennent effet.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

6. Exécutez la commande suivante pour confirmer que `pgaudit.role` est défini sur `rds_pgaudit`.

```
SHOW pgaudit.role;
pgaudit.role
------------------
rds_pgaudit
```

Pour tester la journalisation pgAudit, vous pouvez exécuter plusieurs exemples de commandes que vous souhaitez auditer. Par exemple, vous pouvez exécuter les commandes suivantes.

```
CREATE TABLE t1 (id int);
GRANT SELECT ON t1 TO rds_pgaudit;
SELECT * FROM t1;
id
----
(0 rows)
```

Les journaux de base de données doivent contenir une entrée similaire à ce qui suit.

```
...
2017-06-12 19:09:49 UTC:...:rds_test@postgres:[11701]:LOG: AUDIT:
OBJECT,1,1,READ,SELECT,TABLE,public.t1,select * from t1;
...
```

Pour obtenir des informations sur l'affichage des journaux, veuillez consulter Surveillance des fichiers journaux Amazon Aurora.

Pour en savoir plus sur l'extension pgAudit, veuillez consulter pgAudit sur GitHub.

Exclusion d'utilisateurs ou de bases de données de la journalisation d'audit

Comme indiqué dans Fichiers journaux de base de données Aurora PostgreSQL, les journaux PostgreSQL consomment de l'espace de stockage. L'utilisation de l'extension pgAudit augmente le volume de données collectées dans vos journaux à des degrés divers, en fonction des modifications que vous suivez. Vous n'avez peut-être pas besoin d'auditer chaque utilisateur ou base de données de votre cluster de bases de données Aurora PostgreSQL.

Pour minimiser les impacts sur votre stockage et éviter de capturer inutilement des enregistrements d'audit, vous pouvez exclure les utilisateurs et les bases de données de l'audit. Vous pouvez également modifier la journalisation au cours d'une session donnée. Les exemples suivants montrent comment procéder.

> **Note**
>
> Les paramètres au niveau de la session ont priorité sur les paramètres du groupe de paramètres du cluster de bases de données personnalisé pour l'instance d'écriture du cluster de bases de données Aurora PostgreSQL. Si vous ne souhaitez pas que les utilisateurs de base de données contournent vos paramètres de configuration de journalisation des audits, veillez à modifier leurs autorisations.

Supposons que votre cluster de bases de données Aurora PostgreSQL soit configuré(e) pour auditer le même niveau d'activité pour tous les utilisateurs et bases de données. Vous pouvez ensuite décider de ne pas auditer l'utilisateur `myuser`. Vous pouvez désactiver l'audit pour `myuser` à l'aide de la commande SQL suivante.

```
ALTER USER myuser SET pgaudit.log TO 'NONE';
```

Vous pouvez ensuite utiliser la requête suivante pour vérifier la colonne `user_specific_settings` pour `pgaudit.log` afin de confirmer que le paramètre est défini sur `NONE`.

```
SELECT
    usename AS user_name,
    useconfig AS user_specific_settings
FROM
    pg_user
WHERE
    usename = 'myuser';
```

Vous devez voir la sortie suivante.

```
 user_name | user_specific_settings
-----------+------------------------
 myuser    | {pgaudit.log=NONE}
(1 row)
```

Vous pouvez désactiver la journalisation pour un utilisateur donné au cours de sa session avec la base de données à l'aide de la commande suivante.

```
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'none';
```

Utilisez la requête suivante pour vérifier la colonne des paramètres du fichier pgaudit.log pour une combinaison utilisateur et base de données spécifique.

```
SELECT
    usename AS "user_name",
    datname AS "database_name",
    pg_catalog.array_to_string(setconfig, E'\n') AS "settings"
FROM
    pg_catalog.pg_db_role_setting s
    LEFT JOIN pg_catalog.pg_database d ON d.oid = setdatabase
    LEFT JOIN pg_catalog.pg_user r ON r.usesysid = setrole
WHERE
    usename = 'myuser'
    AND datname = 'mydatabase'
ORDER BY
    1,
    2;
```

Vous voyez des résultats similaires à ce qui suit.

```
  user_name | database_name |     settings
-----------+---------------+------------------
 myuser    | mydatabase    | pgaudit.log=none
(1 row)
```

Après avoir désactivé l'audit pour `myuser`, vous décidez de ne pas suivre les modifications apportées à `mydatabase`. Vous pouvez désactiver l'audit pour cette base de données spécifique à l'aide de la commande suivante.

```
ALTER DATABASE mydatabase SET pgaudit.log to 'NONE';
```

Utilisez ensuite la requête suivante pour vérifier la colonne database_specific_settings afin de confirmer que le fichier pgaudit.log est défini sur NONE.

```
SELECT
a.datname AS database_name,
b.setconfig AS database_specific_settings
FROM
pg_database a
FULL JOIN pg_db_role_setting b ON a.oid = b.setdatabase
WHERE
```

```
a.datname = 'mydatabase';
```

Vous devez voir la sortie suivante.

```
 database_name | database_specific_settings
---------------+----------------------------
 mydatabase    | {pgaudit.log=NONE}
(1 row)
```

Pour rétablir les paramètres par défaut pour myuser, utilisez la commande suivante :

```
ALTER USER myuser RESET pgaudit.log;
```

Pour rétablir les paramètres par défaut pour une base de données, utilisez la commande suivante.

```
ALTER DATABASE mydatabase RESET pgaudit.log;
```

Pour rétablir les paramètres par défaut pour l'utilisateur et la base de données, utilisez la commande suivante.

```
ALTER USER myuser IN DATABASE mydatabase RESET pgaudit.log;
```

Vous pouvez également capturer des événements spécifiques dans le journal en définissant `pgaudit.log` pour l'une des autres valeurs autorisées pour le paramètre `pgaudit.log`. Pour de plus amples informations, consultez Liste des paramètres autorisés pour le paramètre `pgaudit.log`.

```
ALTER USER myuser SET pgaudit.log TO 'read';
ALTER DATABASE mydatabase SET pgaudit.log TO 'function';
ALTER USER myuser IN DATABASE mydatabase SET pgaudit.log TO 'read,function'
```

Référence pour l'extension pgAudit

Vous pouvez spécifier le niveau de détail que vous souhaitez pour votre journal d'audit en modifiant un ou plusieurs des paramètres répertoriés dans cette section.

Contrôle du comportement de pgAudit

Vous pouvez contrôler la journalisation d'audit en modifiant un ou plusieurs des paramètres répertoriés dans la table suivante.

Paramètre	Description
`pgaudit.log`	Spécifie quelles classes d'instructions seront journalisées par la journalisation de l'audit de session. Les valeurs autorisées incluent ddl, function, misc, read, role, write, none, all. Pour de plus amples informations, consultez Liste des paramètres autorisés pour le paramètre `pgaudit.log` .
`pgaudit.log_catalog`	Lorsque cette option est activée (définie sur 1), cela ajoute des instructions à la piste d'audit si toutes les relations d'une instruction se trouvent dans pg_catalog.
`pgaudit.log_level`	Spécifie le niveau de journal qui sera utilisé pour les entrées de journal. Valeurs autorisées : debug5, debug4, debug3, debug2, debug1, info, notice, warning, log
`pgaudit.log_parameter`	Lorsque cette option est activée (définie sur 1), les paramètres transmis avec l'instruction sont capturés dans le journal d'audit.
`pgaudit.log_relation`	Lorsque cette option est activée (définie sur 1), le journal d'audit de session crée une entrée de journal distincte pour chaque relation (TABLE, VIEW, etc.) référencée dans une instruction SELECT ou DML.
`pgaudit.log_statement_once`	Spécifie si la journalisation inclura le texte de l'instruction et les paramètres avec la première entrée de journal pour une combinaison instruction/sous-instruction ou avec chaque entrée.
`pgaudit.role`	Spécifie le rôle principal à utiliser pour la journalisation de l'audit des objets. La seule entrée autorisée est `rds_pgaudit` .

Liste des paramètres autorisés pour le paramètre **`pgaudit.log`**

Valeur	Description
none	Il s’agit de l’option par défaut. Aucune modification de base de données n'est journalisée.

Valeur	Description
Tout	Journalise tout (lecture, écriture, fonction, rôle, ddl, divers).
ddl	Journalise toutes les instructions en langage de définition de données (DDL) qui ne sont pas incluses dans la classe `ROLE`.
fonction	Journalise les appels de fonction et les blocs `DO`.
Misc	Journalise diverses commandes, telles que `DISCARD`, `FETCH`, `CHECKPOINT` , `VACUUM` et `SET`.
lire	Journalise `SELECT` et `COPY` lorsque la source est une relation (comme une table) ou une requête.
rôle	Journalise les instructions relatives aux rôles et privilèges, telles que `GRANT`, `REVOKE`, `CREATE ROLE`, `ALTER ROLE` et `DROP ROLE`.
write	Journalise `INSERT`, `UPDATE`, `DELETE`, `TRUNCATE` et `COPY` lorsque la destination est une relation (table).

Pour journaliser plusieurs types d'événements avec l'audit de session, utilisez une liste séparée par des virgules. Pour journaliser tous les types d'événements, définissez `pgaudit.log` à la valeur `ALL`. Redémarrez l'instance de base de données pour appliquer les modifications.

Avec les audits d'objet, vous pouvez affiner la journalisation d'audit pour que celle-ci fonctionne avec des relations spécifiques. Par exemple, vous pouvez spécifier que vous souhaitez une journalisation d'audit pour les opérations `READ` sur une ou plusieurs tables.

Utilisation de pglogical pour synchroniser les données entre les instances

Toutes les versions d'Aurora PostgreSQL actuellement disponibles prennent en charge l'extension `pglogical`. L'extension pglogical est antérieure à la fonction de réplication logique qui fonctionne de la même manière et qui a été introduite par PostgreSQL dans la version 10. Pour plus d'informations, consultez Présentation de la réplication logique PostgreSQL avec Aurora.

L'extension `pglogical` prend en charge la réplication logique entre deux ou plusieurs clusters de bases de données Aurora PostgreSQL. Elle prend également en charge la réplication entre différentes versions de PostgreSQL, ainsi qu'entre des bases de données fonctionnant sur RDS

pour les instances de base de données PostgreSQL et les clusters de bases de données Aurora PostgreSQL. L'extension `pglogical` utilise un modèle de publication et d'abonnement pour répliquer les changements apportés aux tables et aux autres objets, tels que les séquences, d'un serveur de publication à un abonné. Elle s'appuie sur un emplacement de réplication pour assurer la synchronisation des changements d'un nœud de serveur de publication à un nœud abonné, défini comme suit.

- Le nœud de serveur de publication est le cluster de bases de données Aurora PostgreSQL qui est la source des données à répliquer vers les autres nœuds. Le nœud de serveur de publication définit les tables à répliquer dans un ensemble de publication.
- Le nœud abonné est le cluster de bases de données Aurora PostgreSQL qui reçoit les mises à jour WAL du serveur de publication. L'abonné crée un abonnement pour se connecter au serveur de publication et obtenir les données WAL décodées. Lorsque l'abonné crée l'abonnement, l'emplacement de réplication est créé sur le nœud de serveur de publication.

Vous trouverez ci-après des informations sur la configuration de l'extension `pglogical`.

Rubriques

Exigences et limites de l'extension pglogique

Toutes les versions actuellement disponibles d'Aurora PostgreSQL prennent en charge l'extension `pglogical`.

Le nœud de serveur de publication et le nœud abonné doivent tous deux être configurés pour la réplication logique.

Les tables que vous voulez répliquer d'un diffuseur de publication à un abonné doivent avoir les mêmes noms et le même schéma. Ces tables doivent également contenir les mêmes colonnes, et les colonnes doivent utiliser les mêmes types de données. Les tables des serveurs de publication et des

abonnés doivent avoir les mêmes clés primaires. Nous vous recommandons d'utiliser uniquement PRIMARY KEY comme contrainte unique.

Les tables du nœud abonné peuvent avoir des contraintes plus permissives que celles du nœud de serveur de publication pour les contraintes CHECK et NOT NULL.

L'extension `pglogical` fournit des fonctionnalités telles que la réplication bidirectionnelle qui ne sont pas prises en charge par la fonctionnalité de réplication logique intégrée à PostgreSQL (versions 10 et ultérieures). Pour plus d'informations, consultez PostgreSQL bi-directional replication using pglogical (Réplication bidirectionnelle PostgreSQL utilisant pglogical).

Configuration de l'extension pglogical

Pour configurer l'extension `pglogical` sur votre cluster de bases de données Aurora PostgreSQL, vous ajoutez `pglogical` aux bibliothèques partagées sur le groupe de paramètres de cluster de bases de données personnalisé pour votre cluster de bases de données Aurora PostgreSQL. Vous devez également définir la valeur du paramètre `rds.logical_replication` sur `1`, pour activer le décodage logique. Enfin, vous créez l'extension dans la base de données. Vous pouvez utiliser la AWS Management Console ou AWS CLI pour ces tâches.

Vous devez disposer d'autorisations en tant que rôle `rds_superuser` pour effectuer ces tâches.

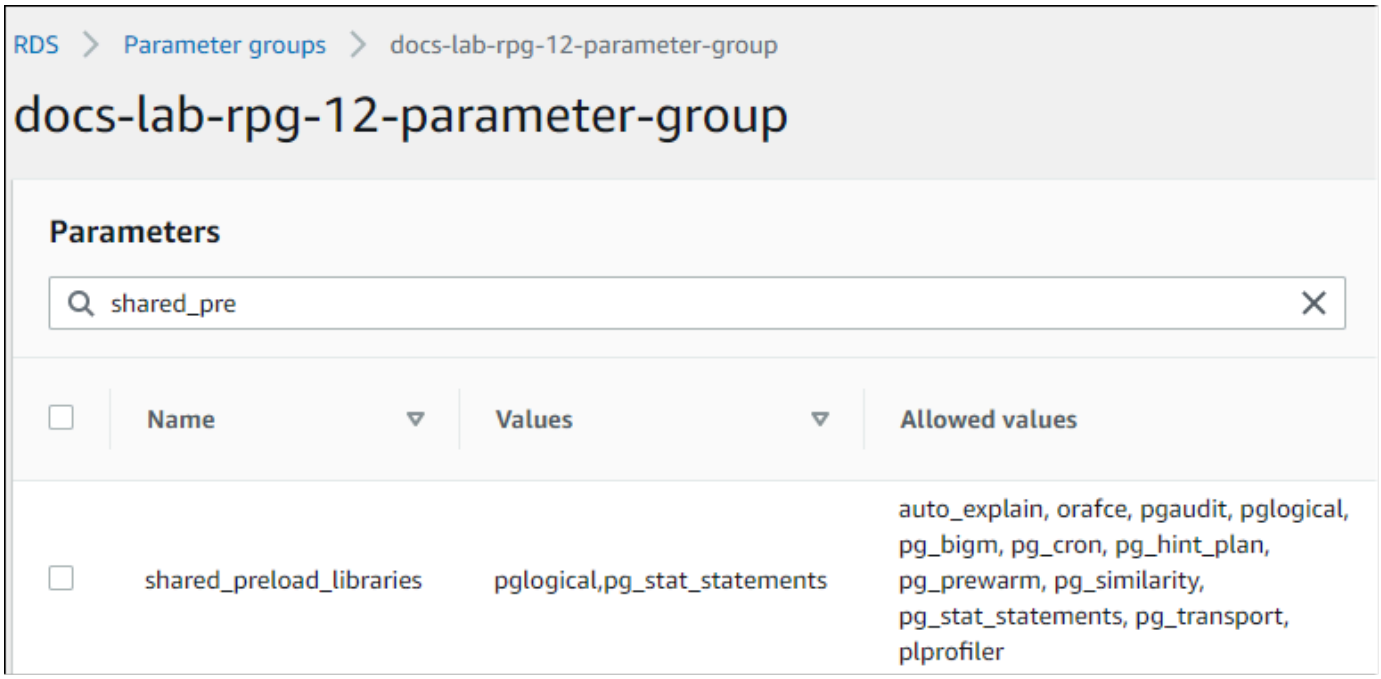
Les étapes suivantes supposent que votre cluster de bases de données Aurora PostgreSQL est associé(e) à un groupe de paramètres de cluster de bases de données personnalisé. Pour plus d'informations sur la création d'un groupe de paramètres de cluster de bases de données, consultez Groupes de paramètres pour Amazon Aurora.

console

Pour configurer l'extension pglogical

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le volet de navigation, choisissez votre instance d'écriture du cluster de bases de données Aurora PostgreSQL .
3. Ouvrez l'onglet Configuration pour votre instance d'écriture du cluster de bases de données Aurora PostgreSQL. Parmi les détails de l'instance, trouvez le lien Groupe de paramètres.
4. Cliquez sur le lien pour ouvrir les paramètres personnalisés associés à votre cluster de bases de données Aurora PostgreSQL.

5. Dans le champ de recherche Parameters (Paramètres), tapez `shared_pre` pour trouver le paramètre `shared_preload_libraries`.
6. Choisissez Edit parameters (Modifier les paramètres) pour accéder aux valeurs des propriétés.
7. Ajoutez `pglogical` à la liste dans le champ Values (Valeurs). Utilisez une virgule pour séparer les éléments de la liste de valeurs.



8. Recherchez le paramètre `rds.logical_replication` et définissez-le sur `1`, pour activer la réplication logique.
9. Redémarrez l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL pour que vos modifications soient prises en compte.
10. Lorsque l'instance est disponible, vous pouvez utiliser `psql` (ou pgAdmin) pour vous connecter à l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

11. Pour vérifier que pglogical est initialisé, exécutez la commande suivante.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pglogical
(1 row)
```

12. Vérifiez le paramètre qui active le décodage logique, comme suit.

```
SHOW wal_level;
wal_level
-----------
 logical
(1 row)
```

13. Créez l'extension, comme suit.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

14. Sélectionnez Enregistrer les modifications.
15. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
16. Sélectionnez votre instance d'écriture du cluster de bases de données Aurora PostgreSQL dans la liste des bases de données, puis choisissez Reboot (Redémarrer) dans le menu Actions.

AWS CLI

Pour configurer l'extension pglogical

Pour configurer pglogical à l'aide d'AWS CLI, vous appelez l'opération modify-db-parameter-group pour modifier certains paramètres dans votre groupe de paramètres personnalisé, comme indiqué dans la procédure suivante.

1. Utilisez la commande AWS CLI suivante pour ajouter `pglogical` au paramètre `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pglogical,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilisez la commande AWS CLI suivante pour définir `rds.logical_replication` sur `1`, afin d'activer la capacité de décodage logique pour l'instance d'écriture du cluster de bases de données Aurora PostgreSQL.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=rds.logical_replication,ParameterValue=1,ApplyMethod=pending-
reboot" \
   --region aws-region
```

3. Utilisez la commande AWS CLI suivante pour redémarrer l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL afin que la bibliothèque pglogical soit initialisée.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

4. Lorsque l'instance est disponible, utilisez `psql` pour vous connecter à l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL.

```
psql --host=111122223333.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

5. Créez l'extension, comme suit.

```
CREATE EXTENSION pglogical;
EXTENSION CREATED
```

6. Redémarrez l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL, à l'aide de la commande AWS CLI suivante.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

Configuration de la réplication logique pour le cluster de bases de données Aurora PostgreSQL

La procédure suivante vous montre comment démarrer la réplication logique entre deux clusters de bases de données Aurora PostgreSQL.. Les étapes supposent que la source (serveur de publication) et la cible (abonné) ont toutes deux l'extension `pglogical` configurée comme indiqué dans le document Configuration de l'extension pglogical.

> **Note**
>
> Le `node_name` du nœud d'un abonné ne peut pas commencer par `rds`.

Pour créer le nœud de serveur de publication et définir les tables à répliquer

Ces étapes supposent que votre cluster de bases de données Aurora PostgreSQL possède une instance d'enregistreur avec une base de données qui contient une ou plusieurs tables que vous voulez répliquer vers un autre nœud. Vous devez recréer la structure de la table du serveur de publication sur l'abonné, donc d'abord, récupérer la structure de la table si nécessaire. Vous pouvez le faire en utilisant la métacommande `psql \d` *`tablename`* et en créant ensuite la même table sur l'instance de l'abonné. La procédure suivante crée un exemple de table sur le serveur de publication (source) à des fins de démonstration.

1. Utilisez `psql` pour vous connecter à l'instance qui possède la table que vous voulez utiliser comme source pour les abonnés.

   ```
   psql --host=source-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password --dbname=labdb
   ```

   Si vous ne disposez pas d'une table existante que vous souhaitez répliquer, vous pouvez créer un exemple de table comme suit.

   a. Créez un exemple de table en utilisant l'instruction SQL suivante.

      ```
      CREATE TABLE docs_lab_table (a int PRIMARY KEY);
      ```

   b. Remplissez la table avec les données générées en utilisant l'instruction SQL suivante.

      ```
      INSERT INTO docs_lab_table VALUES (generate_series(1,5000));
      INSERT 0 5000
      ```

   c. Vérifiez que les données existent dans la table à l'aide de l'instruction SQL suivante.

      ```
      SELECT count(*) FROM docs_lab_table;
      ```

2. Identifiez ce cluster de bases de données Aurora PostgreSQL comme le nœud de serveur de publication, comme suit.

```
SELECT pglogical.create_node(
    node_name := 'docs_lab_provider',
    dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 dbname=labdb');
 create_node
-------------
   3410995529
(1 row)
```

3. Ajoutez la table que vous souhaitez répliquer à l'ensemble de réplication par défaut. Pour plus d'informations sur les ensembles de réplication, consultez Replication sets (Ensembles de réplication) dans la documentation pglogical.

```
SELECT pglogical.replication_set_add_table('default', 'docs_lab_table', 'true',
 NULL, NULL);
 replication_set_add_table
  --------------------------
  t
  (1 row)
```

La configuration du nœud de diffuseur de publication est terminée. Vous pouvez maintenant configurer le nœud abonné pour recevoir les mises à jour du serveur de publication.

Pour configurer le nœud abonné et créer un abonnement pour recevoir des mises à jour

Ces étapes supposent que le cluster de bases de données Aurora PostgreSQL a été configuré avec l'extension `pglogical`. Pour plus d'informations, consultez Configuration de l'extension pglogical.

1. Utilisez `psql` pour vous connecter à l'instance qui doit recevoir les mises à jour du serveur de publication.

```
psql --host=target-instance.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. Sur le cluster de bases de données Aurora PostgreSQL de l'abonné, créez la même table que celle qui existe sur le serveur de publication. Pour cet exemple, la table est `docs_lab_table`. Vous pouvez créer la table comme suit.

```
CREATE TABLE docs_lab_table (a int PRIMARY KEY);
```

3. Vérifiez que cette table est vide.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  0
(1 row)
```

4. Identifiez ce cluster de bases de données Aurora PostgreSQL comme le nœud abonné, comme suit.

```
SELECT pglogical.create_node(
    node_name := 'docs_lab_target',
    dsn := 'host=target-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=********');
 create_node
-------------
   2182738256
(1 row)
```

5. Créez l'abonnement.

```
SELECT pglogical.create_subscription(
   subscription_name := 'docs_lab_subscription',
   provider_dsn := 'host=source-instance.aws-region.rds.amazonaws.com port=5432
 sslmode=require dbname=labdb user=postgres password=*******',
   replication_sets := ARRAY['default'],
   synchronize_data := true,
   forward_origins := '{}' );
 create_subscription
---------------------
1038357190
(1 row)
```

Lorsque vous terminez cette étape, les données de la table du serveur de publication sont créées dans la table de l'abonné. Vous pouvez le vérifier en utilisant la requête SQL suivante.

```
SELECT count(*) FROM docs_lab_table;
 count
-------
  5000
```

```
(1 row)
```

À partir de ce moment, les modifications apportées à la table sur le serveur de publication sont répliquées sur la table sur l’abonné.

Rétablissement de la réplication logique après une mise à niveau majeure

Avant de pouvoir effectuer une mise à niveau majeure d'un cluster de bases de données Aurora PostgreSQL qui est configuré comme un nœud d'édition pour la réplication logique, vous devez supprimer tous les emplacements de réplication, même ceux qui ne sont pas actifs. Nous vous recommandons de détourner temporairement les transactions de base de données du nœud d'édition, de supprimer les emplacements de réplication, de mettre à niveau le cluster de bases de données Aurora PostgreSQL, puis de rétablir et de relancer la réplication.

Les emplacements de réplication sont hébergés uniquement sur le nœud de serveur de publication. Le nœud abonné Aurora PostgreSQL dans un scénario de réplication logique n'a pas d'emplacements à supprimer. Le processus de mise à niveau de la version majeure d'Aurora PostgreSQL prend en charge la mise à niveau de l'abonné vers une nouvelle version majeure de PostgreSQL indépendamment du nœud de serveur de publication. Cependant, le processus de mise à niveau perturbe le processus de réplication et interfère avec la synchronisation des données WAL entre le nœud de serveur de publication et le nœud abonné. Vous devez rétablir la réplication logique entre le serveur de publication et l'abonné après avoir mis à niveau le serveur de publication, l'abonné ou les deux. La procédure suivante vous montre comment déterminer que la réplication a été perturbée et comment résoudre le problème.

Détermination de la perturbation de la réplication logique

Vous pouvez déterminer que le processus de réplication a été interrompu en interrogeant le nœud de serveur de publication ou le nœud abonné, comme suit.

Pour vérifier le nœud de serveur de publication

- Utilisez `psql` pour vous connecter au nœud de serveur de publication, puis interrogez la fonction `pg_replication_slots`. Notez la valeur dans la colonne active. Normalement, cela renvoie la valeur `t` (true), ce qui montre que la réplication est active. Si la requête renvoie la valeur `f` (false), cela indique que la réplication vers l'abonné a cessé.

```
SELECT slot_name,plugin,slot_type,active FROM pg_replication_slots;
                    slot_name              |      plugin      | slot_type | active
```

```
-----------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_labcb4fa94_docs_lab3de412c | pglogical_output | logical   | f
(1 row)
```

Pour vérifier le nœud abonné

Sur le nœud abonné, vous pouvez vérifier l'état de la réplication de trois manières différentes.

- Consultez les journaux PostgreSQL sur le nœud abonné pour trouver des messages d'échec. Le journal identifie l'échec avec des messages qui incluent le code de sortie 1, comme indiqué ci-dessous.

```
2022-07-06 16:17:03 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 14610) exited with exit code 1
2022-07-06 16:19:44 UTC::@:[7361]:LOG: background worker "pglogical apply
 16404:2880255011" (PID 21783) exited with exit code 1
```

- Interrogez la fonction `pg_replication_origin`. Connectez-vous à la base de données sur le nœud abonné en utilisant `psql` et interrogez la fonction `pg_replication_origin`, comme suit.

```
SELECT * FROM pg_replication_origin;
 roident | roname
---------+--------
(0 rows)
```

L'ensemble de résultats vide signifie que la réplication a été perturbée. Normalement, vous obtenez un résultat qui ressemble au suivant.

```
  roident |                      roname
 ---------+---------------------------------------------------
        1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
 (1 row)
```

- Interrogez la fonction `pglogical.show_subscription_status` comme indiqué dans l'exemple suivant.

```
SELECT subscription_name,status,slot_name FROM pglogical.show_subscription_status();
     subscription_name | status |              slot_name
---====----------------+--------+-------------------------------------
 docs_lab_subscription | down   | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
```

```
(1 row)
```

Cette sortie montre que la réplication a été perturbée. Son statut est down. Normalement, la sortie indique le statut `replicating`.

Si votre processus de réplication logique a été perturbé, vous pouvez rétablir la réplication en suivant les étapes suivantes.

Pour rétablir la réplication logique entre les nœuds de serveur de publication et abonné.

Pour rétablir la réplication, vous devez d'abord déconnecter l'abonné du nœud de serveur de publication, puis rétablir l'abonnement, comme indiqué dans les étapes suivantes.

1. Connectez-vous au nœud abonné à l'aide de `psql`, comme suit.

```
psql --host=222222222222.aws-region.rds.amazonaws.com --port=5432 --
username=postgres --password --dbname=labdb
```

2. Désactivez l'abonnement en utilisant la fonction `pglogical.alter_subscription_disable`.

```
SELECT pglogical.alter_subscription_disable('docs_lab_subscription',true);
 alter_subscription_disable
----------------------------
 t
(1 row)
```

3. Obtenez l'identifiant du nœud de serveur de publication en interrogeant `pg_replication_origin`, comme suit.

```
SELECT * FROM pg_replication_origin;
 roident |               roname
---------+-------------------------------------
       1 | pgl_labdb_docs_labcb4fa94_docs_lab3de412c
(1 row)
```

4. Utilisez la réponse de l'étape précédente avec la commande `pg_replication_origin_create` pour attribuer l'identifiant qui peut être utilisé par l'abonnement lorsqu'il est rétabli.

```
SELECT pg_replication_origin_create('pgl_labdb_docs_labcb4fa94_docs_lab3de412c');
  pg_replication_origin_create
------------------------------
                             1
(1 row)
```

5. Activez l'abonnement en transmettant son nom avec un statut `true`, comme indiqué dans l'exemple suivant.

```
SELECT pglogical.alter_subscription_enable('docs_lab_subscription',true);
  alter_subscription_enable
---------------------------
 t
(1 row)
```

Vérifiez le statut du nœud. Son statut doit être `replicating`, tel qu'indiqué dans cet exemple.

```
SELECT subscription_name,status,slot_name
  FROM pglogical.show_subscription_status();
            subscription_name |   status    |              slot_name
------------------------------+-------------+-----------------------------------
 docs_lab_subscription         | replicating |
 pgl_labdb_docs_lab98f517b_docs_lab3de412c
(1 row)
```

Vérifiez le statut de l'emplacement de réplication de l'abonné sur le nœud de serveur de publication. La colonne `active` de l'emplacement doit retourner `t` (true), indiquant que la réplication a été rétablie.

```
SELECT slot_name,plugin,slot_type,active
  FROM pg_replication_slots;
                   slot_name              |      plugin      | slot_type | active
------------------------------------------+------------------+-----------+--------
 pgl_labdb_docs_lab98f517b_docs_lab3de412c | pglogical_output | logical   | t
(1 row)
```

Gestion des emplacements logiques de réplication pour Aurora PostgreSQL

Avant de pouvoir effectuer une mise à niveau de version majeure sur une instance de base de données RDS for PostgreSQL qui sert de nœud de serveur de publication dans un scénario de réplication logique, vous devez supprimer les emplacements de réplication sur l'instance. Le processus de pré-vérification des mises à niveau de versions majeures vous informe que la mise à niveau ne peut pas avoir lieu tant que les emplacements ne sont pas supprimés.

Pour identifier les emplacements de réplication qui ont été créés à l'aide de l'extension `pglogical`, connectez-vous à chaque base de données et obtenez le nom des nœuds. Lorsque vous interrogez le nœud abonné, vous obtenez à la fois le nœud de serveur de publication et le nœud abonné dans la sortie, comme le montre cet exemple.

```
SELECT * FROM pglogical.node;
node_id   |     node_name
------------+------------------
 2182738256 | docs_lab_target
 3410995529 | docs_lab_provider
(2 rows)
```

Vous pouvez obtenir les détails de l'abonnement avec la requête suivante.

```
SELECT sub_name,sub_slot_name,sub_target
  FROM pglogical.subscription;
 sub_name |         sub_slot_name          | sub_target
----------+--------------------------------+------------
  docs_lab_subscription     | pgl_labdb_docs_labcb4fa94_docs_lab3de412c | 2182738256
(1 row)
```

Vous pouvez maintenant supprimer l'abonnement, comme suit.

```
SELECT pglogical.drop_subscription(subscription_name := 'docs_lab_subscription');
 drop_subscription
-------------------
                 1
(1 row)
```

Après avoir supprimé l'abonnement, vous pouvez supprimer le nœud.

```
SELECT pglogical.drop_node(node_name := 'docs-lab-subscriber');
 drop_node
```

```
-----------
 t
(1 row)
```

Vous pouvez vérifier que le nœud n'existe plus, comme suit.

```
SELECT * FROM pglogical.node;
 node_id | node_name
---------+-----------
(0 rows)
```

Référence des paramètres de l'extension pglogical

Dans le tableau, vous pouvez trouver les paramètres associés à l'extension `pglogical`. Les paramètres tels que `pglogical.conflict_log_level` et `pglogical.conflict_resolution` sont utilisés pour gérer les conflits de mise à jour. Des conflits peuvent survenir lorsque des modifications sont apportées localement aux mêmes tables qui sont abonnées aux modifications du serveur de publication. Des conflits peuvent également se produire au cours de divers scénarios, tels que la réplication bidirectionnelle ou lorsque plusieurs abonnés se répliquent à partir du même serveur de publication. Pour plus d'informations, consultez PostgreSQL bi-directional replication using pglogical (Réplication bidirectionnelle PostgreSQL utilisant pglogical).

Paramètre	Description
pglogical.batch_inserts	Insertions de lots si possible Non défini par défaut. Remplacez par « 1 » pour activer, par « 0 » pour désactiver.
pglogical.conflict_log_level	Définit le niveau de journalisation à utiliser pour la journalisation des conflits résolus. Les valeurs de chaîne prises en charge sont debug5, debug4, debug3, debug2, debug1, info, notice, warning, error, log, fatal, panic.
pglogical.conflict_resolution	Définit la méthode à utiliser pour résoudre les conflits lorsque ceux-ci sont résolubles. Les valeurs de chaîne prises en charge sont error, apply_remote, keep_local, last_update_wins, first_update_wins.
pglogical.extra_connection_options	Options de connexion à ajouter à toutes les connexions de nœuds de pairs.

Paramètre	Description
pglogical.synchronous_commit	valeur de validation synchrone spécifique pglogical
pglogical.use_spi	Utilisez l'interface de programmation du serveur (SPI) au lieu de l'API de bas niveau pour appliquer les modifications. Définissez sur « 1 » pour activer, sur « 0 » pour désactiver. Pour plus d'informations sur SPI, consultez Server Programming Interface (Interface de programmation du serveur) dans la documentation PostgreSQL.

Utilisation des encapsuleurs de données externes pris en charge pour Amazon Aurora PostgreSQL

Un encapsuleur de données externes est un type d'extension spécifique qui permet d'accéder à des données externes. Par exemple, l'extension `oracle_fdw` permet à votre instance de base de données Aurora PostgreSQL de fonctionner avec des bases de données Oracle.

Vous trouverez ci-dessous des informations sur plusieurs encapsuleurs de données externes PostgreSQL pris en charge.

Rubriques

- Utilisation de l'extension log_fdw pour accéder au journal de base de données à l'aide de SQL
- Utilisation de l'extension postgres_fdw pour accéder à des données externes
- Travailler avec des bases de données MySQL en utilisant l'extension mysql_fdw
- Utilisation des bases de données Oracle avec l'extension oracle_fdw
- Utilisation de bases de données SQL Server avec l'extension tds_fdw

Utilisation de l'extension log_fdw pour accéder au journal de base de données à l'aide de SQL

Le cluster de base de données Aurora PostgreSQL prend en charge l'extension `log_fdw`, qui vous permet d'accéder au journal de votre moteur de base de données à l'aide d'une interface SQL. L'extension `log_fdw` fournit deux fonctions qui facilitent la création de tables source pour les journaux de base de données :

- `list_postgres_log_files` – Répertorie les fichiers dans le répertoire du journal de base de données et indique la taille des fichiers en octets.
- `create_foreign_table_for_log_file(table_name text, server_name text, log_file_name text)` – Crée un tableau source pour le fichier spécifié dans la base de données actuelle.

Toutes les fonctions créées par `log_fdw` appartiennent à `rds_superuser`. Les membres du rôle `rds_superuser` peuvent accorder l'accès à ces fonctions à d'autres utilisateurs de base de données.

Par défaut, les fichiers journaux sont générés par Amazon Aurora au format `stderr` (erreur standard), comme spécifié dans le paramètre `log_destination`. Il n'y a que deux options pour ce paramètre, `stderr` et `csvlog` (valeurs séparées par des virgules, CSV). Si vous ajoutez l'option `csvlog` au paramètre, Amazon Aurora génère les journaux `stderr` et `csvlog`. Cela peut affecter la capacité de stockage de votre cluster de base de données. Vous devez donc connaître les autres paramètres qui affectent la gestion des journaux. Pour de plus amples informations, consultez Définition de la destination du journal (`stderr`, `csvlog`).

L'un des avantages de la génération de journaux `csvlog` est que l'extension `log_fdw` vous permet de créer des tables externes dont les données sont soigneusement réparties en plusieurs colonnes. Pour ce faire, votre instance doit être associée à un groupe de paramètres de base de données personnalisé afin que vous puissiez modifier le paramètre de `log_destination`. Pour plus d'informations sur la manière de procéder, consultez Groupes de paramètres pour Amazon Aurora.

L'exemple suivant suppose que le paramètre `log_destination` comprend le champ `cvslog`.

Pour utiliser l'extension log_fdw

1. Installez l'extension `log_fdw`.

```
postgres=> CREATE EXTENSION log_fdw;
CREATE EXTENSION
```

2. Créez le serveur de journal en tant qu'encapsuleur de données externes.

```
postgres=> CREATE SERVER log_server FOREIGN DATA WRAPPER log_fdw;
CREATE SERVER
```

3. Sélectionnez l'ensemble des fichiers journaux d'une liste.

```
postgres=> SELECT * FROM list_postgres_log_files() ORDER BY 1;
```

Voici un exemple de réponse.

```
          file_name           | file_size_bytes
------------------------------+-----------------
 postgresql.log.2023-08-09-22.csv |            1111
 postgresql.log.2023-08-09-23.csv |            1172
 postgresql.log.2023-08-10-00.csv |            1744
 postgresql.log.2023-08-10-01.csv |            1102
(4 rows)
```

4. Créez une table avec une seule colonne « log_entry » pour le fichier sélectionné.

```
postgres=> SELECT create_foreign_table_for_log_file('my_postgres_error_log',
     'log_server', 'postgresql.log.2023-08-09-22.csv');
```

La réponse ne fournit aucun détail autre que l'existence de la table.

```
-----------------------------------
(1 row)
```

5. Sélectionnez un exemple de fichier journal. Le code suivant récupère l'heure du journal et la description du message d'erreur.

```
postgres=> SELECT log_time, message FROM my_postgres_error_log ORDER BY 1;
```

Voici un exemple de réponse.

```
            log_time             |                                  message
---------------------------------
+-------------------------------------------------------------------------------
Tue Aug 09 15:45:18.172 2023 PDT | ending log output to stderr
Tue Aug 09 15:45:18.175 2023 PDT | database system was interrupted; last known up
 at 2023-08-09 22:43:34 UTC
Tue Aug 09 15:45:18.223 2023 PDT | checkpoint record is at 0/90002E0
Tue Aug 09 15:45:18.223 2023 PDT | redo record is at 0/90002A8; shutdown FALSE
Tue Aug 09 15:45:18.223 2023 PDT | next transaction ID: 0/1879; next OID: 24578
Tue Aug 09 15:45:18.223 2023 PDT | next MultiXactId: 1; next MultiXactOffset: 0
```

```
Tue Aug 09 15:45:18.223 2023 PDT | oldest unfrozen transaction ID: 1822, in
 database 1
(7 rows)
```

Utilisation de l'extension postgres_fdw pour accéder à des données externes

Vous pouvez accéder aux données d'un tableau sur un serveur de bases de données distant à l'aide de l'extension postgres_fdw. Si vous configurez une connexion distante à partir de votre instance de base de données PostgreSQL, l'accès à votre réplica en lecture est également disponible.

Pour utiliser postgres_fdw pour accéder à un serveur de bases de données distant

1. Installez l'extension postgres_fdw.

```
CREATE EXTENSION postgres_fdw;
```

2. Créez un serveur de données externes à l'aide de CREATE SERVER.

```
CREATE SERVER foreign_server
FOREIGN DATA WRAPPER postgres_fdw
OPTIONS (host 'xxx.xx.xxx.xx', port '5432', dbname 'foreign_db');
```

3. Créez un mappage utilisateur pour identifier le rôle à utiliser sur le serveur distant.

> **Important**
>
> Pour éviter que le mot de passe ne soit consigné dans les journaux, définissez le paramètre `log_statement=none` au niveau de la session. Définir ce réglage au niveau du paramètre ne permet pas de masquer le mot de passe.

```
CREATE USER MAPPING FOR local_user
SERVER foreign_server
OPTIONS (user 'foreign_user', password 'password');
```

4. Créez une table mappée à la table sur le serveur distant.

```
CREATE FOREIGN TABLE foreign_table (
        id integer NOT NULL,
        data text)
```

```
SERVER foreign_server
OPTIONS (schema_name 'some_schema', table_name 'some_table');
```

Travailler avec des bases de données MySQL en utilisant l'extension mysql_fdw

Pour accéder à une base de données compatible MySQL à partir de votre cluster de base de données Aurora PostgreSQL, vous pouvez installer et utiliser l'extension `mysql_fdw`. Cet encapsuleur de données externes vous permet de travailler avec RDS for MySQL, Aurora MySQL, MariaDB et d'autres bases de données compatibles avec MySQL. La connexion de votre cluster de base de données Aurora PostgreSQL à la base de données MySQL est chiffrée au mieux, en fonction des configurations du client et du serveur. Cependant, vous pouvez imposer le chiffrement si vous le souhaitez. Pour de plus amples informations, consultez Utilisation du chiffrement en transit avec l'extension.

L'extension `mysql_fdw` est prise en charge par Amazon Aurora PostgreSQL versions 15.4, 14.9, 13.12 et 12.16, et ultérieures. Elle prend en charge la sélection, l'insertion, la mise à jour et la suppression d'une base de données RDS for PostgreSQL vers des tables sur une instance de base de données compatible MySQL.

Rubriques

- Configuration de votre base de données Aurora PostgreSQL pour utiliser l'extension mysql_fdw
- Exemple : utilisation d'une base de données Aurora MySQL à partir d'Aurora PostgreSQL
- Utilisation du chiffrement en transit avec l'extension

Configuration de votre base de données Aurora PostgreSQL pour utiliser l'extension mysql_fdw

La configuration de l'extension `mysql_fdw` sur votre cluster de base de données Aurora PostgreSQL implique le chargement de l'extension dans votre cluster de base de données, puis la création du point de connexion à l'instance de base de données MySQL. Pour cette tâche, vous devez disposer des informations suivantes sur l'instance de base de données MySQL :

- Nom d'hôte ou point de terminaison. Pour un cluster de base de données Aurora MySQL, vous pouvez trouver le point de terminaison à l'aide de la console. Sélectionnez l'onglet Connectivité et sécurité et regardez dans la section « Point de terminaison et port ».
- Numéro de port. Le numéro de port par défaut pour MySQL est 3306.
- Nom du moteur de la base de données. L'identifiant de la base de données.

Vous devez également fournir un accès sur le groupe de sécurité ou la liste de contrôle d'accès (ACL) pour le port MySQL 3306. Les clusters de bases de données Aurora PostgreSQL et Aurora MySQL doivent avoir accès au port 3306. Si l'accès n'est pas configuré correctement, lorsque vous essayez de vous connecter à une table compatible avec MySQL, vous voyez apparaître un message d'erreur similaire au suivant :

```
ERROR: failed to connect to MySQL: Can't connect to MySQL server on 'hostname.aws-
region.rds.amazonaws.com:3306' (110)
```

Dans la procédure suivante, vous (en tant que compte `rds_superuser`) créez le serveur externe. Vous accordez ensuite l'accès au serveur externe à des utilisateurs spécifiques. Ces utilisateurs créent ensuite leurs propres mappages vers les comptes utilisateurs MySQL appropriés pour travailler avec l'instance de base de données MySQL.

Pour utiliser mysql_fdw pour accéder à un serveur de base de données MySQL

1. Connectez-vous à votre instance de base de données PostgreSQL en utilisant un compte qui a le rôle `rds_superuser`. Si vous avez accepté les valeurs par défaut lors de la création de votre cluster de base de données Aurora PostgreSQL, le nom d'utilisateur est `postgres`, et vous pouvez vous connecter à l'aide de l'outil de ligne de commande `psql` comme suit :

   ```
   psql --host=your-DB-instance.aws-region.rds.amazonaws.com --port=5432 --
   username=postgres --password
   ```

2. Installez l'extension `mysql_fdw` comme suit :

   ```
   postgres=> CREATE EXTENSION mysql_fdw;
   CREATE EXTENSION
   ```

Une fois l'extension installée sur votre cluster de base de données Aurora PostgreSQL, vous devez configurer le serveur externe qui fournit la connexion à une base de données MySQL.

Pour créer le serveur externe

Effectuez ces tâches sur le cluster de base de données Aurora PostgreSQL. Les étapes supposent que vous êtes connecté en tant qu'utilisateur avec des privilèges `rds_superuser`, tels que `postgres`.

1. Créer un serveur externe dans le cluster de base de données Aurora PostgreSQL :

```
postgres=> CREATE SERVER mysql-db FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'db-
name.111122223333.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. Accordez aux utilisateurs appropriés l'accès au serveur externe. Il doit s'agir d'utilisateurs non administrateurs, c'est-à-dire d'utilisateurs sans rôle `rds_superuser`.

```
postgres=> GRANT USAGE ON FOREIGN SERVER mysql-db to user1;
GRANT
```

Les utilisateurs de PostgreSQL créent et gèrent leurs propres connexions à la base de données MySQL via le serveur externe.

Exemple : utilisation d'une base de données Aurora MySQL à partir d'Aurora PostgreSQL

Supposons que vous ayez une table simple sur une instance de base de données Aurora PostgreSQL. Vos utilisateurs Aurora PostgreSQL souhaitent interroger les éléments (`SELECT`), `INSERT`, `UPDATE` et `DELETE` de cette table. Supposons que l'extension `mysql_fdw` a été créée sur votre instance de base de données RDS for PostgreSQL, comme indiqué dans la procédure précédente. Après vous être connecté à l'instance de base de données RDS for PostgreSQL en tant qu'utilisateur disposant de privilèges `rds_superuser`, vous pouvez procéder aux étapes suivantes.

1. Créez un serveur externe sur l'instance de base de données Aurora PostgreSQL :

```
test=> CREATE SERVER mysqldb FOREIGN DATA WRAPPER mysql_fdw OPTIONS (host 'your-
DB.aws-region.rds.amazonaws.com', port '3306');
CREATE SERVER
```

2. Accordez l'utilisation à un utilisateur dépourvu d'autorisations `rds_superuser`, par exemple `user1` :

```
test=> GRANT USAGE ON FOREIGN SERVER mysqldb TO user1;
GRANT
```

3. Connectez-vous en tant que *`user1`*, puis créez un mappage vers l'utilisateur MySQL :

```
test=> CREATE USER MAPPING FOR user1 SERVER mysqldb OPTIONS (username 'myuser',
 password 'mypassword');
CREATE USER MAPPING
```

4. Créez une table externe liée à la table MySQL :

```
test=> CREATE FOREIGN TABLE mytab (a int, b text) SERVER mysqldb OPTIONS (dbname
 'test', table_name '');
CREATE FOREIGN TABLE
```

5. Exécutez une requête simple dans la table externe :

```
test=> SELECT * FROM mytab;
a |   b
---+-------
1 | apple
(1 row)
```

6. Vous pouvez ajouter, modifier et supprimer des données de la table MySQL. Par exemple :

```
test=> INSERT INTO mytab values (2, 'mango');
INSERT 0 1
```

   Exécutez à nouveau la requête SELECT pour voir les résultats :

```
test=> SELECT * FROM mytab ORDER BY 1;
 a |   b
---+-------
1 | apple
2 | mango
(2 rows)
```

Utilisation du chiffrement en transit avec l'extension

La connexion à MySQL à partir d'Aurora PostgreSQL utilise le chiffrement en transit (TLS/SSL) par défaut. Toutefois, la connexion redevient non chiffrée lorsque la configuration du client et du serveur diffère. Vous pouvez imposer le chiffrement pour toutes les connexions sortantes en spécifiant l'option `REQUIRE SSL` sur les comptes d'utilisateur RDS for MySQL. Cette même approche fonctionne également pour les comptes utilisateurs MariaDB et Aurora MySQL.

Pour les comptes utilisateurs MySQL configurés pour `REQUIRE SSL`, la tentative de connexion échoue si une connexion sécurisée ne peut être établie.

Pour appliquer le chiffrement aux comptes d'utilisateurs de bases de données MySQL existants, vous pouvez utiliser la commande `ALTER USER`. La syntaxe varie en fonction de la version MySQL, comme indiqué dans le tableau suivant. Pour plus d'informations, consultez ALTER USER dans le Manuel de référence de MySQL.

MySQL 5.7, MySQL 8.0	MySQL 5.6
`ALTER USER 'user'@'%' REQUIRE SSL;`	`GRANT USAGE ON *.* to 'user'@'%' REQUIRE SSL;`

Pour plus d'informations sur l'extension `mysql_fdw`, consultez la documentation mysql_fdw.

Utilisation des bases de données Oracle avec l'extension oracle_fdw

Pour accéder à une base de données Oracle depuis votre cluster de bases de données Aurora PostgreSQL , vous pouvez installer et utiliser l'extension `oracle_fdw`. Cette extension est un encapsuleur de données externes pour les bases de données Oracle. Pour en savoir plus sur cette extension, consultez la documentation oracle_fdw.

L'extension `oracle_fdw` est prise en charge sur Aurora PostgreSQL 12.7 (Amazon Aurora version 4.2) et les versions ultérieures.

Rubriques

- Activation de l'extension oracle_fdw
- Exemple : utilisation d'un serveur externe lié à une base de données Amazon RDS for Oracle Database
- Utilisation du chiffrement en transit
- Comprendre la vue et les autorisations pg_user_mappings

Activation de l'extension oracle_fdw

Pour utiliser l'extension oracle_fdw, suivez la procédure suivante.

Pour activer l'extension oracle_fdw

- Exécutez la commande suivante en utilisant un compte disposant d'autorisations `rds_superuser`.

```
CREATE EXTENSION oracle_fdw;
```

Exemple : utilisation d'un serveur externe lié à une base de données Amazon RDS for Oracle Database

Les exemples suivants démontrent l'utilisation d'un serveur externe lié à une base de données Amazon RDS for Oracle.

Pour créer un serveur externe lié à une base de données RDS for Oracle

1. Notez ce qui suit sur l'instance de base de données RDS for Oracle :
   - Point de terminaison
   - Port
   - Nom de base de données
2. Créez un serveur externe.

```
test=> CREATE SERVER oradb FOREIGN DATA WRAPPER oracle_fdw OPTIONS (dbserver
 '//endpoint:port/DB_name');
CREATE SERVER
```

3. Accordez l'utilisation à un utilisateur dépourvu d'autorisations `rds_superuser`, par exemple `user1`.

```
test=> GRANT USAGE ON FOREIGN SERVER oradb TO user1;
GRANT
```

4. Connectez-vous en tant que `user1` et créez un mappage à un utilisateur Oracle.

```
test=> CREATE USER MAPPING FOR user1 SERVER oradb OPTIONS (user 'oracleuser',
 password 'mypassword');
CREATE USER MAPPING
```

5. Créez une table externe liée à une table Oracle.

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER oradb OPTIONS (table 'MYTABLE');
CREATE FOREIGN TABLE
```

6. Interrogez la table externe.

```
test=>  SELECT * FROM mytab;
a
---
1
(1 row)
```

Si la requête signale l'erreur suivante, vérifiez votre groupe de sécurité et votre liste de contrôle d'accès (ACL) pour vous assurer que les deux instances peuvent communiquer.

```
ERROR: connection for foreign table "mytab" cannot be established
DETAIL: ORA-12170: TNS:Connect timeout occurred
```

Utilisation du chiffrement en transit

Le chiffrement PostgreSQL vers Oracle en transit est basé sur une combinaison de paramètres de configuration client et serveur. Pour un exemple d'utilisation d'Oracle 21c, consultez A propos de la négociation du chiffrement et de l'intégrité dans la documentation Oracle. Le client utilisé pour oracle_fdw sur Amazon RDS est configuré avec `ACCEPTED`, ce qui signifie que le chiffrement dépend de la configuration du serveur de base de données Oracle et qu'il utilise Oracle Security Library (libnnz) pour le chiffrement.

Si votre base de données se trouve sur RDS for Oracle, consultez la section Oracle native network encryption (Chiffrement réseau natif Oracle) pour configurer le chiffrement.

Comprendre la vue et les autorisations pg_user_mappings

Le catalogue PostgreSQL `pg_user_mapping` stocke le mappage d'un utilisateur Aurora PostgreSQL vers l'utilisateur d'un serveur de données externe (distant). L'accès au catalogue est restreint, mais vous utilisez la vue `pg_user_mappings` pour visualiser les mappages. Dans ce qui suit, vous trouverez un exemple qui présente comment les autorisations s'appliquent avec un exemple de base de données Oracle, mais ces informations s'appliquent plus généralement à tout encapsuleur de données externes.

Dans la sortie suivante, vous pouvez trouver des rôles et des autorisations mappés à trois exemples d'utilisateurs différents. Les utilisateurs `rdssu1` et `rdssu2` sont membres du rôle `rds_superuser`, et `user1` ne l'est pas. L'exemple utilise la métacommande `psql \du` pour lister les rôles existants.

```
test=>  \du
```

```
                                                          List of roles
    Role name    |                         Attributes                         |
                 Member of
-----------------+------------------------------------------------------------
+-----------------------------------------------------------
 rdssu1          |                                                            |
 {rds_superuser}
 rdssu2          |                                                            |
 {rds_superuser}
 user1           |                                                            | {}
```

Tous les utilisateurs, y compris ceux qui disposent de privilèges `rds_superuser`, sont autorisés à voir leurs propres mappages d'utilisateurs (`umoptions`) dans la table `pg_user_mappings`. Comme le montre l'exemple suivant, lorsque `rdssu1` tente d'obtenir tous les mappages d'utilisateurs, une erreur s'affiche en dépit des privilèges `rds_superuser` de `rdssu1` :

```
test=> SELECT * FROM pg_user_mapping;
ERROR: permission denied for table pg_user_mapping
```

Voici quelques exemples.

```
test=> SET SESSION AUTHORIZATION rdssu1;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     | {user=oracleuser,password=mypwd}
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)

test=> SET SESSION AUTHORIZATION rdssu2;
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |            umoptions
-------+-------+---------+--------+------------+---------------------------------
 16414 | 16411 | oradb   |  16412 | user1      |
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     | {user=oracleuser,password=mypwd}
 (3 rows)

test=> SET SESSION AUTHORIZATION user1;
```

```
SET
test=> SELECT * FROM pg_user_mappings;
 umid  | srvid | srvname | umuser | usename    |           umoptions
-------+-------+---------+--------+------------+-------------------------------
 16414 | 16411 | oradb   |  16412 | user1      | {user=oracleuser,password=mypwd}
 16423 | 16411 | oradb   |  16421 | rdssu1     |
 16424 | 16411 | oradb   |  16422 | rdssu2     |
 (3 rows)
```

En raison des différences dans l'implémentation de `information_schema._pg_user_mappings` et de `pg_catalog.pg_user_mappings`, un `rds_superuser` créé manuellement nécessite des autorisations supplémentaires pour afficher les mots de passe dans `pg_catalog.pg_user_mappings`.

Un `rds_superuser` n'a besoin d'aucune autorisation supplémentaire pour afficher les mots de passe dans `information_schema._pg_user_mappings`.

Les utilisateurs qui n'ont pas le rôle `rds_superuser` peuvent afficher les mots de passe dans `pg_user_mappings` uniquement dans les conditions suivantes :

- L'utilisateur actif est celui faisant l'objet du mappage. Il possède le serveur ou détient le privilège `USAGE` sur celui-ci.
- L'utilisateur actuel est le propriétaire du serveur et le mappage est pour `PUBLIC`.

Utilisation de bases de données SQL Server avec l'extension tds_fdw

Vous pouvez utiliser l'extension PostgreSQL `tds_fdw` pour accéder aux bases de données qui prennent en charge le protocole TDS (tabular data stream), comme les bases de données Sybase et Microsoft SQL Server. Cet encapsuleur de données externes vous permet de vous connecter à partir de votre ou de votre cluster de base de données PostgreSQL à des bases de données qui utilisent le protocole TDS, y compris Amazon RDS for Microsoft SQL Server. Pour plus d'informations, consultez la documentation de tds-fdw/tds_fdw sur GitHub.

L'extension `tds_fdw` est prise en charge sur Amazon Aurora PostgreSQL version 13.6 et versions ultérieures.

Configuration de votre base de données Aurora PostgreSQL pour utiliser l'extension tds_fdw

Dans les procédures suivantes, vous trouverez un exemple de configuration et d'utilisation de `tds_fdw` avec un cluster de base de données Aurora PostgreSQL. Avant de pouvoir vous connecter

à une base de données SQL Server à l'aide de `tds_fdw`, vous devez obtenir les détails suivants pour l'instance :

- Nom d'hôte ou point de terminaison. Pour une instance de base de données RDS for SQL Server, vous pouvez trouver le point de terminaison en utilisant la console. Sélectionnez l'onglet Connectivité et sécurité et regardez dans la section « Point de terminaison et port ».
- Numéro de port. Le numéro de port par défaut de Microsoft SQL Server est 1433.
- Nom du moteur de la base de données. L'identifiant de la base de données.

Vous devez également fournir un accès au groupe de sécurité ou à la liste de contrôle d'accès (ACL) pour le port du serveur SQL 1433. Le cluster de base de données Aurora PostgreSQL et l'instance de base de données RDS for SQL Server ont tous deux besoin d'accéder au port 1433. Si l'accès n'est pas configuré correctement, lorsque vous essayez d'interroger le serveur Microsoft SQL, le message d'erreur suivant s'affiche :

```
ERROR: DB-Library error: DB #: 20009, DB Msg: Unable to connect:
Adaptive Server is unavailable or does not exist (mssql2019.aws-
region.rds.amazonaws.com), OS #: 0, OS Msg: Success, Level: 9
```

Pour utiliser tds_fdw pour vous connecter à une base de données SQL Server

1. Connectez-vous à votre instance principale du cluster de base de données Aurora PostgreSQL en utilisant un compte qui dispose du rôle `rds_superuser` :

   ```
   psql --host=your-cluster-name-instance-1.aws-region.rds.amazonaws.com --port=5432
    --username=test --password
   ```

2. Installez l'extension `tds_fdw` :

   ```
   test=> CREATE EXTENSION tds_fdw;
   CREATE EXTENSION
   ```

Une fois l'extension installée sur votre cluster de base de données Aurora PostgreSQL , vous configurez le serveur externe.

Pour créer le serveur externe

Effectuez ces tâches sur le cluster de base de données Aurora PostgreSQL en utilisant un compte qui dispose de privilèges `rds_superuser`.

1. Créer un serveur externe dans le cluster de base de données Aurora PostgreSQL :

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS
 (servername 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database
 'tds_fdw_testing');
CREATE SERVER
```

   Pour accéder à des données non-ASCII côté SQL Server, créez un lien vers le serveur avec l'option character_set dans le cluster de base de données Aurora PostgreSQL :

```
test=> CREATE SERVER sqlserverdb FOREIGN DATA WRAPPER tds_fdw OPTIONS (servername
 'mssql2019.aws-region.rds.amazonaws.com', port '1433', database 'tds_fdw_testing',
 character_set 'UTF-8');
CREATE SERVER
```

2. Accordez des autorisations à un utilisateur qui n'a pas de privilèges de rôle `rds_superuser`, par exemple, `user1` :

```
test=> GRANT USAGE ON FOREIGN SERVER sqlserverdb TO user1;
```

3. Connectez-vous en tant que user1 et créez un mappage vers un utilisateur SQL Server :

```
test=> CREATE USER MAPPING FOR user1 SERVER sqlserverdb OPTIONS (username
 'sqlserveruser', password 'password');
CREATE USER MAPPING
```

4. Créez une table externe liée à une table SQL Server :

```
test=> CREATE FOREIGN TABLE mytab (a int) SERVER sqlserverdb OPTIONS (table
 'MYTABLE');
CREATE FOREIGN TABLE
```

5. Interrogez la table externe :

```
test=> SELECT * FROM mytab;
 a
```

```
---
 1
(1 row)
```

Utilisation du chiffrement en transit pour la connexion

La connexion d'Aurora PostgreSQL à SQL Server utilise le chiffrement en transit (TLS/SSL) selon la configuration de la base de données SQL Server. Si le serveur SQL n'est pas configuré pour le chiffrement, le client RDS for PostgreSQL qui émet la requête à la base de données du serveur SQL revient au mode non chiffré.

Vous pouvez renforcer le chiffrement de la connexion aux instances de base de données RDS for SQL Server en définissant le paramètre `rds.force_ssl`. Pour savoir comment procéder, consultez Forcer les connexions à votre instance de base de données pour utiliser SSL. Pour plus d'informations sur la configuration SSL/TLS pour RDS for SQL Server, consultez Utilisation de SSL avec une instance DB Microsoft SQL Server.

Utilisation de Trusted Language Extensions pour PostgreSQL

Trusted Language Extensions pour PostgreSQL est un kit de développement open source permettant de créer des extensions PostgreSQL. Il vous permet de créer des extensions PostgreSQL à hautes performances et de les exécuter en toute sécurité sur votre cluster de bases de données Aurora PostgreSQL. En utilisant Trusted Language Extensions (TLE) pour PostgreSQL, vous pouvez créer des extensions PostgreSQL qui suivent l'approche documentée pour étendre les fonctionnalités de PostgreSQL. Pour plus d'informations, consultez Packaging Related Objects into an Extension (Empaquetage d'objets associés dans une extension) dans la documentation PostgreSQL.

L'un des principaux avantages de TLE est que vous pouvez l'utiliser dans des environnements qui ne donnent pas accès au système de fichiers sous-jacent à l'instance PostgreSQL. Auparavant, l'installation d'une nouvelle extension nécessitait l'accès au système de fichiers. TLE supprime cette contrainte. Il fournit un environnement de développement permettant de créer de nouvelles extensions pour n'importe quelle base de données PostgreSQL, y compris celles qui s'exécutent sur vos clusters de bases de données Aurora PostgreSQL.

TLE est conçu pour empêcher l'accès à des ressources dangereuses pour les extensions que vous créez à l'aide de TLE. Son environnement d'exécution limite l'impact de tout défaut d'extension à une seule connexion de base de données. TLE permet également aux administrateurs de base de données de contrôler précisément qui peut installer les extensions et fournit un modèle d'autorisations pour les exécuter.

TLE est pris en charge sur Aurora PostgreSQL version 14.5 et ultérieures.

Le runtime et l'environnement de développement Trusted Language Extensions sont fournis sous la forme de l'extension PostgreSQL `pg_tle`, version 1.0.1. Elle prend en charge la création d'extensions dans les langages JavaScript, Perl, Tcl, PL/pgSQL et SQL. Vous installez l'extension `pg_tle` dans votre cluster de bases de données Aurora PostgreSQL de la même manière que vous installez les autres extensions PostgreSQL. Une fois le kit `pg_tle` configuré, les développeurs peuvent l'utiliser pour créer de nouvelles extensions PostgreSQL, appelées extensions TLE.

Dans les rubriques suivantes, vous apprendrez comment configurer le kit Trusted Language Extensions et comment commencer à créer vos propres extensions TLE.

Rubriques

- Exigences relatives à l’utilisation de Trusted Language Extensions pour PostgreSQL
- Configuration de Trusted Language Extensions dans votre cluster de bases de données Aurora PostgreSQL
- Présentation de Trusted Language Extensions pour PostgreSQL
- Création d'extensions TLE pour Aurora PostgreSQL
- Suppression de vos extensions TLE d'une base de données
- Désinstallation de Trusted Language Extensions pour PostgreSQL
- Utilisation des hooks PostgreSQL avec vos extensions TLE
- Référence de fonction pour Trusted Language Extensions pour PostgreSQL
- Référence des hooks pour Trusted Language Extensions pour PostgreSQL

Terminologie

Pour vous aider à mieux comprendre Trusted Language Extensions, consultez le glossaire suivant des termes utilisés dans cette rubrique.

Trusted Language Extensions pour PostgreSQL

: Trusted Language Extensions pour PostgreSQL est le nom officiel du kit de développement open source fourni en tant qu'extension `pg_tle`. Il peut être utilisé sur n'importe quel système PostgreSQL. Pour plus d'informations, consultez aws/pg_tle sur GitHub.

Trusted Language Extensions

: Trusted Language Extensions est le nom court de Trusted Language Extensions pour PostgreSQL. Ce nom court et son abréviation (TLE) sont également utilisés dans cette documentation.

langage approuvé

: Un langage approuvé est un langage de programmation ou de script doté d'attributs de sécurité spécifiques. Par exemple, les langages approuvés limitent généralement l'accès au système de fichiers et limitent l'utilisation de propriétés réseau spécifiées. Le kit de développement TLE est conçu pour prendre en charge les langages approuvés. PostgreSQL prend en charge plusieurs langages utilisés pour créer des extensions approuvées ou non approuvées. Pour voir un exemple, consultez Trusted and Untrusted PL/Perl (Langage PL/Perl approuvé et non approuvé) dans la documentation PostgreSQL. Lorsque vous créez une extension à l'aide du kit

Trusted Language Extensions, l'extension utilise intrinsèquement des mécanismes linguistiques approuvés.

extension TLE

Une extension TLE est une extension PostgreSQL créée à l'aide du kit de développement Trusted Language Extensions (TLE).

Exigences relatives à l’utilisation de Trusted Language Extensions pour PostgreSQL

Vous trouverez ci-dessous les exigences relatives à la configuration et à l’utilisation du kit de développement TLE.

- Versions d’Aurora PostgreSQL – Trusted Language Extensions est pris en charge sur Aurora PostgreSQL version 14.5 et versions ultérieures uniquement.
  - Si vous devez mettre à niveau votre cluster de bases de données Aurora PostgreSQL, consultez Mise à niveau des clusters de base de données Amazon Aurora PostgreSQL.
  - Si vous ne possédez pas encore de cluster de bases de données Aurora exécutant PostgreSQL, vous pouvez en créer un(e). Pour plus d’informations, consultez Création et connexion à un cluster de bases de données Aurora PostgreSQL.
- Nécessite les privilèges **`rds_superuser`** – Pour installer et configurer l’extension `pg_tle`, votre rôle d’utilisateur de base de données doit disposer des autorisations du rôle `rds_superuser`. Par défaut, ce rôle est accordé à l’utilisateur `postgres` qui crée le cluster de bases de données Aurora PostgreSQL.
- Nécessite un groupe de paramètres de base de données personnalisé : votre cluster de bases de données Aurora PostgreSQL doit être configuré avec un groupe de paramètres de base de données personnalisé. Vous utilisez le groupe de paramètres de base de données personnalisé pour l’instance d’enregistreur de votre cluster de bases de données Aurora PostgreSQL.
  - Si votre cluster de bases de données Aurora PostgreSQL n’est pas configuré avec un groupe de paramètres de base de données personnalisé, vous devez en créer un(e) et l’associer à l’instance d’enregistreur de votre cluster de bases de données Aurora PostgreSQL. Pour un bref résumé des étapes, consultez Création et application d’un groupe de paramètres de base de données personnalisé.
  - Si votre cluster de bases de données Aurora PostgreSQL est déjà configuré à l’aide d’un groupe de paramètres de base de données personnalisé, vous pouvez configurer Trusted Language

Extensions. Pour en savoir plus, consultez Configuration de Trusted Language Extensions dans votre cluster de bases de données Aurora PostgreSQL.

Création et application d'un groupe de paramètres de base de données personnalisé

Utilisez les étapes suivantes pour créer un groupe de paramètres de base de données personnalisé et configurer votre cluster de bases de données Aurora PostgreSQL afin de l'utiliser.

Console

Pour créer un groupe de paramètres de base de données personnalisé et l'utiliser avec votre cluster de bases de données Aurora PostgreSQL

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Parameter groups (Groupes de paramètres) dans le menu Amazon RDS.
3. Choisissez Créer un groupe de paramètres.
4. Dans la page Parameter group details (Détails des groupes de paramètres), entrez les informations suivantes.
   - Pour Parameter group family (Famille de groupes de paramètres), choisissez aurora-postgresql14.
   - Pour Type, choisissez DB Parameter Group (Groupe de paramètres de base de données).
   - Pour Group name (Nom du groupe), attribuez un nom significatif à votre groupe de paramètres dans le contexte de vos opérations.
   - Pour Description, entrez une description utile afin que les autres membres de votre équipe puissent la trouver facilement.
5. Choisissez Créer. Votre groupe de paramètres de base de données personnalisé est créé dans votre Région AWS. Vous pouvez désormais modifier votre cluster de bases de données Aurora PostgreSQL afin de l'utiliser dans les étapes suivantes.
6. Choisissez Databases (Bases de données) dans le menu Amazon RDS.
7. Choisissez le cluster de bases de données Aurora PostgreSQL que vous souhaitez utiliser avec TLE parmi les éléments répertoriés, puis choisissez Modify (Modifier).
8. Dans la page Modify DB cluster settings (Modifier les paramètres du cluster de bases de données), recherchez Database options (Options de base de données) et utilisez le sélecteur pour choisir votre groupe de paramètres de base de données personnalisé.

9. Choisissez Continue (Continuer) pour enregistrer la modification.
10. Choisissez Apply immediately (Appliquer immédiatement) afin de continuer à configurer le cluster de bases de données Aurora PostgreSQL pour utiliser TLE.

Pour continuer à configurer votre système pour Trusted Language Extensions, consultez Configuration de Trusted Language Extensions dans votre cluster de bases de données Aurora PostgreSQL.

Pour plus d'informations sur l'utilisation de groupes de paramètres de base de données et de cluster de bases de données, consultez Groupes de paramètres de cluster de bases de données pour les clusters de bases de données Amazon Aurora.

AWS CLI

Vous pouvez éviter de spécifier l'argument `--region` lorsque vous utilisez des commandes CLI en configurant votre AWS CLI avec votre Région AWS par défaut. Pour plus d'informations, consultez Principes de base de la configuration dans le guide de l'utilisateur AWS Command Line Interface.

Pour créer un groupe de paramètres de base de données personnalisé et l'utiliser avec votre cluster de bases de données Aurora PostgreSQL

1. Utilisez la commande AWS CLI create-db-parameter-group pour créer un groupe de paramètres de base de données personnalisé basé sur aurora-postgresql14 pour votre Région AWS. Notez que dans cette étape vous créez un groupe de paramètres de base de données à appliquer à l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-db-parameter-group \
     --region aws-region \
     --db-parameter-group-name custom-params-for-pg-tle \
     --db-parameter-group-family aurora-postgresql14 \
     --description "My custom DB parameter group for Trusted Language Extensions"
   ```

   Pour Windows :

   ```
   aws rds create-db-parameter-group ^
     --region aws-region ^
     --db-parameter-group-name custom-params-for-pg-tle ^
     --db-parameter-group-family aurora-postgresql14 ^
   ```

```
  --description "My custom DB parameter group for Trusted Language Extensions"
```

Votre groupe de paramètres de base de données personnalisé est disponible dans votre Région AWS. Vous pouvez donc modifier l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL afin de l'utiliser.

2. Utilisez la commande AWS CLI modify-db-instance pour appliquer votre groupe de paramètres de base de données personnalisé à l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL. Cette commande redémarre immédiatement l'instance active.

   Pour Linux, macOS ou Unix :

   ```
   aws rds modify-db-instance \
     --region aws-region \
     --db-instance-identifier your-writer-instance-name \
     --db-parameter-group-name custom-params-for-pg-tle \
     --apply-immediately
   ```

   Pour Windows :

   ```
   aws rds modify-db-instance ^
     --region aws-region ^
     --db-instance-identifier your-writer-instance-name ^
     --db-parameter-group-name custom-params-for-pg-tle ^
     --apply-immediately
   ```

Pour continuer à configurer votre système pour Trusted Language Extensions, consultez Configuration de Trusted Language Extensions dans votre cluster de bases de données Aurora PostgreSQL.

Pour plus d'informations, consultez Groupes de paramètres de base de données pour les instances de base de données Amazon Aurora.

Configuration de Trusted Language Extensions dans votre cluster de bases de données Aurora PostgreSQL

Les étapes suivantes supposent que votre cluster de bases de données Aurora PostgreSQL est associé à un groupe de paramètres de cluster de bases de données personnalisé. Vous pouvez utiliser la AWS Management Console ou AWS CLI pour effectuer ces étapes.

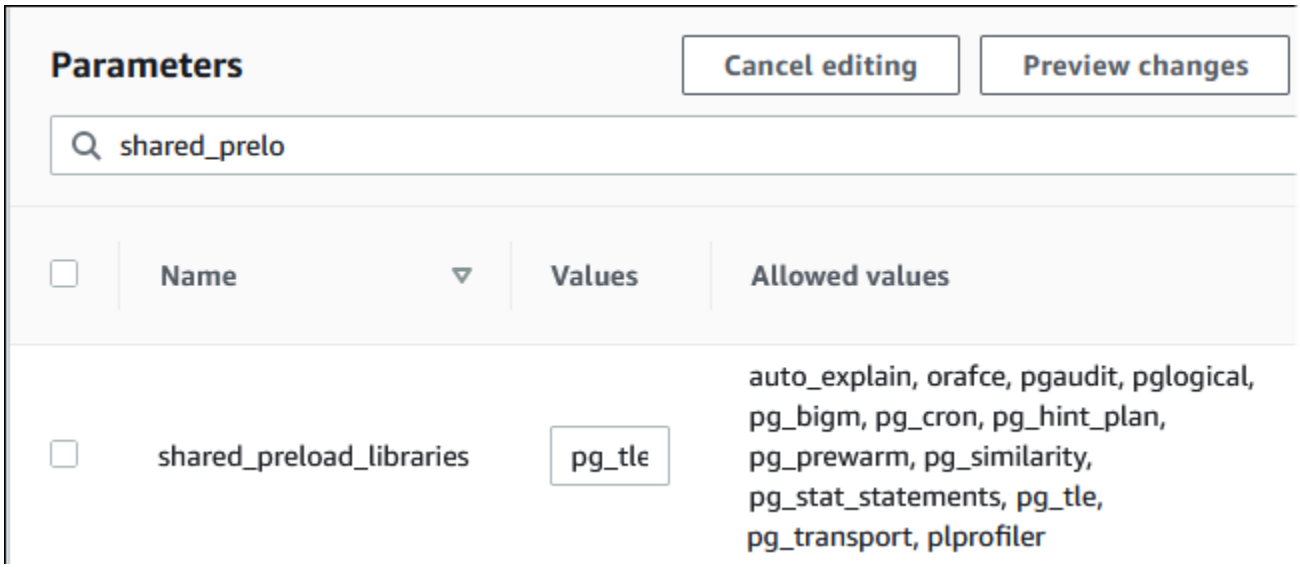
Lorsque vous configurez Trusted Language Extensions dans votre cluster de bases de données Aurora PostgreSQL , vous l'installez dans une base de données spécifique à l'usage des utilisateurs de base de données autorisés sur cette base de données.

console

Pour configurer Trusted Language Extensions

Effectuez les étapes suivantes à l'aide d'un compte membre du groupe (rôle) `rds_superuser`.

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le volet de navigation, choisissez votre instance d'écriture du cluster de bases de données Aurora PostgreSQL .
3. Ouvrez l'onglet Configuration pour votre instance d'écriture du cluster de bases de données Aurora PostgreSQL. Parmi les détails de l'instance, trouvez le lien Groupe de paramètres.
4. Cliquez sur le lien pour ouvrir les paramètres personnalisés associés à votre cluster de bases de données Aurora PostgreSQL.
5. Dans le champ de recherche Parameters (Paramètres), tapez `shared_pre` pour trouver le paramètre `shared_preload_libraries`.
6. Choisissez Edit parameters (Modifier les paramètres) pour accéder aux valeurs des propriétés.
7. Ajoutez `pg_tle` à la liste dans le champ Values (Valeurs). Utilisez une virgule pour séparer les éléments de la liste de valeurs.



8. Redémarrez l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL afin que vos modifications du paramètre `shared_preload_libraries` prennent effet.
9. Lorsque l'instance est disponible, vérifiez que `pg_tle` a été initialisé. Utilisez `psql` pour vous connecter à l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL, puis exécutez la commande suivante.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

10. Une fois l'extension `pg_tle` initialisée, vous pouvez maintenant créer l'extension.

```
CREATE EXTENSION pg_tle;
```

Vous pouvez vérifier que l'extension est installée en utilisant la métacommande `psql` suivante.

```
labdb=> \dx
                          List of installed extensions
  Name   | Version |   Schema   |                Description
---------+---------+------------+-------------------------------------------
 pg_tle  | 1.0.1   | pgtle      | Trusted-Language Extensions for PostgreSQL
 plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
```

11. Accordez le rôle `pgtle_admin` au nom d'utilisateur principal que vous avez créé pour votre cluster de bases de données Aurora PostgreSQL lors de sa configuration. Si vous avez accepté la valeur par défaut, il s'agit de `postgres`.

```
labdb=> GRANT pgtle_admin TO postgres;
GRANT ROLE
```

Vous pouvez vérifier que l'octroi a eu lieu à l'aide de la métacommande `psql`, comme illustré dans l'exemple suivant. Seuls les rôles `pgtle_admin` et `postgres` sont affichés dans la sortie. Pour plus d'informations, consultez Comprendre les rôles et les autorisations PostgreSQL.

```
labdb=> \du
                          List of roles
    Role name    |           Attributes            |              Member of
```

```
----------------+----------------------------------
+----------------------------------
pgtle_admin     | Cannot login                      | {}
postgres        | Create role, Create DB          +| {rds_superuser,pgtle_admin}
                | Password valid until infinity    |...
```

12. Fermez la session `psql` à l'aide de la métacommande `\q`.

```
\q
```

Pour commencer à créer des extensions TLE, consultez Exemple : création d'une extension de langage approuvé utilisant SQL.

AWS CLI

Vous pouvez éviter de spécifier l'argument `--region` lorsque vous utilisez des commandes CLI en configurant votre AWS CLI avec votre Région AWS par défaut. Pour plus d'informations, consultez Configuration basics (Principes de base de la configuration) dans le guide de l'utilisateur AWS Command Line Interface.

Pour configurer Trusted Language Extensions

1. Utilisez la commande AWS CLI modify-db-parameter-group pour ajouter `pg_tle` au paramètre `shared_preload_libraries`.

```
aws rds modify-db-parameter-group \
   --db-parameter-group-name custom-param-group-name \
   --parameters
 "ParameterName=shared_preload_libraries,ParameterValue=pg_tle,ApplyMethod=pending-
reboot" \
   --region aws-region
```

2. Utilisez la commande AWS CLI reboot-db-instance pour redémarrer l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL et initialiser la bibliothèque `pg_tle`.

```
aws rds reboot-db-instance \
    --db-instance-identifier writer-instance \
    --region aws-region
```

3. Lorsque l'instance est disponible, vous pouvez vérifier que `pg_tle` a été initialisé. Utilisez `psql` pour vous connecter à l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL, puis exécutez la commande suivante.

```
SHOW shared_preload_libraries;
shared_preload_libraries
--------------------------
rdsutils,pg_tle
(1 row)
```

   Une fois `pg_tle` initialisé, vous pouvez maintenant créer l'extension.

```
CREATE EXTENSION pg_tle;
```

4. Accordez le rôle `pgtle_admin` au nom d'utilisateur principal que vous avez créé pour votre cluster de bases de données Aurora PostgreSQL lors de sa configuration. Si vous avez accepté la valeur par défaut, il s'agit de `postgres`.

```
GRANT pgtle_admin TO postgres;
GRANT ROLE
```

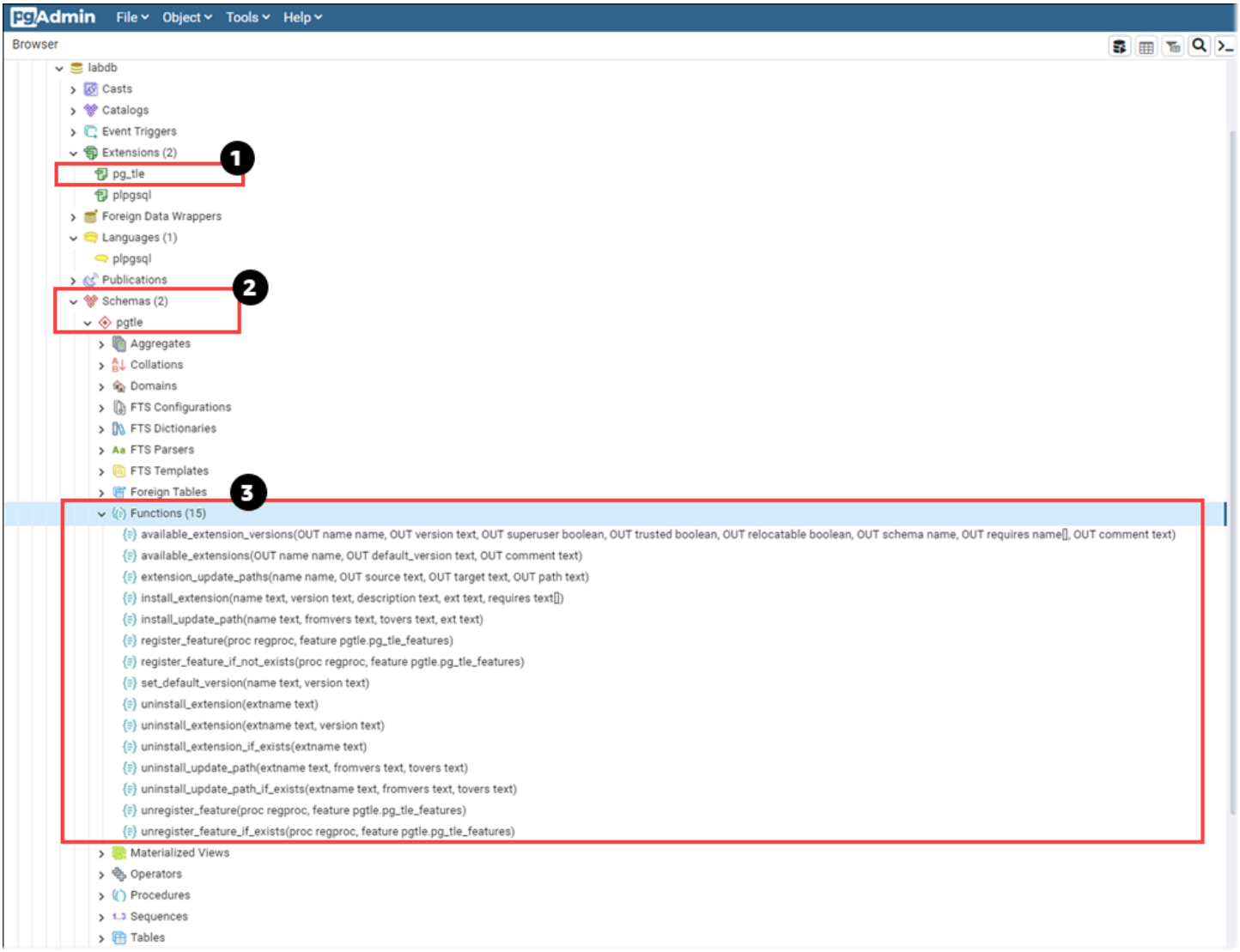
5. Fermez la session `psql` comme suit.

```
labdb=> \q
```

Pour commencer à créer des extensions TLE, consultez Exemple : création d'une extension de langage approuvé utilisant SQL.

Présentation de Trusted Language Extensions pour PostgreSQL

Trusted Language Extensions pour PostgreSQL est une extension PostgreSQL que vous installez dans votre cluster de bases de données Aurora PostgreSQL de la même manière que vous configurez les autres extensions PostgreSQL. Dans l'image suivante d'un exemple de base de données dans l'outil client pgAdmin, vous pouvez voir certains des composants incluant l'extension `pg_tle`.



Vous pouvez voir les détails suivants.

1. Le kit de développement Trusted Language Extensions (TLE) pour PostgreSQL est fourni en tant qu'extension `pg_tle`. En tant que tel, `pg_tle` est ajouté aux extensions disponibles pour la base de données dans laquelle il est installé.
2. TLE a son propre schéma, `pgtle`. Ce schéma contient des fonctions d'assistance (3) pour installer et gérer les extensions que vous créez.
3. TLE fournit plus d'une douzaine de fonctions d'assistance pour installer, enregistrer et gérer vos extensions. Pour en savoir plus sur ces fonctions, consultez Référence de fonction pour Trusted Language Extensions pour PostgreSQL.

L'extension `pg_tle` comprend les autres composants suivants :

- Le rôle **`pgtle_admin`** : le rôle `pgtle_admin` est créé lors de l'installation de l'extension `pg_tle`. Ce rôle est privilégié et doit être traité comme tel. Nous vous recommandons vivement de respecter le principe du moindre privilège lorsque vous accordez le rôle `pgtle_admin` aux utilisateurs de base de données. En d'autres termes, accordez le rôle `pgtle_admin` uniquement aux utilisateurs de base de données autorisés à créer, installer et gérer de nouvelles extensions TLE, telles que `postgres`.
- La table **`pgtle.feature_info`** : la table `pgtle.feature_info` est une table protégée qui contient des informations sur vos extensions TLE, vos hooks, ainsi que les procédures et fonctions stockées personnalisées qu'ils utilisent. Si vous disposez de privilèges `pgtle_admin`, vous pouvez utiliser les fonctions Trusted Language Extensions suivantes pour ajouter et mettre à jour ces informations dans la table.
  - pgtle.register_feature
  - pgtle.register_feature_if_not_exists
  - pgtle.unregister_feature
  - pgtle.unregister_feature_if_exists

Création d'extensions TLE pour Aurora PostgreSQL

Vous pouvez installer toutes les extensions que vous créez avec TLE dans n'importe quel cluster de bases de données Aurora PostgreSQL disposant de l'extension `pg_tle`. L'extension `pg_tle` s'applique à la base de données PostgreSQL dans laquelle elle est installée. Les extensions que vous créez à l'aide de TLE sont appliquées à la même base de données.

Utilisez les différentes fonctions `pgtle` pour installer le code qui constitue votre extension TLE. Les fonctions Trusted Language Extensions suivantes nécessitent toutes le rôle `pgtle_admin`.

- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(name, version)
- pgtle.uninstall_extension_if_exists

- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists
- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

Exemple : création d'une extension de langage approuvé utilisant SQL

L'exemple suivant montre comment créer une extension TLE nommée `pg_distance` et contenant quelques fonctions SQL permettant de calculer des distances à l'aide de différentes formules. Dans la liste, vous trouverez la fonction de calcul de la distance de Manhattan et la fonction de calcul de la distance euclidienne. Pour plus d'informations sur la différence entre ces formules, consultez Taxicab geometry (Géométrie de Manhattan) et Euclidean geometry (Géométrie euclidienne) sur Wikipedia.

Vous pouvez utiliser cet exemple dans votre propre cluster de bases de données Aurora PostgreSQL si vous disposez de l'extension `pg_tle` configurée comme indiqué dans Configuration de Trusted Language Extensions dans votre cluster de bases de données Aurora PostgreSQL.

> **Note**
>
> Vous devez disposer des privilèges du rôle `pgtle_admin` pour suivre cette procédure.

Pour créer l'exemple d'extension TLE

Les étapes suivantes utilisent un exemple de base de données nommé `labdb`. Cette base de données appartient à l'utilisateur principal `postgres`. Le rôle `postgres` possède également les autorisations du rôle `pgtle_admin`.

1. Utilisez `psql` pour vous connecter à l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL.

   ```
   psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
   --port=5432 --username=postgres --password --dbname=labdb
   ```

2. Créez une extension TLE nommée `pg_distance` en copiant le code suivant et en le collant dans votre console de session `psql`.

   ```
   SELECT pgtle.install_extension
   ```

```
(
 'pg_distance',
 '0.1',
  'Distance functions for two points',
$_pg_tle_$
    CREATE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8, norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL;

    CREATE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL;

    CREATE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL;
$_pg_tle_$
);
```

Vous devez voir la sortie suivante.

```
install_extension
---------------
 t
(1 row)
```

Les artefacts qui constituent l'extension `pg_distance` sont désormais installés dans votre base de données. Ces artefacts incluent le fichier de contrôle et le code de l'extension, qui sont des éléments qui doivent être présents pour que l'extension puisse être créée à l'aide de la commande `CREATE EXTENSION`. En d'autres termes, il vous reste à créer l'extension pour mettre ses fonctions à la disposition des utilisateurs de la base de données.

3. Pour créer cette extension, utilisez la commande `CREATE EXTENSION` comme vous le feriez pour toute autre extension. Comme pour les autres extensions, l'utilisateur de la base de données doit disposer des autorisations `CREATE` dans la base de données.

```
CREATE EXTENSION pg_distance;
```

4. Pour tester l'extension TLE `pg_distance`, vous pouvez l'utiliser pour calculer la distance de Manhattan entre quatre points.

```
labdb=> SELECT manhattan_dist(1, 1, 5, 5);
8
```

Pour calculer la distance euclidienne entre le même ensemble de points, vous pouvez utiliser ce qui suit.

```
labdb=> SELECT euclidean_dist(1, 1, 5, 5);
5.656854249492381
```

L'extension `pg_distance` charge les fonctions dans la base de données et les met à la disposition de tous les utilisateurs dotés d'autorisations sur la base de données.

Modification de votre extension TLE

Pour améliorer les performances des requêtes pour les fonctions incluses dans cette extension TLE, ajoutez les deux attributs PostgreSQL suivants à leurs spécifications.

- `IMMUTABLE` : l'attribut `IMMUTABLE` garantit que l'optimiseur de requêtes peut utiliser des optimisations pour améliorer les temps de réponse aux requêtes. Pour plus d'informations, consultez Function Volatility Categories (Catégories de volatilité des fonctions) dans la documentation PostgreSQL.
- `PARALLEL SAFE` : l'attribut `PARALLEL SAFE` est un autre attribut qui permet à PostgreSQL d'exécuter la fonction en mode parallèle. Pour plus d'informations, consultez CREATE FUNCTION dans la documentation PostgreSQL.

Dans l'exemple suivant, vous pouvez voir comment la fonction `pgtle.install_update_path` est utilisée pour ajouter ces attributs à chaque fonction afin de créer une version `0.2` de l'extension TLE `pg_distance`. Pour de plus amples informations sur cette fonction, veuillez consulter pgtle.install_update_path. Vous devez avoir le rôle `pgtle_admin` pour effectuer cette tâche.

Pour mettre à jour une extension TLE existante et spécifier la version par défaut

1. Connectez-vous à l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL à l'aide de `psql` ou d'un autre outil client, tel que pgAdmin.

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. Modifiez l'extension TLE existante en copiant le code suivant et en le collant dans votre console de session `psql`.

```
SELECT pgtle.install_update_path
(
 'pg_distance',
 '0.1',
 '0.2',
$_pg_tle_$
    CREATE OR REPLACE FUNCTION dist(x1 float8, y1 float8, x2 float8, y2 float8,
 norm int)
    RETURNS float8
    AS $$
      SELECT (abs(x2 - x1) ^ norm + abs(y2 - y1) ^ norm) ^ (1::float8 / norm);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION manhattan_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 1);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;

    CREATE OR REPLACE FUNCTION euclidean_dist(x1 float8, y1 float8, x2 float8, y2
 float8)
    RETURNS float8
    AS $$
      SELECT dist(x1, y1, x2, y2, 2);
    $$ LANGUAGE SQL IMMUTABLE PARALLEL SAFE;
$_pg_tle_$
);
```

Vous voyez une réponse similaire à la suivante.

```
install_update_path
--------------------
 t
(1 row)
```

Vous pouvez faire de cette version de l'extension la version par défaut, afin que les utilisateurs de base de données n'aient pas à spécifier de version lorsqu'ils créent ou mettent à jour l'extension dans leur base de données.

3. Pour spécifier que la version modifiée (version 0.2) de votre extension TLE est la version par défaut, utilisez la fonction `pgtle.set_default_version` comme indiqué dans l'exemple suivant.

```
SELECT pgtle.set_default_version('pg_distance', '0.2');
```

Pour de plus amples informations sur cette fonction, veuillez consulter pgtle.set_default_version.

4. Une fois le code en place, vous pouvez mettre à jour l'extension TLE installée de la manière habituelle, en utilisant la commande `ALTER EXTENSION ... UPDATE`, comme indiqué ici :

```
ALTER EXTENSION pg_distance UPDATE;
```

Suppression de vos extensions TLE d'une base de données

Vous pouvez supprimer vos extensions TLE en utilisant la commande `DROP EXTENSION` de la même manière que vous le feriez pour les autres extensions PostgreSQL. La suppression de l'extension ne supprime pas les fichiers d'installation qui composent l'extension et qui permettent aux utilisateurs de la recréer. Pour supprimer l'extension et ses fichiers d'installation, effectuez la procédure en deux étapes suivante.

Pour supprimer l'extension TLE et supprimer ses fichiers d'installation

1. Utilisez `psql` ou un autre outil client pour vous connecter à l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL..

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=dbname
```

2. Supprimez l'extension comme vous le feriez pour n'importe quelle extension PostgreSQL.

```
DROP EXTENSION your-TLE-extension
```

Par exemple, si vous créez l'extension `pg_distance` comme détaillé dans Exemple : création d'une extension de langage approuvé utilisant SQL, vous pouvez la supprimer comme suit.

```
DROP EXTENSION pg_distance;
```

Vous voyez une sortie confirmant que l'extension a été supprimée, comme suit.

```
DROP EXTENSION
```

À ce stade, l'extension n'est plus active dans la base de données. Cependant, ses fichiers d'installation et son fichier de contrôle sont toujours disponibles dans la base de données, ce qui permet aux utilisateurs de la base de données de recréer l'extension s'ils le souhaitent.

- Si vous souhaitez garder intacts les fichiers d'extension afin que les utilisateurs de la base de données puissent créer votre extension TLE, vous pouvez vous arrêter ici.
- Pour supprimer tous les fichiers qui composent l'extension, passez à l'étape suivante.

3. Pour supprimer tous les fichiers d'installation de votre extension, utilisez la fonction `pgtle.uninstall_extension`. Cette fonction supprime tous les fichiers de code et de contrôle relatifs à votre extension.

```
SELECT pgtle.uninstall_extension('your-tle-extension-name');
```

Par exemple, pour supprimer tous les fichiers d'installation `pg_distance`, utilisez la commande suivante.

```
SELECT pgtle.uninstall_extension('pg_distance');
 uninstall_extension
---------------------
 t
(1 row)
```

Désinstallation de Trusted Language Extensions pour PostgreSQL

Si vous ne souhaitez plus créer vos propres extensions TLE à l'aide de TLE, vous pouvez supprimer l'extension `pg_tle` et supprimer tous les artefacts. Cette action inclut la suppression de toutes les extensions TLE de la base de données et la suppression du schéma `pgtle`.

Pour supprimer l'extension **`pg_tle`** et son schéma d'une base de données

1. Utilisez `psql` ou un autre outil client pour vous connecter à l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL..

   ```
   psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
   port=5432 --username=postgres --password --dbname=dbname
   ```

2. Supprimez l'extension `pg_tle` de la base de données. Si vos propres extensions TLE s'exécutent encore dans la base de données, vous devez les supprimer également. Pour ce faire, vous pouvez utiliser le mot clé `CASCADE`, comme illustré ci-dessous.

   ```
   DROP EXTENSION pg_tle CASCADE;
   ```

   Si l'extension `pg_tle` n'est toujours pas active dans la base de données, vous n'avez pas besoin d'utiliser le mot clé `CASCADE`.

3. Supprimez le schéma `pgtle`. Cette action supprime toutes les fonctions de gestion de la base de données.

   ```
   DROP SCHEMA pgtle CASCADE;
   ```

   La commande renvoie ce qui suit une fois le processus terminé.

   ```
   DROP SCHEMA
   ```

   L'extension `pg_tle`, son schéma et ses fonctions, ainsi que tous les artefacts sont supprimés. Pour créer de nouvelles extensions à l'aide de TLE, répétez la procédure de configuration. Pour de plus amples informations, consultez Configuration de Trusted Language Extensions dans votre cluster de bases de données Aurora PostgreSQL.

Utilisation des hooks PostgreSQL avec vos extensions TLE

Un hook est un mécanisme de rappel disponible dans PostgreSQL qui permet aux développeurs d'appeler des fonctions personnalisées ou d'autres routines lors d'opérations de base de données normales. Le kit de développement TLE prend en charge les hooks PostgreSQL afin que vous puissiez intégrer des fonctions personnalisées au comportement PostgreSQL au moment de l'exécution. Par exemple, vous pouvez utiliser un hook pour associer le processus d'authentification à votre propre code personnalisé, ou pour modifier le processus de planification et d'exécution des requêtes en fonction de vos besoins spécifiques.

Vos extensions TLE peuvent utiliser des hooks. Si un hook a une portée globale, il s'applique à toutes les bases de données. Par conséquent, si votre extension TLE utilise un hook global, vous devez créer votre extension TLE dans toutes les bases de données auxquelles vos utilisateurs peuvent accéder.

Lorsque vous utilisez l'extension `pg_tle` pour créer votre propre kit Trusted Language Extensions, vous pouvez utiliser les hooks disponibles à partir d'une API SQL pour développer les fonctions de votre extension. Vous devez enregistrer tous les hooks avec `pg_tle`. Pour certains hooks, vous devrez peut-être également définir différents paramètres de configuration. Par exemple, le hook de vérification `passcode` peut être activé, désactivé ou requis. Pour plus d'informations sur les exigences spécifiques relatives aux hooks `pg_tle` disponibles, consultez Référence des hooks pour Trusted Language Extensions pour PostgreSQL.

Exemple : création d'une extension utilisant un hook PostgreSQL

L'exemple présenté dans cette section utilise un hook PostgreSQL pour vérifier le mot de passe fourni lors d'opérations SQL spécifiques et empêche les utilisateurs de base de données de définir leurs mots de passe sur l'un de ceux contenus dans la table `password_check.bad_passwords`. La table contient les dix choix de mots de passe les plus couramment utilisés, mais les plus faciles à déchiffrer.

Pour configurer cet exemple dans votre cluster de bases de données Aurora PostgreSQL, vous devez avoir déjà installé Trusted Language Extensions. Pour en savoir plus, consultez Configuration de Trusted Language Extensions dans votre cluster de bases de données Aurora PostgreSQL.

Pour configurer l'exemple de hook de vérification de mot de passe

1. Utilisez `psql` pour vous connecter à l'instance d'écriture de votre cluster de bases de données Aurora PostgreSQL.

```
psql --host=db-instance-123456789012.aws-region.rds.amazonaws.com
--port=5432 --username=postgres --password --dbname=labdb
```

2. Copiez le code à partir de Listing du code du hook de vérification de mot de passe et collez-le dans votre base de données.

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
```

```
          RAISE EXCEPTION 'Cannot use passwords from the common password
 dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password
 dictionary';
        END IF;
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

Lorsque l'extension a été chargée dans votre base de données, le résultat suivant s'affiche.

```
 install_extension
-------------------
 t
(1 row)
```

3. Toujours connecté à la base de données, vous pouvez maintenant créer l'extension.

```
CREATE EXTENSION my_password_check_rules;
```

4. Vous pouvez confirmer que l'extension a été créée dans la base de données à l'aide de la métacommande `psql` suivante.

```
\dx
                        List of installed extensions
          Name           | Version |   Schema   |
 Description
```

```
-------------------------+---------+------------
+-------------------------------------------------------------
 my_password_check_rules | 1.0     | public     | Prevent use of any of the top-ten
 most common bad passwords
 pg_tle                  | 1.0.1   | pgtle      | Trusted-Language Extensions for
 PostgreSQL
 plpgsql                 | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

5. Ouvrez une autre session de terminal pour travailler avec AWS CLI. Vous devez modifier votre groupe de paramètres de base de données personnalisé pour activer le hook de vérification de mot de passe. Pour ce faire, utilisez la commande CLI modify-db-parameter-group, comme illustré dans l'exemple suivant.

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

   L'application de la modification du groupe de paramètres peut prendre quelques minutes. Toutefois, ce paramètre étant dynamique, vous n'avez pas besoin de redémarrer l'instance d'écriture du cluster de bases de données Aurora PostgreSQL pour que le paramètre prenne effet.

6. Ouvrez la session `psql` et interrogez la base de données pour vérifier que le hook password_check a été activé.

```
labdb=> SHOW pgtle.enable_password_check;
pgtle.enable_password_check
----------------------------
on
(1 row)
```

Le hook de vérification de mot de passe est désormais actif. Vous pouvez le tester en créant un nouveau rôle et en utilisant l'un des mauvais mots de passe, comme illustré dans l'exemple suivant.

```
CREATE ROLE test_role PASSWORD 'password';
ERROR:  Cannot use passwords from the common password dictionary
```

```
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 21 at RAISE
SQL statement "SELECT password_check.passcheck_hook(
    $1::pg_catalog.text,
    $2::pg_catalog.text,
    $3::pgtle.password_types,
    $4::pg_catalog.timestamptz,
    $5::pg_catalog.bool)"
```

La sortie a été formatée pour être lisible.

L'exemple suivant montre que le comportement `pgsql` de la métacomcommande interactive `\password` est également affecté par le hook password_check.

```
postgres=> SET password_encryption TO 'md5';
SET
postgres=> \password
Enter new password for user "postgres":*****
Enter it again:*****
ERROR:  Cannot use passwords from the common password dictionary
CONTEXT:  PL/pgSQL function
 password_check.passcheck_hook(text,text,pgtle.password_types,timestamp with time
 zone,boolean) line 12 at RAISE
SQL statement "SELECT password_check.passcheck_hook($1::pg_catalog.text,
 $2::pg_catalog.text, $3::pgtle.password_types, $4::pg_catalog.timestamptz,
 $5::pg_catalog.bool)"
```

Vous pouvez supprimer cette extension TLE et désinstaller ses fichiers sources si vous le souhaitez. Pour de plus amples informations, consultez Suppression de vos extensions TLE d'une base de données.

Listing du code du hook de vérification de mot de passe

L'exemple de code affiché ici définit la spécification de l'extension TLE `my_password_check_rules`. Lorsque vous copiez ce code et que vous le collez dans votre base de données, le code de l'extension `my_password_check_rules` est chargé dans la base de données et le hook `password_check` est enregistré pour être utilisé par l'extension.

```
SELECT pgtle.install_extension (
  'my_password_check_rules',
  '1.0',
```

```
  'Do not let users use the 10 most commonly used passwords',
$_pgtle_$
  CREATE SCHEMA password_check;
  REVOKE ALL ON SCHEMA password_check FROM PUBLIC;
  GRANT USAGE ON SCHEMA password_check TO PUBLIC;

  CREATE TABLE password_check.bad_passwords (plaintext) AS
  VALUES
    ('123456'),
    ('password'),
    ('12345678'),
    ('qwerty'),
    ('123456789'),
    ('12345'),
    ('1234'),
    ('111111'),
    ('1234567'),
    ('dragon');
  CREATE UNIQUE INDEX ON password_check.bad_passwords (plaintext);

  CREATE FUNCTION password_check.passcheck_hook(username text, password text,
 password_type pgtle.password_types, valid_until timestamptz, valid_null boolean)
  RETURNS void AS $$
    DECLARE
      invalid bool := false;
    BEGIN
      IF password_type = 'PASSWORD_TYPE_MD5' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE ('md5' || md5(bp.plaintext || username)) = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
      ELSIF password_type = 'PASSWORD_TYPE_PLAINTEXT' THEN
        SELECT EXISTS(
          SELECT 1
          FROM password_check.bad_passwords bp
          WHERE bp.plaintext = password
        ) INTO invalid;
        IF invalid THEN
          RAISE EXCEPTION 'Cannot use passwords from the common password dictionary';
        END IF;
```

```
      END IF;
    END
  $$ LANGUAGE plpgsql SECURITY DEFINER;

  GRANT EXECUTE ON FUNCTION password_check.passcheck_hook TO PUBLIC;

  SELECT pgtle.register_feature('password_check.passcheck_hook', 'passcheck');
$_pgtle_$
);
```

Référence de fonction pour Trusted Language Extensions pour PostgreSQL

Consultez la documentation de référence suivante sur les fonctions disponibles dans Trusted Language Extensions pour PostgreSQL. Utilisez ces fonctions pour installer, enregistrer, mettre à jour et gérer vos extensions TLE, c'est-à-dire les extensions PostgreSQL que vous développez à l'aide du kit de développement Trusted Language Extensions.

Fonctions

- pgtle.available_extensions
- pgtle.available_extension_versions
- pgtle.extension_update_paths
- pgtle.install_extension
- pgtle.install_update_path
- pgtle.register_feature
- pgtle.register_feature_if_not_exists
- pgtle.set_default_version
- pgtle.uninstall_extension(name)
- pgtle.uninstall_extension(name, version)
- pgtle.uninstall_extension_if_exists
- pgtle.uninstall_update_path
- pgtle.uninstall_update_path_if_exists
- pgtle.unregister_feature
- pgtle.unregister_feature_if_exists

pgtle.available_extensions

La fonction `pgtle.available_extensions` est une fonction à renvoi d'ensemble. Elle renvoie toutes les extensions TLE disponibles dans la base de données. Chaque ligne renvoyée contient des informations sur une seule extension TLE.

Prototype de fonction

```
pgtle.available_extensions()
```

Rôle

Aucune.

Arguments

Aucune.

Sortie

- `name` : nom de l'extension TLE.
- `default_version` : la version de l'extension TLE à utiliser lorsque la commande `CREATE EXTENSION` est appelée sans version spécifiée.
- `description` : une description plus détaillée de l'extension TLE.

Exemple d'utilisation

```
SELECT * FROM pgtle.available_extensions();
```

pgtle.available_extension_versions

La fonction `available_extension_versions` est une fonction à renvoi d'ensemble. Elle renvoie une liste de toutes les extensions TLE disponibles et de leurs versions. Chaque ligne contient des informations sur une version spécifique de l'extension TLE donnée, y compris si elle nécessite un rôle spécifique.

Prototype de fonction

```
pgtle.available_extension_versions()
```

Rôle

Aucune.

Arguments

Aucune.

Sortie

- `name` : nom de l'extension TLE.
- `version` : version de l'extension TLE.
- `superuser` : cette valeur est toujours `false` pour vos extensions TLE. Les autorisations nécessaires pour créer l'extension TLE ou la mettre à jour sont les mêmes que pour la création d'autres objets dans la base de données donnée.
- `trusted` : cette valeur est toujours `false` pour une extension TLE.
- `relocatable` : cette valeur est toujours `false` pour une extension TLE.
- `schema` : spécifie le nom du schéma dans lequel l'extension TLE est installée.
- `requires` : tableau contenant les noms des autres extensions requises par cette extension TLE.
- `description` : description détaillée de l'extension TLE.

Pour plus d'informations sur les valeurs de sortie, consultez Packaging Related Objects into an Extension > Extension Files (Packaging des objets connexes dans une extension > Fichiers d'extension) dans la documentation de PostgreSQL.

Exemple d'utilisation

```
SELECT * FROM pgtle.available_extension_versions();
```

pgtle.extension_update_paths

La fonction `extension_update_paths` est une fonction à renvoi d'ensemble. Elle renvoie une liste de tous les chemins de mise à jour possibles pour une extension TLE. Chaque ligne comprend les mises à niveau ou les rétrogradations disponibles pour cette extension TLE.

Prototype de fonction

```
pgtle.extension_update_paths(name)
```

Rôle

Aucune.

Arguments

`name` : nom de l'extension TLE à partir de laquelle obtenir les chemins de mise à niveau.

Sortie

- `source` : version source d'une mise à jour.
- `target` : version cible d'une mise à jour.
- `path` : chemin de mise à niveau utilisé pour mettre à jour une extension TLE d'une version `source` à une version `target`, par exemple, `0.1--0.2`.

Exemple d'utilisation

```
SELECT * FROM pgtle.extension_update_paths('your-TLE');
```

pgtle.install_extension

La fonction `install_extension` vous permet d'installer les artefacts qui composent votre extension TLE dans la base de données, après quoi elle peut être créée à l'aide de la commande `CREATE EXTENSION`.

Prototype de fonction

```
pgtle.install_extension(name text, version text, description text, ext text, requires
 text[] DEFAULT NULL::text[])
```

Rôle

Aucune.

Arguments

- `name` : nom de l'extension TLE. Cette valeur est utilisée lors d'un appel de `CREATE EXTENSION`.
- `version` : version de l'extension TLE.

- `description` : description détaillée de l'extension TLE. Cette description est affichée dans le champ `comment` de `pgtle.available_extensions()`.
- `ext` : contenu de l'extension TLE. Cette valeur contient des objets tels que des fonctions.
- `requires` : paramètre facultatif qui spécifie les dépendances pour cette extension TLE. L'extension `pg_tle` est automatiquement ajoutée en tant que dépendance.

Plusieurs de ces arguments sont les mêmes que ceux qui sont inclus dans un fichier de contrôle d'extension pour installer une extension PostgreSQL sur le système de fichiers d'une instance PostgreSQL. Pour plus d'informations, consultez Extension Files (Fichiers d'extension) dans Packaging Related Objects into an Extension (Packaging des objets connexes dans une extension) de la documentation PostgreSQL.

Sortie

Cette fonction renvoie `OK` en cas de réussite ou `NULL` en cas d'erreur.

- `OK` : l'extension TLE a été installée avec succès dans la base de données.
- `NULL` : l'extension TLE n'a pas été installée dans la base de données.

Exemple d'utilisation

```
SELECT pgtle.install_extension(
 'pg_tle_test',
 '0.1',
 'My first pg_tle extension',
$_pgtle_$
  CREATE FUNCTION my_test()
  RETURNS INT
  AS $$
    SELECT 42;
  $$ LANGUAGE SQL IMMUTABLE;
$_pgtle_$
);
```

pgtle.install_update_path

La fonction `install_update_path` fournit un chemin de mise à jour entre deux versions différentes d'une extension TLE. Cette fonction permet aux utilisateurs de votre extension TLE de mettre à jour sa version en utilisant la syntaxe `ALTER EXTENSION ... UPDATE`.

Prototype de fonction

```
pgtle.install_update_path(name text, fromvers text, tovers text, ext text)
```

Rôle

`pgtle_admin`

Arguments

- `name` : nom de l'extension TLE. Cette valeur est utilisée lors d'un appel de `CREATE EXTENSION`.
- `fromvers` : version source de l'extension TLE pour la mise à niveau.
- `tovers` : version de destination de l'extension TLE pour la mise à niveau.
- `ext` : contenu de la mise à jour. Cette valeur contient des objets tels que des fonctions.

Sortie

Aucune.

Exemple d'utilisation

```
SELECT pgtle.install_update_path('pg_tle_test', '0.1', '0.2',
  $_pgtle_$
    CREATE OR REPLACE FUNCTION my_test()
    RETURNS INT
    AS $$
      SELECT 21;
    $$ LANGUAGE SQL IMMUTABLE;
  $_pgtle_$
);
```

pgtle.register_feature

La fonction `register_feature` ajoute la fonctionnalité interne PostgreSQL spécifiée à la table `pgtle.feature_info`. Les hooks PostgreSQL sont un exemple de fonctionnalité interne de PostgreSQL. Le kit de développement Trusted Language Extensions prend en charge l'utilisation des hooks PostgreSQL. Actuellement, cette fonction prend en charge la fonctionnalité suivante.

- `passcheck` : enregistre le hook de vérification de mot de passe avec votre procédure ou fonction qui personnalise le comportement de vérification de mot de passe de PostgreSQL.

Prototype de fonction

```
pgtle.register_feature(proc regproc, feature pg_tle_feature)
```

Rôle

`pgtle_admin`

Arguments

- `proc` : nom d'une procédure ou d'une fonction stockée à utiliser pour la fonctionnalité.
- `feature` : nom de la fonctionnalité `pg_tle` (tel que `passcheck`) à enregistrer avec la fonction.

Sortie

Aucune.

Exemple d'utilisation

```
SELECT pgtle.register_feature('pw_hook', 'passcheck');
```

pgtle.register_feature_if_not_exists

La fonction `pgtle.register_feature_if_not_exists` ajoute la fonctionnalité PostgreSQL spécifiée à la table `pgtle.feature_info` et identifie l'extension TLE ou toute autre procédure ou fonction qui utilise la fonctionnalité. Pour plus d'informations sur les hooks et le kit Trusted Language Extensions, consultez Utilisation des hooks PostgreSQL avec vos extensions TLE.

Prototype de fonction

```
pgtle.register_feature_if_not_exists(proc regproc, feature pg_tle_feature)
```

Rôle

`pgtle_admin`

Arguments

- `proc` : nom d'une procédure ou d'une fonction stockée qui contient la logique (code) à utiliser comme fonctionnalité pour votre extension TLE. Par exemple, le code `pw_hook`.

- `feature` : nom de la fonctionnalité PostgreSQL à enregistrer pour la fonction TLE. Actuellement, la seule fonctionnalité disponible est le hook `passcheck`. Pour plus d'informations, consultez Crochet de vérification du mot de passe (passcheck).

Sortie

Renvoie `true` après l'enregistrement de la fonction pour l'extension spécifiée. Renvoie `false` si la fonctionnalité est déjà enregistrée.

Exemple d'utilisation

```
SELECT pgtle.register_feature_if_not_exists('pw_hook', 'passcheck');
```

pgtle.set_default_version

La fonction `set_default_version` vous permet de spécifier une valeur `default_version` pour votre extension TLE. Vous pouvez utiliser cette fonction pour définir un chemin de mise à niveau et désigner la version comme étant la version par défaut pour votre extension TLE. Lorsque les utilisateurs de la base de données spécifient votre extension TLE dans les commandes `CREATE EXTENSION` et `ALTER EXTENSION ... UPDATE`, cette version de votre extension TLE est créée dans la base de données pour cet utilisateur.

Cette fonction renvoie `true` en cas de réussite. Si l'extension TLE spécifiée dans l'argument `name` n'existe pas, la fonction renvoie une erreur. De même, si la `version` de l'extension TLE n'existe pas, elle renvoie une erreur.

Prototype de fonction

```
pgtle.set_default_version(name text, version text)
```

Rôle

`pgtle_admin`

Arguments

- `name` : nom de l'extension TLE. Cette valeur est utilisée lors d'un appel de `CREATE EXTENSION`.
- `version` : version de l'extension TLE à définir par défaut.

Sortie

- `true` : lorsque la définition de la version par défaut réussit, la fonction renvoie `true`.
- `ERROR` : renvoie un message d'erreur si une extension TLE avec le nom ou la version spécifiés n'existe pas.

Exemple d'utilisation

```
SELECT * FROM pgtle.set_default_version('my-extension', '1.1');
```

pgtle.uninstall_extension(name)

La fonction `uninstall_extension` supprime toutes les versions d'une extension TLE d'une base de données. Cette fonction empêche les appels futurs de `CREATE EXTENSION` d'installer l'extension TLE. Si l'extension TLE n'existe pas dans la base de données, une erreur est signalée.

La fonction `uninstall_extension` n'abandonne pas une extension TLE qui est actuellement active dans la base de données. Pour supprimer une extension TLE actuellement active, vous devez appeler explicitement `DROP EXTENSION`.

Prototype de fonction

```
pgtle.uninstall_extension(extname text)
```

Rôle

`pgtle_admin`

Arguments

- `extname` : nom de l'extension TLE à désinstaller. Ce nom est le même que celui utilisé avec `CREATE EXTENSION` pour charger l'extension TLE à utiliser dans une base de données spécifiée.

Sortie

Aucune.

Exemple d'utilisation

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test');
```

pgtle.uninstall_extension(name, version)

La fonction `uninstall_extension(name, version)` supprime la version spécifiée de l'extension TLE de la base de données. Cette fonction empêche `CREATE EXTENSION` et `ALTER EXTENSION` d'installer ou de mettre à jour une extension TLE à la version spécifiée. Cette fonction supprime également tous les chemins de mise à jour pour la version spécifiée de l'extension TLE. Cette fonction ne désinstallera pas l'extension TLE si elle est actuellement active dans la base de données. Vous devez appeler explicitement `DROP EXTENSION` pour retirer l'extension TLE. Pour désinstaller toutes les versions d'une extension TLE, consultez pgtle.uninstall_extension(name).

Prototype de fonction

```
pgtle.uninstall_extension(extname text, version text)
```

Rôle

`pgtle_admin`

Arguments

- `extname` : nom de l'extension TLE. Cette valeur est utilisée lors d'un appel de `CREATE EXTENSION`.
- `version` : la version de l'extension TLE à désinstaller de la base de données.

Sortie

Aucune.

Exemple d'utilisation

```
SELECT * FROM pgtle.uninstall_extension('pg_tle_test', '0.2');
```

pgtle.uninstall_extension_if_exists

La fonction `uninstall_extension_if_exists` supprime toutes les versions d'une extension TLE d'une base de données spécifiée. Si l'extension TLE n'existe pas, la fonction ne renvoie rien (aucun message d'erreur n'est affiché). Si l'extension spécifiée est actuellement active dans une base de données, cette fonction ne la supprime pas. Vous devez explicitement appeler `DROP EXTENSION` pour supprimer l'extension TLE avant d'utiliser cette fonction pour désinstaller ses artefacts.

Prototype de fonction

```
pgtle.uninstall_extension_if_exists(extname text)
```

Rôle

`pgtle_admin`

Arguments

- `extname` : nom de l'extension TLE. Cette valeur est utilisée lors d'un appel de `CREATE EXTENSION`.

Sortie

La fonction `uninstall_extension_if_exists` renvoie `true` après avoir désinstallé l'extension spécifiée. Si l'extension spécifiée n'existe pas, la fonction renvoie `false`.

- `true` : renvoie `true` après la désinstallation de l'extension TLE.
- `false` : renvoie `false` lorsque l'extension TLE n'existe pas dans la base de données.

Exemple d'utilisation

```
SELECT * FROM pgtle.uninstall_extension_if_exists('pg_tle_test');
```

pgtle.uninstall_update_path

La fonction `uninstall_update_path` supprime le chemin de mise à jour spécifique pour l'extension TLE. Cela empêche `ALTER EXTENSION ... UPDATE TO` de l'utiliser comme chemin de mise à jour.

Si l'extension TLE est actuellement utilisée par l'une des versions de ce chemin de mise à jour, elle reste dans la base de données.

Si le chemin de mise à jour spécifié n'existe pas, cette fonction génère une erreur.

Prototype de fonction

```
pgtle.uninstall_update_path(extname text, fromvers text, tovers text)
```

Rôle

`pgtle_admin`

Arguments

- `extname` : nom de l'extension TLE. Cette valeur est utilisée lors d'un appel de `CREATE EXTENSION`.
- `fromvers` : la version source de l'extension TLE utilisée sur le chemin de mise à jour.
- `tovers` : la version destination de l'extension TLE utilisée sur le chemin de mise à jour.

Sortie

Aucune.

Exemple d'utilisation

```
SELECT * FROM pgtle.uninstall_update_path('pg_tle_test', '0.1', '0.2');
```

pgtle.uninstall_update_path_if_exists

La fonction `uninstall_update_path_if_exists` est similaire à `uninstall_update_path`, car elle supprime le chemin de mise à jour spécifié d'une extension TLE. Toutefois, si le chemin de mise à jour n'existe pas, cette fonction ne renvoie pas de message d'erreur. Au lieu de cela, la fonction renvoie `false`.

Prototype de fonction

```
pgtle.uninstall_update_path_if_exists(extname text, fromvers text, tovers text)
```

Rôle

`pgtle_admin`

Arguments

- `extname` : nom de l'extension TLE. Cette valeur est utilisée lors d'un appel de `CREATE EXTENSION`.

- `fromvers` : la version source de l'extension TLE utilisée sur le chemin de mise à jour.
- `tovers` : la version destination de l'extension TLE utilisée sur le chemin de mise à jour.

Sortie

- `true` : la fonction a mis à jour avec succès le chemin pour l'extension TLE.
- `false` : la fonction n'a pas pu mettre à jour le chemin pour l'extension TLE.

Exemple d'utilisation

```
SELECT * FROM pgtle.uninstall_update_path_if_exists('pg_tle_test', '0.1', '0.2');
```

pgtle.unregister_feature

La fonction `unregister_feature` permet de supprimer les fonctions qui ont été enregistrées pour utiliser des fonctionnalités `pg_tle`, telles que les hooks. Pour obtenir des informations sur l'enregistrement d'une fonctionnalité, consultez pgtle.register_feature.

Prototype de fonction

```
pgtle.unregister_feature(proc regproc, feature pg_tle_features)
```

Rôle

`pgtle_admin`

Arguments

- `proc` : nom d'une fonction stockée à enregistrer avec une fonctionnalité `pg_tle`.
- `feature` : nom de la fonctionnalité `pg_tle` à enregistrer avec la fonction. Par exemple, `passcheck` est une fonctionnalité qui peut être enregistrée pour être utilisée par les extensions Trusted Language Extensions (TLE) que vous développez. Pour plus d'informations, consultez Crochet de vérification du mot de passe (passcheck).

Sortie

Aucune.

Exemple d'utilisation

```
SELECT * FROM pgtle.unregister_feature('pw_hook', 'passcheck');
```

pgtle.unregister_feature_if_exists

La fonction `unregister_feature` permet de supprimer les fonctions qui ont été enregistrées pour utiliser des fonctionnalités `pg_tle`, telles que les hooks. Pour plus d'informations, consultez Utilisation des hooks PostgreSQL avec vos extensions TLE. Renvoie `true` après avoir réussi à annuler l'enregistrement de la fonctionnalité. Renvoie `false` si la fonctionnalité n'a pas été enregistrée.

Pour obtenir des informations sur l'enregistrement de fonctionnalités `pg_tle` pour vos extensions TLE, consultez pgtle.register_feature.

Prototype de fonction

```
pgtle.unregister_feature_if_exists('proc regproc', 'feature pg_tle_features')
```

Rôle

`pgtle_admin`

Arguments

- `proc` : nom de la fonction stockée qui a été enregistrée pour inclure une fonctionnalité `pg_tle`.
- `feature` : nom de la fonctionnalité `pg_tle` qui a été enregistrée avec l'extension de langage approuvé.

Sortie

Renvoie `true` ou `false`, comme suit.

- `true` : la fonction a annulé l'enregistrement de la fonctionnalité à l'extension.
- `false` : la fonction n'a pas pu annuler l'enregistrement de la fonctionnalité à l'extension TLE.

Exemple d'utilisation

```
SELECT * FROM pgtle.unregister_feature_if_exists('pw_hook', 'passcheck');
```

Référence des hooks pour Trusted Language Extensions pour PostgreSQL

Le kit Trusted Language Extensions pour PostgreSQL prend en charge les hooks PostgreSQL. Un hook est un mécanisme de rappel interne mis à la disposition des développeurs pour étendre les fonctionnalités de base de PostgreSQL. En utilisant des hooks, les développeurs peuvent implémenter leurs propres fonctions ou procédures à utiliser lors de diverses opérations de base de données, modifiant ainsi le comportement de PostgreSQL. Par exemple, vous pouvez utiliser un hook `passcheck` pour personnaliser la façon dont PostgreSQL gère les mots de passe fournis lors de la création ou de la modification de mots de passe pour les utilisateurs (rôles).

Consultez la documentation suivante pour en savoir plus sur le hook passcheck disponible pour vos extensions TLE. Pour en savoir plus sur les hooks disponibles, y compris le hook d'authentification client, consultez Hooks Trusted Language Extensions.

Crochet de vérification du mot de passe (passcheck)

Le crochet `passcheck` permet de personnaliser le comportement de PostgreSQL pendant le processus de vérification du mot de passe pour les commandes SQL et la métacommande `psql`suivantes.

- `CREATE ROLE` *`username`* `...PASSWORD` : pour plus d'informations, consultez CREATE USER (CRÉER UN RÔLE) dans la documentation PostgreSQL.
- `ALTER ROLE` *`username`*`...PASSWORD` : pour plus d'informations, consultez ALTER ROLE (ALTÉRER UN RÔLE) dans la documentation PostgreSQL.
- `\password` *`username`* : cette métacommande `psql` interactive modifie de manière sécurisée le mot de passe de l'utilisateur spécifié en hachant le mot de passe avant d'utiliser la syntaxe `ALTER ROLE ... PASSWORD` de manière transparente. La métacommande est un encapsuleur sécurisé pour la commande `ALTER ROLE ... PASSWORD`, et le crochet s'applique donc au comportement de la métacommande `psql`.

Pour obtenir un exemple, consultez Listing du code du hook de vérification de mot de passe.

Table des matières

- Notes d’utilisation

Prototype de fonction

```
passcheck_hook(username text, password text, password_type pgtle.password_types,
 valid_until timestamptz, valid_null boolean)
```

Arguments

Une fonction de crochet `passcheck` accepte les arguments suivants.

- `username` : nom (sous forme de texte) du rôle (nom d’utilisateur) qui définit un mot de passe.
- `password` : texte brut ou mot de passe haché. Le mot de passe saisi doit correspondre au type spécifié dans `password_type`.
- `password_type` : spécifiez le format `pgtle.password_type` du mot de passe. Ce format peut être l’une des options suivantes.
  - `PASSWORD_TYPE_PLAINTEXT` : un mot de passe en texte brut.
  - `PASSWORD_TYPE_MD5` : un mot de passe qui a été haché à l’aide de l’algorithme MD5 (message digest 5).
  - `PASSWORD_TYPE_SCRAM_SHA_256` : un mot de passe qui a été haché en utilisant l’algorithme SCRAM-SHA-256.
- `valid_until` : spécifiez l’heure à laquelle le mot de passe devient invalide. Cet argument est facultatif. Si vous utilisez cet argument, spécifiez l’heure comme une valeur `timestamptz`.
- `valid_null` : si cette valeur booléenne est définie sur `true`, l’option `valid_until` est définie sur `NULL`.

Configuration

La fonction `pgtle.enable_password_check` contrôle si le crochet passcheck est actif. Le crochet passcheck a trois paramètres possibles.

- `off` : désactive le crochet de vérification du mot de passe `passcheck`. C’est la valeur par défaut.
- on : active le crochet de vérification du mot de passe `passcode` afin que les mots de passe soient vérifiés dans la table.
- `require` : nécessite la définition d’un crochet de vérification du mot de passe.

Notes d’utilisation

Pour activer ou désactiver le crochet `passcheck`, vous devez modifier le groupe de paramètres de base de données personnalisé pour l’instance d’enregistreur de votre cluster de bases de données Aurora PostgreSQL.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-parameter-group \
    --region aws-region \
    --db-parameter-group-name your-custom-parameter-group \
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-parameter-group ^
    --region aws-region ^
    --db-parameter-group-name your-custom-parameter-group ^
    --parameters
 "ParameterName=pgtle.enable_password_check,ParameterValue=on,ApplyMethod=immediate"
```

Référence d’Amazon Aurora PostgreSQL

Les rubriques suivantes fournissent des informations sur les classements, les fonctions, les paramètres et les événements d’attente dans Amazon Aurora PostgreSQL.

Rubriques

Classements Aurora PostgreSQL pour EBCDIC et autres migrations de mainframe

La migration des applications mainframe vers de nouvelles plateformes telles qu’AWS préserve en général le comportement des applications. Pour préserver le comportement de l’application sur une nouvelle plateforme exactement comme il l’était sur le mainframe, il faut que les données migrées soient assemblées en utilisant les mêmes règles d’assemblage et de tri. Par exemple, de nombreuses solutions de migration Db2 déplacent les valeurs nulles vers u0180 (position Unicode 0180), de sorte que ces classements trient u0180 en premier. C’est un exemple de la façon dont les classements peuvent varier par rapport à leur source mainframe et de la raison pour laquelle il est nécessaire de choisir un classement qui correspond mieux au classement EBCDIC original.

Aurora PostgreSQL 14.3 et les versions ultérieures fournissent de nombreux classements ICU et EBCDIC pour prendre en charge une telle migration vers AWS en utilisant le service AWS Mainframe Modernization. Pour en savoir plus sur ce service, consultez Qu’est-ce que AWS Mainframe Modernization ?

Dans le tableau suivant, vous pouvez trouver les classements fournis par Aurora PostgreSQL. Ces classements suivent les règles EBCDIC et garantissent que les applications de mainframe fonctionnent de la même manière sur AWS que dans l’environnement du mainframe. Le nom du classement comprend la page de code correspondante (cpnnnn), ce qui vous permet de choisir le classement approprié pour votre source mainframe. Par exemple, utilisez `en-US-cp037-x-icu` pour obtenir le classement des données EBCDIC provenant d’une application mainframe qui utilise la page de code 037.

Classements EBCDIC	Classements AWS Blu Age	Classements AWS Micro Focus
da-DK-cp1142-x-icu	da-DK-cp1142b-x-icu	da-DK-cp1142m-x-icu
da-DK-cp277-x-icu	da-DK-cp277b-x-icu	–
de-DE-cp1141-x-icu	de-DE-cp1141b-x-icu	de-DE-cp1141m-x-icu
de-DE-cp273-x-icu	de-DE-cp273b-x-icu	–
en-GB-cp1146-x-icu	en-GB-cp1146b-x-icu	en-GB-cp1146m-x-icu
en-GB-cp285-x-icu	en-GB-cp285b-x-icu	–
en-US-cp037-x-icu	en-US-cp037b-x-icu	–
en-US-cp1140-x-icu	en-US-cp1140b-x-icu	en-US-cp1140m-x-icu
es-ES-cp1145-x-icu	es-ES-cp1145b-x-icu	es-ES-cp1145m-x-icu
es-ES-cp284-x-icu	es-ES-cp284b-x-icu	–
fi-FI-cp1143-x-icu	fi-FI-cp1143b-x-icu	fi-FI-cp1143m-x-icu
fi-FI-cp278-x-icu	fi-FI-cp278b-x-icu	–
fr-FR-cp1147-x-icu	fr-FR-cp1147b-x-icu	fr-FR-cp1147m-x-icu
fr-FR-cp297-x-icu	fr-FR-cp297b-x-icu	–
it-IT-cp1144-x-icu	it-IT-cp1144b-x-icu	it-IT-cp1144m-x-icu
it-IT-cp280-x-icu	it-IT-cp280b-x-icu	–
nl-BE-cp1148-x-icu	nl-BE-cp1148b-x-icu	nl-BE-cp1148m-x-icu
nl-BE-cp500-x-icu	nl-BE-cp500b-x-icu	–

Pour en savoir plus sur AWS Blu Age, consultez Tutoriel : runtime géré pour AWS Blu Age dans le Guide de l'utilisateur AWS Mainframe Modernization.

Pour obtenir plus d'informations sur l'utilisation de AWS Micro Focus, consultez Tutorial: Managed Runtime for Micro Focus (Tutoriel : runtime géré pour Micro Focus) dans le Guide de l'utilisateur AWS Mainframe Modernization.

Pour plus d'informations sur la gestion des classements dans PostgreSQL, consultez Collation Support (Prise en charge des classements) dans la documentation de PostgreSQL.

Les classements pris en charge dans Aurora PostgreSQL

Les classements sont un ensemble de règles qui déterminent la manière dont les chaînes de caractères stockées dans la base de données sont triées et comparées. Les classements jouent un rôle fondamental dans le système informatique et sont inclus dans le système d'exploitation. Les classements changent au fil du temps lorsque de nouveaux caractères sont ajoutés aux langues ou lorsque les règles de classement changent.

Les bibliothèques de classement définissent des règles et des algorithmes spécifiques pour un classement. Les bibliothèques de classement les plus populaires utilisées dans PostgreSQL sont GNU C (glibc) et les composants d'internationalisation pour Unicode (ICU). Par défaut, Aurora PostgreSQL utilise le classement glibc qui inclut les ordres de tri des caractères Unicode pour les séquences de caractères multi-octets.

Lorsque vous créez un nouveau cluster de bases de données Aurora PostgreSQL, le classement disponible est vérifié dans le système d'exploitation. Les paramètres PostgreSQL de la commande `CREATE DATABASE` `LC_COLLATE` et `LC_CTYPE` sont utilisés pour spécifier un classement, qui constitue le classement par défaut dans cette base de données. Vous pouvez également utiliser le paramètre `LOCALE` dans `CREATE DATABASE` pour définir ces paramètres. Cela détermine le classement par défaut pour les chaînes de caractères dans la base de données et les règles de classification des caractères sous forme de lettres, de chiffres ou de symboles. Vous pouvez également choisir un classement à utiliser sur une colonne, un index ou une requête.

Aurora PostgreSQL dépend de la bibliothèque glibc du système d'exploitation pour la prise en charge du classement. L'instance Aurora PostgreSQL est régulièrement mise à jour avec les dernières versions du système d'exploitation. Ces mises à jour incluent parfois une version plus récente de la bibliothèque glibc. Dans de rares cas, les nouvelles versions de glibc modifient l'ordre de tri ou le classement de certains caractères, ce qui peut entraîner un tri différent des données ou la production d'entrées d'index non valides. Si vous découvrez des problèmes d'ordre de tri pour le classement lors d'une mise à jour, vous devrez peut-être reconstruire les index.

Pour réduire les impacts possibles des mises à jour glibc, Aurora PostgreSQL inclut désormais une bibliothèque de classement par défaut indépendante. Cette bibliothèque de classement est disponible dans Aurora PostgreSQL 14.6, 13.9, 12.13, 11.18 et les versions mineures plus récentes. Elle est compatible avec glibc 2.26-59.amzn2 et assure la stabilité de l'ordre de tri afin d'éviter des résultats de requêtes incorrects.

Référence sur les fonctions Aurora PostgreSQL

Vous trouverez ci-dessous une liste des fonctions Aurora PostgreSQL disponibles pour vos clusters de bases de données Aurora qui exécutent le moteur de base de données compatible avec l'édition Aurora PostgreSQL. Ces fonctions Aurora PostgreSQL s'ajoutent aux fonctions PostgreSQL standard. Pour de plus amples informations sur les fonctions PostgreSQL standard, consultez PostgreSQL : fonctions et opérateurs.

Présentation

Vous pouvez utiliser les fonctions suivantes pour les instances de base de données Amazon RDS exécutant Aurora PostgreSQL :

- aurora_db_instance_identifier
- aurora_ccm_status
- aurora_global_db_instance_status
- aurora_global_db_status
- aurora_list_builtins
- aurora_replica_status
- aurora_stat_activity
- aurora_stat_backend_waits
- aurora_stat_bgwriter
- aurora_stat_database
- aurora_stat_dml_activity
- aurora_stat_get_db_commit_latency
- aurora_stat_logical_wal_cache
- aurora_stat_memctx_usage
- aurora_stat_optimized_reads_cache
- aurora_stat_plans

- aurora_stat_reset_wal_cache
- aurora_stat_statements
- aurora_stat_system_waits
- aurora_stat_wait_event
- aurora_stat_wait_type
- aurora_version
- aurora_volume_logical_start_lsn
- aurora_wait_report

aurora_db_instance_identifier

Indique nom de l'instance de base de données à laquelle vous êtes connecté.

Syntaxe

```
aurora_db_instance_identifier()
```

Arguments

Aucun

Type de retour

Chaîne VARCHAR

Notes d’utilisation

Cette fonction affiche le nom de l'instance de base de données du cluster d'Aurora Édition compatible avec PostgreSQL pour la connexion de votre client de base de données ou de votre application.

Cette fonction est disponible à partir de la version d'Aurora PostgreSQL 13.7, 12.11, 11.16, 10.21 et pour toutes les autres versions ultérieures.

Exemples

L'exemple suivant montre les résultats de l'appel de la fonction `aurora_db_instance_identifier`.

```
=> SELECT aurora_db_instance_identifier();
aurora_db_instance_identifier
------------------------------
 test-my-instance-name
```

Vous pouvez joindre les résultats de cette fonction avec la fonction `aurora_replica_status` pour obtenir des détails sur l'instance de base de données pour votre connexion. La fonction aurora_replica_status seule ne vous indique pas l'instance de base de données que vous utilisez. L'exemple suivant vous montre comment procéder.

```
=> SELECT *
    FROM aurora_replica_status() rt,
         aurora_db_instance_identifier() di
    WHERE rt.server_id = di;
-[ RECORD 1 ]----------------------+----------------------
server_id                         | test-my-instance-name
session_id                        | MASTER_SESSION_ID
durable_lsn                       | 88492069
highest_lsn_rcvd                  |
current_read_lsn                  |
cur_replay_latency_in_usec        |
active_txns                       |
is_current                        | t
last_transport_error              | 0
last_error_timestamp              |
last_update_timestamp             | 2022-06-03 11:18:25+00
feedback_xmin                     |
feedback_epoch                    |
replica_lag_in_msec               |
log_stream_speed_in_kib_per_second | 0
log_buffer_sequence_number        | 0
oldest_read_view_trx_id           |
oldest_read_view_lsn              |
pending_read_ios                  | 819
```

aurora_ccm_status

Affiche l'état du gestionnaire de cache du cluster.

Syntaxe

```
aurora_ccm_status()
```

Arguments

Aucune.

Type de retour

Registre SETOF comprenant les colonnes suivantes :

- `buffers_sent_last_minute` – Nombre de tampons envoyés au lecteur désigné au cours de la dernière minute.
- `buffers_found_last_minute` : nombre de tampons fréquemment consultés, identifiés au cours de la dernière minute.
- `buffers_sent_last_scan` – Nombre de tampons envoyés au lecteur désigné au cours de la dernière analyse terminée du cache du tampon.
- `buffers_found_last_scan` – Nombre de tampons fréquemment consultés envoyés au cours de la dernière analyse terminée du cache du tampon. Les tampons déjà mis en cache sur le lecteur désigné ne sont pas envoyés.
- `buffers_sent_current_scan` – Nombre de tampons envoyés au cours de l'analyse actuelle.
- `buffers_found_current_scan` – Nombre de tampons fréquemment consultés qui ont été identifiés au cours de l'analyse actuelle.
- `current_scan_progress` – Nombre de tampons visités jusqu'à maintenant au cours de l'analyse actuelle.

Notes d’utilisation

Vous pouvez utiliser cette fonction pour vérifier et surveiller la fonctionnalité de gestion de cache du cluster (CCM). Cette fonction fonctionne uniquement si la CCM est active sur votre cluster de bases de données Aurora PostgreSQL. Pour utiliser cette fonction, connectez-vous à l'instance de base de données d'écriture sur votre cluster de bases de données Aurora PostgreSQL.

Activez la CCM pour un cluster de bases de données Aurora PostgreSQL en définissant `apg_ccm_enabled` sur 1 dans le groupe de paramètres de cluster de bases de données personnalisé du cluster. Pour savoir comment procéder, veuillez consulter la section Configuration de la gestion des caches de clusters.

La gestion de cache du cluster est active sur un cluster de bases de données Aurora PostgreSQL lorsque le cluster dispose d'une instance Aurora Reader configurée comme suit :

- L'instance Aurora Reader utilise le même type et la même taille de classe d'instance de base de données que l'instance d'enregistreur du cluster.
- L'instance Aurora Reader est configurée en tant que niveau 0 pour le cluster. Si le cluster possède plusieurs lecteurs, il s'agit de son seul lecteur de niveau 0.

La définition de plusieurs lecteurs sur le niveau 0 désactive la CCM. Lorsque la CCM est désactivée, l'appel de cette fonction renvoie le message d'erreur suivant :

```
ERROR: Cluster Cache Manager is disabled
```

Vous pouvez également utiliser l'extension pg_buffercache PostgreSQL pour analyser le cache du tampon. Pour plus d'informations, consultez pg_buffercache dans la documentation de PostgreSQL.

Pour plus d'informations, consultez Introduction to Aurora PostgreSQL cluster cache management.

Exemples

L'exemple suivant montre les résultats de l'appel de la fonction `aurora_ccm_status`. Ce premier exemple montre les statistiques CCM.

```
=> SELECT * FROM aurora_ccm_status();
 buffers_sent_last_minute | buffers_found_last_minute | buffers_sent_last_scan |
 buffers_found_last_scan | buffers_sent_current_scan | buffers_found_current_scan |
 current_scan_progress
--------------------------+---------------------------+------------------------
+-------------------------+---------------------------+---------------------------
+-----------------------
                   2242000 |                   2242003 |               17920442 |
         17923410 |                  14098000 |                  14100964 |
   15877443
```

Pour plus de détails, vous pouvez utiliser l'affichage développé, comme indiqué ci-dessous :

```
\x
Expanded display is on.
SELECT *  FROM aurora_ccm_status();
[ RECORD 1 ]----------------------+---------
```

```
buffers_sent_last_minute           | 2242000
buffers_found_last_minute          | 2242003
buffers_sent_last_scan             | 17920442
buffers_found_last_scan            | 17923410
buffers_sent_current_scan          | 14098000
buffers_found_current_scan         | 14100964
current_scan_progress              | 15877443
```

Cet exemple montre comment vérifier le débit d'activation et le pourcentage d'activation.

```
=> SELECT buffers_sent_last_minute * 8/60 AS warm_rate_kbps,
100 * (1.0-buffers_sent_last_scan/buffers_found_last_scan) AS warm_percent
FROM aurora_ccm_status ();
 warm_rate_kbps | warm_percent
----------------+--------------
 16523 |        100.0
```

aurora_global_db_instance_status

Affiche l'état de toutes les instances Aurora, y compris les réplicas dans un cluster de bases de données global Aurora.

Syntaxe

```
aurora_global_db_instance_status()
```

Arguments

Aucun

Type de retour

Registre SETOF comprenant les colonnes suivantes :

- `server_id` – Identifiant de l'instance de base de données.
- `session_id` – Identifiant unique de la session en cours. La valeur `MASTER_SESSION_ID` identifie l'instance de base de données d'enregistreur (principale).
- `aws_region` – La Région AWS dans laquelle cette instance de base de données globale s'exécute. Pour obtenir la liste des régions, consultez Disponibilité dans les Régions.

- `durable_lsn` – Numéro de séquence de journal (LSN) rendu durable dans le stockage. Un numéro de séquence de journal (LSN) est un numéro séquentiel unique qui identifie un enregistrement dans le journal des transactions de la base de données. Les LSN sont classés de telle sorte qu'un LSN plus grand représente une transaction ultérieure.
- `highest_lsn_rcvd` – LSN le plus élevé reçu par l'instance de base de données en provenance de l'instance de base de données d'enregistreur.
- `feedback_epoch` – Époque utilisée par l'instance de base de données lorsqu'elle génère des informations de zone hébergée. Le terme zone hébergée fait référence à une instance de base de données qui prend en charge les connexions et les requêtes pendant que la base de données principale est en mode de récupération ou en mode veille. Les informations de zone hébergée incluent l'époque (instant dans le passé) et d'autres détails sur l'instance de base de données utilisée comme une zone hébergée. Pour de plus amples informations, veuillez consulter la documentation PostgreSQL sur Veille à chaud.
- `feedback_xmin` – ID de transaction actif minimal (le plus ancien) utilisé par l'instance de base de données.
- `oldest_read_view_lsn` : le LSN le plus ancien utilisé par l'instance de base de données pour lire à partir du stockage.
- `visibility_lag_in_msec` – Latence de cette instance de base de données par rapport à l'instance de base de données d'enregistreur en millisecondes.

Notes d’utilisation

Cette fonction affiche les statistiques de réplication pour un cluster de bases de données Aurora. Pour chaque instance de base de données Aurora PostgreSQL du cluster, la fonction affiche une ligne de données qui inclut tous les réplicas entre régions dans une configuration de base de données globale.

Vous pouvez exécuter cette fonction à partir de n'importe quelle instance d'un cluster de bases de données Aurora PostgreSQL ou d'une base de données globale Aurora PostgreSQL. La fonction renvoie des détails sur la latence décalage pour toutes les instances de réplica.

Pour en savoir plus sur la surveillance de la latence à l'aide de cette fonction (`aurora_global_db_instance_status`) ou de `aurora_global_db_status`, consultez Surveillance des bases de données globales basées sur Aurora PostgreSQL.

Pour plus d'informations sur les bases de données globales Aurora, consultez Présentation d’Amazon Aurora Global Database.

Pour commencer à utiliser les bases de données globales Aurora, consultez Mise en route avec Amazon Aurora Global Database ou la FAQ Amazon Aurora.

Exemples

Cet exemple montre les statistiques des instances entre régions.

```
=> SELECT *
  FROM aurora_global_db_instance_status();
               server_id                |              session_id              |
  aws_region  | durable_lsn | highest_lsn_rcvd | feedback_epoch | feedback_xmin |
 oldest_read_view_lsn | visibility_lag_in_msec
--------------------------------------+--------------------------------------
+-------------+-------------+------------------+----------------+--------------
+----------------------+------------------------
 db-119-001-instance-01                  | MASTER_SESSION_ID                    | eu-
west-1    |  2534560273 |           [NULL] |         [NULL] |        [NULL] |
     [NULL] |                [NULL]
 db-119-001-instance-02                  | 4ecff34d-d57c-409c-ba28-278b31d6fc40 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560266 |                      6
 db-119-001-instance-03                  | 3e8a20fc-be86-43d5-95e5-bdf19d27ad6b | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560266 |                      6
 db-119-001-instance-04                  | fc1b0023-e8b4-4361-bede-2a7e926cead6 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560254 |                     23
 db-119-001-instance-05                  | 30319b74-3f08-4e13-9728-e02aa1aa8649 | eu-
west-1    |  2534560266 |       2534560273 |              0 |      19669196 |
 2534560254 |                     23
 db-119-001-global-instance-1            | a331ffbb-d982-49ba-8973-527c96329c60 | eu-
central-1 |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                    996
 db-119-001-global-instance-1            | e0955367-7082-43c4-b4db-70674064a9da | eu-
west-2    |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                     14
 db-119-001-global-instance-1-eu-west-2a  | 1248dc12-d3a4-46f5-a9e2-85850491a897 | eu-
west-2    |  2534560254 |       2534560266 |              0 |      19669196 |
 2534560247 |                      0
```

Cet exemple montre comment vérifier la latence de réplica globale en millisecondes.

```
=> SELECT CASE
```

```
          WHEN 'MASTER_SESSION_ID' = session_id THEN 'Primary'
         ELSE 'Secondary'
      END AS global_role,
      aws_region,
      server_id,
      visibility_lag_in_msec
  FROM aurora_global_db_instance_status()
  ORDER BY 1, 2, 3;
   global_role |  aws_region  |               server_id               |
 visibility_lag_in_msec
-------------+-------------+---------------------------------------
+-----------------------
 Primary     | eu-west-1    | db-119-001-instance-01                |
 [NULL]
 Secondary   | eu-central-1 | db-119-001-global-instance-1          |
     13
 Secondary   | eu-west-1    | db-119-001-instance-02                |
     10
 Secondary   | eu-west-1    | db-119-001-instance-03                |
      9
 Secondary   | eu-west-1    | db-119-001-instance-04                |
      2
 Secondary   | eu-west-1    | db-119-001-instance-05                |
     18
 Secondary   | eu-west-2    | db-119-001-global-instance-1          |
     14
 Secondary   | eu-west-2    | db-119-001-global-instance-1-eu-west-2a |
     13
```

Cet exemple montre comment vérifier la latence minimale, maximale et moyenne par Région AWS à partir de la configuration de base de données globale.

```
=> SELECT 'Secondary' global_role,
       aws_region,
       min(visibility_lag_in_msec) min_lag_in_msec,
       max(visibility_lag_in_msec) max_lag_in_msec,
       round(avg(visibility_lag_in_msec),0) avg_lag_in_msec
  FROM aurora_global_db_instance_status()
 WHERE aws_region NOT IN (SELECT   aws_region
                            FROM aurora_global_db_instance_status()
                           WHERE session_id='MASTER_SESSION_ID')
                        GROUP BY aws_region
UNION ALL
```

```
SELECT  'Primary' global_role,
        aws_region,
        NULL,
        NULL,
        NULL
    FROM aurora_global_db_instance_status()
   WHERE session_id='MASTER_SESSION_ID'
ORDER BY 1, 5;
 global_role |  aws_region  | min_lag_in_msec | max_lag_in_msec | avg_lag_in_msec
-------------+--------------+-----------------+-----------------+-----------------
 Primary     | eu-west-1    |          [NULL] |          [NULL] |          [NULL]
 Secondary   | eu-central-1 |             133 |             133 |             133
 Secondary   | eu-west-2    |               0 |             495 |             248
```

aurora_global_db_status

Affiche des informations sur divers aspects du retard de la base de données globale Aurora, en particulier le retard du stockage Aurora sous-jacent (appelé « retard de durabilité ») et le retard entre l'objectif de point de reprise (RPO).

Syntaxe

```
aurora_global_db_status()
```

Arguments

Aucune.

Type de retour

Registre SETOF comprenant les colonnes suivantes :

- `aws_region` : Région AWS dans laquelle se trouve ce cluster de bases de données. Pour obtenir la liste complète des Régions AWS par moteur, consultez Régions et zones de disponibilité.
- `highest_lsn_written` – Numéro de séquence de journal (LSN) le plus élevé actuellement existant sur ce cluster de bases de données. Un numéro de séquence de journal (LSN) est un numéro séquentiel unique qui identifie un enregistrement dans le journal des transactions de la base de données. Les LSN sont classés de telle sorte qu'un LSN plus grand représente une transaction ultérieure.

- `durability_lag_in_msec` – Différence dans les valeurs d'horodatage entre `highest_lsn_written` sur un cluster de bases de données secondaire et `highest_lsn_written` sur le cluster de bases de données principal. La valeur -1 identifie le cluster de bases de données principal de la base de données globale Aurora.
- `rpo_lag_in_msec` : retard de l'objectif de point de reprise (RPO). Le retard RPO correspond au temps nécessaire au stockage de la transaction utilisateur COMMIT la plus récente sur un cluster de bases de données secondaire, après qu'elle a été stockée sur le cluster de bases de données principal de la base de données globale Aurora. La valeur -1 indique le cluster de bases de données principal (le retard n'est donc pas pertinent).

  En termes simples, cette métrique calcule l'objectif de point de reprise pour chaque cluster de bases de données Aurora PostgreSQL dans la base de données globale Aurora, c'est-à-dire la quantité de données qui risque d'être perdue en cas de panne. Comme pour la latence, le RPO est mesuré dans le temps.
- `last_lag_calculation_time` – Horodatage qui spécifie l'heure à laquelle les valeurs ont été calculées pour la dernière fois pour `durability_lag_in_msec` et `rpo_lag_in_msec`. Une valeur temporelle telle que `1970-01-01 00:00:00+00` signifie qu'il s'agit du cluster de bases de données principal.
- `feedback_epoch` – Époque utilisée par le cluster de bases de données secondaire lorsqu'il génère des informations de zone hébergée. Le terme zone hébergée fait référence à une instance de base de données qui prend en charge les connexions et les requêtes pendant que la base de données principale est en mode de récupération ou en mode veille. Les informations de zone hébergée incluent l'époque (instant dans le passé) et d'autres détails sur l'instance de base de données utilisée comme une zone hébergée. Pour plus d'informations, consultez Veille à chaud dans la documentation PostgreSQL.
- `feedback_xmin` – ID de transaction actif minimum (le plus ancien) utilisé par un cluster de bases de données secondaire.

Notes d'utilisation

Toutes les versions actuellement disponibles d'Aurora PostgreSQL prennent en charge cette fonction. Cette fonction affiche les statistiques de réplication pour une base de données globale Aurora. Elle affiche une ligne pour chaque cluster de bases de données dans une base de données globale Aurora PostgreSQL. Vous pouvez exécuter cette fonction à partir de n'importe quelle instance de votre base de données globale Aurora PostgreSQL.

Pour évaluer la latence de réplication de la base de données globale Aurora, qui est la latence des données visible, consultez aurora_global_db_instance_status.

Pour en savoir plus sur l'utilisation de `aurora_global_db_status` et `aurora_global_db_instance_status` pour surveiller la latence de la base de données globale Aurora, consultez Surveillance des bases de données globales basées sur Aurora PostgreSQL. Pour plus d'informations sur les bases de données globales Aurora, consultez Présentation d'Amazon Aurora Global Database.

Exemples

Cet exemple montre comment afficher les statistiques de stockage entre régions.

```
=> SELECT CASE
          WHEN '-1' = durability_lag_in_msec THEN 'Primary'
          ELSE 'Secondary'
       END AS global_role,
       *
  FROM aurora_global_db_status();
 global_role | aws_region | highest_lsn_written | durability_lag_in_msec |
 rpo_lag_in_msec | last_lag_calculation_time  | feedback_epoch | feedback_xmin
-------------+------------+---------------------+------------------------
+-----------------+----------------------------+----------------+---------------
 Primary     | eu-west-1  |           131031557 |                     -1 |
 -1 | 1970-01-01 00:00:00+00     |             0 |             0
 Secondary   | eu-west-2  |           131031554 |                    410 |
  0 | 2021-06-01 18:59:36.124+00 |             0 |         12640
 Secondary   | eu-west-3  |           131031554 |                    410 |
  0 | 2021-06-01 18:59:36.124+00 |             0 |         12640
```

aurora_list_builtins

Répertorie toutes les fonctions intégrées disponibles d'Aurora PostgreSQL, ainsi que de brèves descriptions et des détails sur les fonctions.

Syntaxe

```
aurora_list_builtins()
```

Arguments

Aucun

Type de retour

Registre SETOF

Exemples

L'exemple suivant montre les résultats de l'appel de la fonction `aurora_list_builtins`.

```
=> SELECT *
FROM aurora_list_builtins();

               Name                | Result data type |                   Argument data
 types                     | Type | Volatility |  Parallel  | Security |
             Description
-----------------------------------+-----------------
+-------------------------------------------------------------+------+------------
+------------+----------
+-------------------------------------------------------------------------
 aurora_version                    | text             |
                           | func | stable     | safe       | invoker  | Amazon Aurora
 PostgreSQL-Compatible Edition version string
 aurora_stat_wait_type             | SETOF record     | OUT type_id smallint, OUT
 type_name text                 | func | volatile   | restricted | invoker  | Lists all
 supported wait types
 aurora_stat_wait_event            | SETOF record     | OUT type_id smallint, OUT
 event_id integer, OUT event_na.| func | volatile   | restricted | invoker  | Lists all
 supported wait events
                                   |                  |.me text
                           |      |            |            |          |
 aurora_list_builtins              | SETOF record     | OUT "Name" text, OUT "Result
 data type" text, OUT "Argum.| func | stable     | safe       | invoker  | Lists all
 Aurora built-in functions
                                   |                  |.ent data types" text, OUT
 "Type" text, OUT "Volatility" .|      |            |            |          |
                                   |                  |.text, OUT "Parallel" text, OUT
 "Security" text, OUT "Des.|      |            |            |          |
                                   |                  |.cription" text
                           |      |            |            |          |
 .
 .
 .
 aurora_stat_file                  | SETOF record     | OUT filename text, OUT
 allocated_bytes bigint, OUT used_.| func | stable     | safe       | invoker  | Lists
 all files present in Aurora storage
```

```
                                    |                 |.bytes bigint
                          |      |           |           |          |
aurora_stat_get_db_commit_latency | bigint          | oid
                           | func | stable     | restricted | invoker  | Per DB commit
latency in microsecs
```

aurora_replica_status

Affiche l'état de tous les nœuds de lecture Aurora PostgreSQL.

Syntaxe

```
aurora_replica_status()
```

Arguments

Aucun

Type de retour

Registre SETOF avec les colonnes suivantes :

- `server_id` : l'ID (identifiant) de l'instance de base de données.
- `session_id` : un identifiant unique pour la séance en cours, renvoyé pour l'instance principale et les instances de lecture comme suit :
  - Pour l'instance principale, `session_id` est toujours égal à `` `MASTER_SESSION_ID' ``.
  - Pour les instances de lecture, `session_id` est toujours égal au `UUID` (identifiant universel et unique) de l'instance de lecture.
- `durable_lsn` : le numéro de séquence du journal (LSN) qui a été stocké.
  - Pour le volume principal, le LSN durable du volume principal (VDL) actuellement en vigueur.
  - Pour tout volume secondaire, le VDL du volume principal sur lequel le volume secondaire a été appliqué.

> **Note**
>
> Un numéro de séquence de journal (LSN) est un numéro séquentiel unique qui identifie un enregistrement dans le journal des transactions de la base de données. Les LSN sont

ordonnés de telle sorte qu'un LSN plus grand représente une transaction qui a eu lieu plus tard dans la séquence.

- `highest_lsn_rcvd` : le LSN le plus élevé (le plus récent) reçu par l'instance de base de données en provenance de l'instance de base de données en écriture.
- `current_read_lsn` : le LSN de l'instantané le plus récent qui a été appliqué à tous les lecteurs.
- `cur_replay_latency_in_usec` : le nombre de microsecondes attendu pour relire le journal sur le volume secondaire.
- `active_txns` : le nombre de transactions actuellement actives.
- `is_current` : non utilisé.
- `last_transport_error` : dernier code d'erreur de réplication.
- `last_error_timestamp` : horodatage de la dernière erreur de réplication.
- `last_update_timestamp` : horodatage de la dernière mise à jour de l'état du réplica. Depuis Aurora PostgreSQL 13.9, la valeur de `last_update_timestamp` pour l'instance de base de données à laquelle vous êtes connecté est définie sur `NULL`.
- `feedback_xmin` : la valeur feedback_xmin de secours du réplica. ID de transaction actif minimum (le plus ancien) utilisé par l'instance de base de données.
- `feedback_epoch` : l'époque utilisée par l'instance de base de données lorsqu'elle génère des informations de secours.
- `replica_lag_in_msec` : temps de retard de l'instance de lecteur par rapport à l'instance d'enregistreur, en millisecondes.
- `log_stream_speed_in_kib_per_second` : le débit du flux des journaux en kilo-octets par seconde.
- `log_buffer_sequence_number` : le numéro de séquence de la mémoire tampon du journal.
- `oldest_read_view_trx_id` : non utilisé.
- `oldest_read_view_lsn` : LSN le plus ancien utilisé par l'instance de base de données pour lire à partir du stockage.
- `pending_read_ios` : les lectures de pages en suspens qui sont toujours en attente sur le réplica.
- `read_ios` : le nombre total de pages lues sur le réplica.
- `iops` : non utilisé.
- `cpu` : utilisation du processeur par le démon de stockage Aurora pour chaque nœud du cluster. Pour plus d'informations sur l'utilisation du CPU par l'instance, consultez Métriques de niveau instance pour Amazon Aurora.

Notes d’utilisation

Toutes les versions actuellement disponibles d’Aurora PostgreSQL prennent en charge cette fonction. La fonction `aurora_replica_status` renvoie les valeurs du gestionnaire d’état des réplicas d’un cluster de bases de données Aurora PostgreSQL. Vous pouvez utiliser cette fonction pour obtenir des informations sur l’état de la réplication sur votre cluster de bases de données Aurora PostgreSQL, y compris les métriques pour toutes les instances de base de données dans votre cluster de bases de données Aurora. Par exemple, vous pouvez effectuer les opérations suivantes :

- Get information about the type of instance (writer, reader) in the Aurora PostgreSQL DB cluster (Obtenir des informations sur le type d’instance (écriture, lecture) dans le cluster de bases de données Aurora PostgreSQL) : vous pouvez obtenir ces informations en lisant les valeurs des colonnes suivantes :
  - `server_id` : contient le nom de l’instance que vous avez spécifié lorsque vous avez créé l’instance. Dans certains cas, comme pour l’instance principale (écriture), le nom est généralement créé pour vous en ajoutant -instance-1 au nom spécifié pour votre cluster de bases de données Aurora PostgreSQL.
  - `session_id` : le champ `session_id` indique si l’instance est une instance de lecture ou d’écriture. Pour une instance d’enregistreur, `session_id` est toujours défini sur `"MASTER_SESSION_ID"`. Pour une instance de lecture, `session_id` est défini sur l’`UUID` de l’instance de lecture en question.
- Diagnose common replication issues, such as replica lag (Diagnostiquer les problèmes de réplication courants, tels que le retard de réplica) : le retard de réplica est le temps, en millisecondes, pendant lequel le cache des pages d’une instance de lecture est en retard sur celui de l’instance d’enregistreur. Ce retard se produit parce que les clusters Aurora utilisent la réplication asynchrone, comme décrit dans Réplication avec Amazon Aurora. La valeur est indiquée dans la colonne `replica_lag_in_msec` des résultats renvoyés par cette fonction. Un retard peut également se produire lorsqu’une requête est annulée en raison de conflits avec la récupération sur un serveur de veille. Vous pouvez consulter `pg_stat_database_conflicts()` pour vérifier qu’un tel conflit est à l’origine du retard du réplica (ou non). Pour plus d’informations, consultez The Statistics Collector (Collecteur de statistiques) dans la documentation de PostgreSQL. Pour en savoir plus sur la haute disponibilité et la réplication, consultez la FAQ Amazon Aurora.

  Amazon CloudWatch enregistre les résultats `replica_lag_in_msec` dans le temps, en tant que métrique `AuroraReplicaLag`. Pour obtenir plus d’informations sur l’utilisation des métriques

CloudWatch pour Aurora, consultez Surveillance des métriques Amazon Aurora avec Amazon CloudWatch.

Pour en savoir plus sur le dépannage et les redémarrages des réplicas en lecture Aurora, consultez la section Why did my Amazon Aurora read replica fall behind and restart? (Pourquoi mon réplica en lecture Amazon Aurora a-t-il pris du retard et redémarré ?) dans le Centre AWS Support.

Exemples

L'exemple suivant montre comment obtenir l'état de réplication de toutes les instances d'un cluster de bases de données Aurora PostgreSQL :

```
=> SELECT *
FROM aurora_replica_status();
```

L'exemple suivant montre l'instance d'enregistreur dans le cluster de bases de données Aurora PostgreSQL `docs-lab-apg-main` :

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role
FROM aurora_replica_status()
WHERE session_id = 'MASTER_SESSION_ID';
        server_id       | instance_role
------------------------+---------------
 db-119-001-instance-01 | writer
```

L'exemple suivant répertorie toutes les instances de lecture d'un cluster :

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role
FROM aurora_replica_status()
WHERE session_id <> 'MASTER_SESSION_ID';
        server_id       | instance_role
------------------------+---------------
db-119-001-instance-02  | reader
```

```
db-119-001-instance-03  | reader
db-119-001-instance-04  | reader
db-119-001-instance-05  | reader
(4 rows)
```

L'exemple suivant répertorie toutes les instances, le retard de chaque instance par rapport à l'instance d'enregistreur et le temps écoulé depuis la dernière mise à jour :

```
=> SELECT server_id,
    CASE
        WHEN 'MASTER_SESSION_ID' = session_id THEN 'writer'
        ELSE 'reader'
    END AS instance_role,
    replica_lag_in_msec AS replica_lag_ms,
    round(extract (epoch FROM (SELECT age(clock_timestamp(), last_update_timestamp))) *
 1000) AS last_update_age_ms
FROM aurora_replica_status()
ORDER BY replica_lag_in_msec NULLS FIRST;
       server_id        | instance_role | replica_lag_ms | last_update_age_ms
------------------------+---------------+----------------+--------------------
 db-124-001-instance-03 | writer        |         [NULL] |               1756
 db-124-001-instance-01 | reader        |             13 |               1756
 db-124-001-instance-02 | reader        |             13 |               1756
(3 rows)
```

aurora_stat_activity

Renvoie une ligne par processus serveur, affichant les informations relatives à l'activité actuelle de ce processus.

Syntaxe

```
aurora_stat_activity();
```

Arguments

Aucun

Type de retour

Renvoie une ligne par processus serveur. Outre les colonnes `pg_stat_activity`, le champ suivant est ajouté :

- planid — identifiant du plan

Notes d’utilisation

Vue supplémentaire de `pg_stat_activity` renvoyant des mêmes colonnes avec une colonne `plan_id` supplémentaire qui montre le plan d’exécution de la requête.

`aurora_compute_plan_id` doit être activé pour que la vue renvoie un plan_id.

Cette fonction est disponible à partir des versions d’Aurora PostgreSQL 14.10, 15.5 et pour toutes les autres versions ultérieures.

Exemples

L’exemple de requête ci-dessous agrège la charge maximale par query_id et plan_id.

```
db1=# select count(*), query_id, plan_id
db1-# from aurora_stat_activity() where state = 'active'
db1-# and pid <> pg_backend_pid()
db1-# group by query_id, plan_id
db1-# order by 1 desc;

count |  query_id             |  plan_id
-------+----------------------+-------------
 11    | -5471422286312252535 | -2054628807
 3     | -6907107586630739258 | -815866029
 1     | 5213711845501580017  |  300482084
(3 rows)
```

Si le plan utilisé pour query_id change, un nouveau plan_id sera indiqué par aurora_stat_activity.

```
count  |  query_id             |  plan_id
-------+----------------------+-------------
 10    | -5471422286312252535 | 1602979607
 1     | -6907107586630739258 | -1809935983
 1     | -2446282393000597155 | -207532066
```

```
(3 rows)
```

aurora_stat_backend_waits

Affiche les statistiques relatives à l'activité d'attente pour un processus de backend spécifique.

Syntaxe

```
aurora_stat_backend_waits(pid)
```

Arguments

`pid` – ID du processus de backend. Vous pouvez obtenir des ID de processus à l'aide de la vue `pg_stat_activity`.

Type de retour

Registre SETOF comprenant les colonnes suivantes :

- `type_id` – Nombre qui indique le type d'événement d'attente, tel que 1 pour un verrou léger (`LWLock`), 3 pour un verrou, ou 6 pour une session client, pour en citer quelques exemples. Ces valeurs deviennent significatives lorsque vous joignez les résultats de cette fonction avec des colonnes issues de la fonction `aurora_stat_wait_type`, comme illustré dans les Exemples.
- `event_id` – Numéro d'identification de l'événement d'attente. Joignez cette valeur aux colonnes issues de `aurora_stat_wait_event` pour obtenir des noms d'événement significatifs.
- `waits` – Nombre d'attentes cumulées pour l'ID de processus spécifié.
- `wait_time` – Temps d'attente en millisecondes.

Notes d’utilisation

Vous pouvez utiliser cette fonction pour analyser des événements d'attente (session) de backend spécifiques, survenus depuis l'ouverture d'une connexion. Pour obtenir des informations plus significatives sur les noms et les types d'événements d'attente, vous pouvez combiner cette fonction `aurora_stat_wait_type` et `aurora_stat_wait_event` à l'aide de JOIN, comme indiqué dans les exemples.

Exemples

Cet exemple illustre toutes les attentes, tous les types et tous les noms d'événement pour l'ID de processus de backend 3027.

```
=> SELECT type_name, event_name, waits, wait_time
        FROM aurora_stat_backend_waits(3027)
NATURAL JOIN aurora_stat_wait_type()
NATURAL JOIN aurora_stat_wait_event();
type_name |       event_name       | waits | wait_time
----------+------------------------+-------+------------
 LWLock    | ProcArrayLock          |     3 |         27
 LWLock    | wal_insert             |   423 |      16336
 LWLock    | buffer_content         | 11840 |    1033634
 LWLock    | lock_manager           | 23821 |    5664506
 Lock      | tuple                  | 10258 |  152280165
 Lock      | transactionid          | 78340 | 1239808783
 Client    | ClientRead             | 34072 |   17616684
 IO        | ControlFileSyncUpdate  |     2 |          0
 IO        | ControlFileWriteUpdate |     4 |         32
 IO        | RelationMapRead        |     2 |        795
 IO        | WALWrite               | 36666 |      98623
 IO        | XactSync               |  4867 |    7331963
```

Cet exemple illustre les types d'attentes et les événements d'attente actuels et cumulatifs pour toutes les sessions actives (`pg_stat_activity state <> 'idle'`) (mais sans la session actuelle qui appelle la fonction (`pid <> pg_backend_pid()`).

```
=> SELECT a.pid,
            a.usename,
            a.app_name,
            a.current_wait_type,
            a.current_wait_event,
            a.current_state,
            wt.type_name AS wait_type,
            we.event_name AS wait_event,
            a.waits,
            a.wait_time
        FROM (SELECT pid,
                     usename,
                     left(application_name,16) AS app_name,
                     coalesce(wait_event_type,'CPU') AS current_wait_type,
                     coalesce(wait_event,'CPU') AS current_wait_event,
```

```
                    state AS current_state,
                    (aurora_stat_backend_waits(pid)).*
                FROM pg_stat_activity
               WHERE pid <> pg_backend_pid()
                 AND state <> 'idle') a
NATURAL JOIN aurora_stat_wait_type() wt
NATURAL JOIN aurora_stat_wait_event() we;
  pid  | username  | app_name | current_wait_type | current_wait_event | current_state |
 wait_type |       wait_event       | waits | wait_time
-------+----------+----------+-------------------+--------------------+---------------
+----------+------------------------+-------+-----------
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | wal_insert             |  1937 |     29975
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | buffer_content         | 22903 |    760498
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | lock_manager           | 10012 |    223207
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 Lock      | tuple                  | 20315 |  63081529
 .
 .
 .
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 IO        | WALWrite               | 93293 |    237440
 30099 | postgres | pgbench  | Lock              | transactionid      | active        |
 IO        | XactSync               | 13010 |  19525143
 30100 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | ProcArrayLock          |     6 |        53
 30100 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | wal_insert             |  1913 |     25450
 30100 | postgres | pgbench  | Lock              | transactionid      | active        |
 LWLock    | buffer_content         | 22874 |    778005
 .
 .
 .
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 LWLock    | ProcArrayLock          |     3 |        71
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 LWLock    | wal_insert             |  1940 |     27741
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 LWLock    | buffer_content         | 22962 |    776352
 30109 | postgres | pgbench  | IO                | XactSync           | active        |
 LWLock    | lock_manager           |  9879 |    218826
```

```
 30109 | postgres | pgbench  | IO                | XactSync          | active        |
 Lock      | tuple                  | 20401 |  63581306
 30109 | postgres | pgbench  | IO                | XactSync          | active        |
 Lock      | transactionid          | 50769 | 211645008
 30109 | postgres | pgbench  | IO                | XactSync          | active        |
 Client    | ClientRead             | 89901 |  44192439
```

Cet exemple illustre les trois (3) premiers types d'attente et événements d'attente cumulatifs en cours pour toutes les sessions actives (`pg_stat_activity state <> 'idle'`), à l'exception de la session actuelle (`pid <>pg_backend_pid()`).

```
=> SELECT top3.*
       FROM (SELECT a.pid,
                    a.usename,
                    a.app_name,
                    a.current_wait_type,
                    a.current_wait_event,
                    a.current_state,
                    wt.type_name AS wait_type,
                    we.event_name AS wait_event,
                    a.waits,
                    a.wait_time,
                    RANK() OVER (PARTITION BY pid ORDER BY a.wait_time DESC)
               FROM (SELECT pid,
                            usename,
                            left(application_name,16) AS app_name,
                            coalesce(wait_event_type,'CPU') AS current_wait_type,
                            coalesce(wait_event,'CPU') AS current_wait_event,
                            state AS current_state,
                            (aurora_stat_backend_waits(pid)).*
                       FROM pg_stat_activity
                      WHERE pid <> pg_backend_pid()
                        AND state <> 'idle') a
       NATURAL JOIN aurora_stat_wait_type() wt
       NATURAL JOIN aurora_stat_wait_event() we) top3
 WHERE RANK <=3;
  pid  | usename  | app_name | current_wait_type | current_wait_event | current_state |
 wait_type |   wait_event    |  waits  | wait_time  | rank
-------+----------+----------+-------------------+--------------------+---------------
+-----------+-----------------+---------+------------+------
 20567 | postgres | psql     | CPU               | CPU                | active        |
 LWLock    | wal_insert      |   25000 |   67512003 |    1
```

```
20567 | postgres | psql     | CPU               | CPU               | active        |
IO        | WALWrite        | 3071758 |    1016961 |    2
20567 | postgres | psql     | CPU               | CPU               | active        |
IO        | BufFileWrite    |   20750 |     184559 |    3
27743 | postgres | pgbench  | Lock              | transactionid     | active        |
Lock      | transactionid   |  237350 | 1265580011 |    1
27743 | postgres | pgbench  | Lock              | transactionid     | active        |
Lock      | tuple           |   93641 |  341472318 |    2
27743 | postgres | pgbench  | Lock              | transactionid     | active        |
Client    | ClientRead      |  417556 |  204796837 |    3
.
.
.
27745 | postgres | pgbench  | IO                | XactSync          | active        |
Lock      | transactionid   |  238068 | 1265816822 |    1
27745 | postgres | pgbench  | IO                | XactSync          | active        |
Lock      | tuple           |   93210 |  338312247 |    2
27745 | postgres | pgbench  | IO                | XactSync          | active        |
Client    | ClientRead      |  419157 |  207836533 |    3
27746 | postgres | pgbench  | Lock              | transactionid     | active        |
Lock      | transactionid   |  237621 | 1264528811 |    1
27746 | postgres | pgbench  | Lock              | transactionid     | active        |
Lock      | tuple           |   93563 |  339799310 |    2
27746 | postgres | pgbench  | Lock              | transactionid     | active        |
Client    | ClientRead      |  417304 |  208372727 |    3
```

aurora_stat_bgwriter

`aurora_stat_bgwriter` est une vue statistique comprenant des informations sur les écritures dans le cache Optimized Reads.

Syntaxe

```
aurora_stat_bgwriter()
```

Arguments

Aucun

Type de retour

Registre SETOF contenant toutes les colonnes `pg_stat_bgwriter` et les colonnes supplémentaires suivantes. Pour plus d'informations sur les colonnes `pg_stat_bgwriter`, consultez `pg_stat_bgwriter`.

Vous pouvez réinitialiser les statistiques de cette fonctionnalité en utilisant `pg_stat_reset_shared("bgwriter")`.

- `orcache_blks_written` : nombre total de blocs de données écrits dans le cache Optimized Reads.
- `orcache_blk_write_time` : si `track_io_timing` est activé, le temps total passé à écrire des blocs de données dans le cache Optimized Reads est enregistré en millisecondes. Pour plus d'informations, consultez track_io_timing.

Notes d'utilisation

Cette fonctionnalité est disponible dans les versions suivantes d'Aurora PostgreSQL :

- Version 15.4 et versions 15 ultérieures
- Version 14.9 et versions 14 ultérieures

Exemples

```
=> select * from aurora_stat_bgwriter();
-[ RECORD 1 ]-----------------+-----------
orcache_blks_written          | 246522
orcache_blk_write_time        | 339276.404
```

aurora_stat_database

Elle contient toutes les colonnes de pg_stat_database et en ajoute de nouvelles à la fin.

Syntaxe

```
aurora_stat_database()
```

Arguments

Aucun

Type de retour

Registre SETOF contenant toutes les colonnes `pg_stat_database` et les colonnes supplémentaires suivantes. Pour plus d'informations sur les colonnes `pg_stat_database`, consultez `pg_stat_database`.

- `storage_blks_read` : nombre total de blocs partagés lus à partir du stockage Aurora dans cette base de données.
- `orcache_blks_hit` : nombre total d'accès au cache Optimized Reads dans cette base de données.
- `local_blks_read` : nombre total de blocs locaux lus dans cette base de données.
- `storage_blk_read_time` : si `track_io_timing` est activé, le temps total passé à lire des blocs de données à partir du stockage Aurora est enregistré en millisecondes, sinon la valeur est nulle. Pour plus d'informations, consultez track_io_timing.
- `local_blk_read_time` : si `track_io_timing` est activé, le temps total passé à lire des blocs de données locales est enregistré en millisecondes, sinon la valeur est nulle. Pour plus d'informations, consultez track_io_timing.
- `orcache_blk_read_time` : si `track_io_timing` est activé, le temps total passé à lire des blocs de données à partir du cache Optimized Reads est enregistré en millisecondes, sinon la valeur est nulle. Pour plus d'informations, consultez track_io_timing.

> **Note**
>
> La valeur de `blks_read` est la somme de `storage_blks_read`, `orcache_blks_hit` et `local_blks_read`.
> La valeur de `blk_read_time` est la somme de `storage_blk_read_time`, `orcache_blk_read_time` et `local_blk_read_time`.

Notes d'utilisation

Cette fonctionnalité est disponible dans les versions suivantes d'Aurora PostgreSQL :

- Version 15.4 et versions 15 ultérieures
- Version 14.9 et versions 14 ultérieures

Exemples

L'exemple suivant montre comment elle transporte toutes les colonnes `pg_stat_database` et en ajoute six nouvelles à la fin :

```
=> select * from aurora_stat_database() where datid=14717;
-[ RECORD 1 ]------------+------------------------------
datid                    | 14717
datname                  | postgres
numbackends              | 1
xact_commit              | 223
xact_rollback            | 4
blks_read                | 1059
blks_hit                 | 11456
tup_returned             | 27746
tup_fetched              | 5220
tup_inserted             | 165
tup_updated              | 42
tup_deleted              | 91
conflicts                | 0
temp_files               | 0
temp_bytes               | 0
deadlocks                | 0
checksum_failures        |
checksum_last_failure    |
blk_read_time            | 3358.689
blk_write_time           | 0
session_time             | 1076007.997
active_time              | 3684.371
idle_in_transaction_time | 0
sessions                 | 10
sessions_abandoned       | 0
sessions_fatal           | 0
sessions_killed          | 0
stats_reset              | 2023-01-12 20:15:17.370601+00
orcache_blks_hit         | 425
orcache_blk_read_time    | 89.934
storage_blks_read        | 623
storage_blk_read_time    | 3254.914
```

```
local_blks_read          | 0
local_blk_read_time      | 0
```

aurora_stat_dml_activity

Indique l'activité cumulative pour chaque type d'opération de langage de manipulation de données (DML) sur une base de données dans un cluster Aurora PostgreSQL.

Syntaxe

```
aurora_stat_dml_activity(database_oid)
```

Arguments

database_oid

ID d'objet (OID) de la base de données dans le cluster Aurora PostgreSQL.

Type de retour

Registre SETOF

Notes d’utilisation

La fonction `aurora_stat_dml_activity` est disponible uniquement avec Aurora PostgreSQL version 3.1 compatible avec le moteur PostgreSQL 11.6 et versions ultérieures.

Utilisez cette fonction sur les clusters Aurora PostgreSQL avec un grand nombre de bases de données pour identifier les bases de données qui ont une activité DML supérieure ou plus lente, ou les deux.

La fonction `aurora_stat_dml_activity` renvoie le nombre de fois que les opérations ont été exécutées et la latence cumulative en microsecondes pour les opérations SELECT, INSERT, UPDATE et DELETE. Le rapport inclut uniquement les opérations DML réussies.

Vous pouvez réinitialiser cette statistique à l'aide de la fonction d'accès aux statistiques PostgreSQL `pg_stat_reset`. Vous pouvez vérifier quand cette statistique a été réinitialisée pour la dernière fois à l'aide de la fonction `pg_stat_get_db_stat_reset_time`. Pour plus d'informations sur

les fonctions d'accès aux statistiques PostgreSQL, consultez Collecteur de statistiques dans la documentation PostgreSQL.

Exemples

L'exemple suivant montre comment signaler les statistiques d'activité DML pour la base de données connectée.

```
--Define the oid variable from connected database by using \gset
=> SELECT oid,
          datname
     FROM pg_database
    WHERE datname=(select current_database()) \gset
=> SELECT *
     FROM aurora_stat_dml_activity(:oid);
select_count | select_latency_microsecs | insert_count | insert_latency_microsecs |
 update_count | update_latency_microsecs | delete_count | delete_latency_microsecs
--------------+--------------------------+--------------+--------------------------
+--------------+--------------------------+--------------+--------------------------
       178957 |                 66684115 |       171065 |                 28876649 |
    519538 |            1454579206167 |            1 |                    53027


-- Showing the same results with expanded display on
=> SELECT *
     FROM aurora_stat_dml_activity(:oid);
-[ RECORD 1 ]------------+-------------
select_count             | 178957
select_latency_microsecs | 66684115
insert_count             | 171065
insert_latency_microsecs | 28876649
update_count             | 519538
update_latency_microsecs | 1454579206167
delete_count             | 1
delete_latency_microsecs | 53027
```

L'exemple suivant montre les statistiques d'activité DML pour toutes les bases de données du cluster Aurora PostgreSQL. Ce groupe comprend deux bases de données, `postgres` et `mydb`. La liste séparée par des virgules correspond aux champs `select_count`, `select_latency_microsecs`, `insert_count`, `insert_latency_microsecs`, `update_count`, `update_latency_microsecs`, `delete_count` et `delete_latency_microsecs`.

Aurora PostgreSQL crée et utilise une base de données système nommée `rdsadmin` pour prendre en charge les opérations administratives telles que les sauvegardes, les restaurations, la surveillance de l'état, la réplication, etc. Ces opérations DML n'ont aucun impact sur votre cluster Aurora PostgreSQL.

```
=> SELECT oid,
    datname,
    aurora_stat_dml_activity(oid)
    FROM pg_database;
oid  |    datname     |                   aurora_stat_dml_activity
-------+----------------
+-----------------------------------------------------------------
 14006 | template0      | (,,,,,,,)
 16384 | rdsadmin       | (2346623,1211703821,4297518,817184554,0,0,0,0)
     1 | template1      | (,,,,,,,)
 14007 | postgres       |
 (178961,66716329,171065,28876649,519538,1454579206167,1,53027)
 16401 | mydb           | (200246,64302436,200036,107101855,600000,83659417514,0,0)
```

L'exemple suivant montre les statistiques d'activité DML pour toutes les bases de données, organisées en colonnes pour une meilleure lisibilité.

```
SELECT db.datname,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 1), '()') AS select_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 2), '()') AS select_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 3), '()') AS insert_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 4), '()') AS insert_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 5), '()') AS update_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 6), '()') AS update_latency_microsecs,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 7), '()') AS delete_count,
       BTRIM(SPLIT_PART(db.asdmla::TEXT, ',', 8), '()') AS delete_latency_microsecs
FROM  (SELECT datname,
              aurora_stat_dml_activity(oid) AS asdmla
         FROM pg_database
      ) AS db;

    datname     | select_count | select_latency_microsecs | insert_count |
 insert_latency_microsecs | update_count | update_latency_microsecs | delete_count |
 delete_latency_microsecs
----------------+--------------+--------------------------+--------------
+--------------------------+--------------+--------------------------+--------------
+--------------------------
```

```
 template0      |              |                          |              |
            |              |                          |              |
 rdsadmin       | 4206523      | 2478812333               | 7009414      | 1338482258
            | 0            | 0                        | 0            | 0
 template1      |              |                          |              |
            |              |                          |              |
 fault_test     | 66           | 452099                   | 0            | 0
            | 0            | 0                        | 0            | 0
 db_access_test | 1            | 5982                     | 0            | 0
            | 0            | 0                        | 0            | 0
 postgres       | 42035        | 95179203                 | 5752         | 2678832898
            | 21157        | 441883182488             | 2            | 1520
 mydb           | 71           | 453514                   | 0            | 0
            | 1            | 190                      | 1            | 152
```

L'exemple suivant montre la latence cumulative moyenne (latence cumulative divisée par le nombre) pour chaque opération DML pour la base de données avec l'OID 16401.

```
=> SELECT select_count,
          select_latency_microsecs,
          select_latency_microsecs/NULLIF(select_count,0) select_latency_per_exec,
          insert_count,
          insert_latency_microsecs,
          insert_latency_microsecs/NULLIF(insert_count,0) insert_latency_per_exec,
          update_count,
          update_latency_microsecs,
          update_latency_microsecs/NULLIF(update_count,0) update_latency_per_exec,
          delete_count,
          delete_latency_microsecs,
          delete_latency_microsecs/NULLIF(delete_count,0) delete_latency_per_exec
     FROM aurora_stat_dml_activity(16401);
-[ RECORD 1 ]------------+-------------
select_count             | 451312
select_latency_microsecs | 80205857
select_latency_per_exec  | 177
insert_count             | 451001
insert_latency_microsecs | 123667646
insert_latency_per_exec  | 274
update_count             | 1353067
update_latency_microsecs | 200900695615
update_latency_per_exec  | 148478
delete_count             | 12
delete_latency_microsecs | 448
```

```
delete_latency_per_exec  | 37
```

aurora_stat_get_db_commit_latency

Obtient la latence de validation cumulative en microsecondes pour les bases de données PostgreSQL Aurora. La latence de validation est mesurée comme le temps entre le moment où un client envoie une demande de validation et le moment où il reçoit la confirmation de validation.

Syntaxe

```
aurora_stat_get_db_commit_latency(database_oid)
```

Arguments

database_oid

ID d'objet (OID) de la base de données Aurora PostgreSQL.

Type de retour

Registre SETOF

Notes d'utilisation

Amazon CloudWatch utilise cette fonction pour calculer la latence de validation moyenne. Pour plus d'informations sur les métriques Amazon CloudWatch et sur la manière de résoudre une latence de validation élevée, consultez Affichage des métriques dans la console Amazon RDS et Prendre de meilleures décisions concernant Amazon RDS avec les métriques Amazon CloudWatch.

Vous pouvez réinitialiser cette statistique à l'aide de la fonction d'accès aux statistiques PostgreSQL `pg_stat_reset`. Vous pouvez vérifier quand cette statistique a été réinitialisée pour la dernière fois à l'aide de la fonction `pg_stat_get_db_stat_reset_time`. Pour plus d'informations sur les fonctions d'accès aux statistiques PostgreSQL, consultez Collecteur de statistiques dans la documentation PostgreSQL.

Exemples

L'exemple suivant obtient la latence de validation cumulative pour chaque base de données dans le cluster `pg_database`.

```
=> SELECT oid,
```

```
    datname,
    aurora_stat_get_db_commit_latency(oid)
    FROM pg_database;

  oid  |    datname     | aurora_stat_get_db_commit_latency
-------+----------------+-----------------------------------
 14006 | template0      |                                 0
 16384 | rdsadmin       |                         654387789
     1 | template1      |                                 0
 16401 | mydb           |                            229556
 69768 | postgres       |                             22011
```

L'exemple suivant obtient la latence de validation cumulative pour la base de données actuellement connectée. Avant d'appeler la fonction `aurora_stat_get_db_commit_latency`, l'exemple utilise d'abord `\gset` pour définir une variable pour l'argument `oid` et définit sa valeur à partir de la base de données connectée.

```
--Get the oid value from the connected database before calling
 aurora_stat_get_db_commit_latency
=> SELECT oid
     FROM pg_database
    WHERE datname=(SELECT current_database()) \gset
=> SELECT *
     FROM aurora_stat_get_db_commit_latency(:oid);

 aurora_stat_get_db_commit_latency
-----------------------------------
                        1424279160
```

L'exemple suivant obtient la latence de validation cumulative pour la base de données `mydb` dans le cluster `pg_database`. Ensuite, il réinitialise cette statistique à l'aide de la fonction `pg_stat_reset` et affiche les résultats. Enfin, il utilise la fonction `pg_stat_get_db_stat_reset_time` pour vérifier quand cette statistique a été réinitialisée pour la dernière fois.

```
=> SELECT oid,
    datname,
    aurora_stat_get_db_commit_latency(oid)
    FROM pg_database
    WHERE datname = 'mydb';

  oid  |  datname  | aurora_stat_get_db_commit_latency
-------+-----------+-----------------------------------
```

```
 16427 | mydb      |                              3320370


=> SELECT pg_stat_reset();
 pg_stat_reset
---------------


=> SELECT oid,
          datname,
          aurora_stat_get_db_commit_latency(oid)
     FROM pg_database
    WHERE datname = 'mydb';
  oid  |  datname  | aurora_stat_get_db_commit_latency
-------+-----------+-----------------------------------
 16427 | mydb      |                                 6


=> SELECT *
     FROM pg_stat_get_db_stat_reset_time(16427);

 pg_stat_get_db_stat_reset_time
--------------------------------
 2021-04-29 21:36:15.707399+00
```

aurora_stat_logical_wal_cache

Affiche l’utilisation du cache du journal d’écriture anticipée (WAL) par option.

Syntaxe

```
SELECT * FROM aurora_stat_logical_wal_cache()
```

Arguments

Aucun

Type de retour

Registre SETOF comprenant les colonnes suivantes :

- `name` : nom de l’emplacement de réplication.

- `active_pid` : ID du processus walsender.
- `cache_hit` : nombre total d'accès au cache wal depuis la dernière réinitialisation.
- `cache_miss` : nombre total d'erreurs d'accès au cache wal depuis la dernière réinitialisation.
- `blks_read` : nombre total de requêtes de lecture de cache wal.
- `hit_rate` : taux d'accès au cache WAL (cache_hit / blks_read).
- `last_reset_timestamp` : dernière fois que le compteur a été remis à zéro.

Notes d'utilisation

Cette fonction est disponible pour les versions suivantes d'Aurora PostgreSQL.

- Version 15.2 et toutes les versions ultérieures
- 14.7 et versions ultérieures
- 13.8 et versions ultérieures
- 12.12 et versions ultérieures
- 11.17 et versions ultérieures

Exemples

L'exemple suivant montre deux emplacements de réplication avec une seule fonction `aurora_stat_logical_wal_cache` active.

```
=> SELECT *
     FROM aurora_stat_logical_wal_cache();
    name    | active_pid | cache_hit | cache_miss | blks_read | hit_rate |
 last_reset_timestamp
------------+------------+-----------+------------+-----------+----------
+-------------------------------
 test_slot1 |      79183 |        24 |          0 |        24 | 100.00%  | 2022-08-05
 17:39:56.830635+00
 test_slot2 |            |         1 |          0 |         1 | 100.00%  | 2022-08-05
 17:34:04.036795+00
(2 rows)
```

aurora_stat_memctx_usage

Signale l'utilisation du contexte de mémoire pour chaque processus PostgreSQL.

Syntaxe

```
aurora_stat_memctx_usage()
```

Arguments

Aucun

Type de retour

Registre SETOF comprenant les colonnes suivantes :

- `pid` : ID du processus.
- `name` : nom du contexte de mémoire.
- `allocated` : nombre d'octets obtenus à partir du sous-système de mémoire sous-jacent par le contexte de mémoire.
- `used` : nombre d'octets validés vers les clients du contexte de mémoire.
- `instances` : nombre de contextes existants de ce type.

Notes d'utilisation

Cette fonction affiche l'utilisation du contexte de mémoire pour chaque processus PostgreSQL. Certains processus sont étiquetés `anonymous`. Les processus ne sont pas exposés car ils contiennent des mots clés restreints.

Cette fonction est disponible à compter des versions suivantes d'Aurora PostgreSQL :

- Version 15.3 et versions 15 ultérieures
- Version 14.8 et versions 14 ultérieures
- Version 13.11 et versions 13 ultérieures
- Version 12.15 et versions 12 ultérieures
- Version 11.20 et versions 11 ultérieures

Exemples

L'exemple suivant montre les résultats de l'appel de la fonction `aurora_stat_memctx_usage`.

```
=> SELECT *
```

```
     FROM aurora_stat_memctx_usage();

    pid| name                              | allocated |   used   | instances
-------+-----------------------------------+-----------+----------+-----------
123864 | Miscellaneous                     |     19520 |   15064 |         3
123864 | Aurora File Context               |      8192 |     616 |         1
123864 | Aurora WAL Context                |      8192 |     296 |         1
123864 | CacheMemoryContext                |    524288 |  422600 |         1
123864 | Catalog tuple context             |     16384 |   13736 |         1
123864 | ExecutorState                     |     32832 |   28304 |         1
123864 | ExprContext                       |      8192 |    1720 |         1
123864 | GWAL record construction          |      1024 |     832 |         1
123864 | MdSmgr                            |      8192 |     296 |         1
123864 | MessageContext                    |    532480 |  353832 |         1
123864 | PortalHeapMemory                  |      1024 |     488 |         1
123864 | PortalMemory                      |      8192 |     576 |         1
123864 | printtup                          |      8192 |     296 |         1
123864 | RelCache hash table entries       |      8192 |    8152 |         1
123864 | RowDescriptionContext             |      8192 |    1344 |         1
123864 | smgr relation context             |      8192 |     296 |         1
123864 | Table function arguments          |      8192 |     352 |         1
123864 | TopTransactionContext             |      8192 |     632 |         1
123864 | TransactionAbortContext           |     32768 |     296 |         1
123864 | WAL record construction           |     50216 |   43904 |         1
123864 | hash table                        |     65536 |   52744 |         6
123864 | Relation metadata                 |    191488 |  124240 |        87
104992 | Miscellaneous                     |      9280 |    7728 |         3
104992 | Aurora File Context               |      8192 |     376 |         1
104992 | Aurora WAL Context                |      8192 |     296 |         1
104992 ||Autovacuum Launcher               |      8192 |     296 |         1
104992 | Autovacuum database list          |     16384 |     744 |         2
104992 | CacheMemoryContext                |    262144 |  140288 |         1
104992 | Catalog tuple context             |      8192 |     296 |         1
104992 | GWAL record construction          |      1024 |     832 |         1
104992 | MdSmgr                            |      8192 |     296 |         1
104992 | PortalMemory                      |      8192 |     296 |         1
104992 | RelCache hash table entries       |      8192 |     296 |         1
104992 | smgr relation context             |      8192 |     296 |         1
104992 | Autovacuum start worker (tmp)     |      8192 |     296 |         1
104992 | TopTransactionContext             |     16384 |     592 |         2
104992 | TransactionAbortContext           |     32768 |     296 |         1
104992 | WAL record construction           |     50216 |   43904 |         1
104992 | hash table                        |     49152 |   34024 |         4
```

```
(39 rows)
```

Certains mots clés restreints seront masqués et le résultat se présentera comme suit :

```
postgres=>SELECT *
     FROM aurora_stat_memctx_usage();

    pid| name                             | allocated |   used  | instances
-------+----------------------------------+-----------+---------+-----------
  5482 | anonymous                        |      8192 |     456 |         1
  5482 | anonymous                        |      8192 |     296 |         1
```

aurora_stat_optimized_reads_cache

Cette fonction affiche les statistiques du cache à plusieurs niveaux.

Syntaxe

```
aurora_stat_optimized_reads_cache()
```

Arguments

Aucun

Type de retour

Registre SETOF comprenant les colonnes suivantes :

- `total_size` : taille totale du cache Optimized Reads.
- `used_size` : taille de page utilisée dans le cache Optimized Reads.

Notes d’utilisation

Cette fonctionnalité est disponible dans les versions suivantes d'Aurora PostgreSQL :

- Version 15.4 et versions 15 ultérieures
- Version 14.9 et versions 14 ultérieures

Exemples

Voici un exemple de la sortie d'une instance r6gd.8xlarge :

```
=> select pg_size_pretty(total_size) as total_size, pg_size_pretty(used_size)
                as used_size from aurora_stat_optimized_reads_cache();
total_size | used_size
-----------+-----------
1054 GB    | 975 GB
```

aurora_stat_plans

Renvoie une ligne pour chaque plan d’exécution suivi.

Syntaxe

```
aurora_stat_plans(
    showtext
)
```

Arguments

- showtext : affiche le texte de la requête et du plan. Les valeurs valides sont NULL, true ou false. True affiche le texte de la requête et du plan.

Type de retour

Renvoie une ligne pour chaque plan suivi qui contient toutes les colonnes d’`aurora_stat_statements` et les colonnes supplémentaires suivantes.

- planid — identifiant du plan
- explain_plan — explique le texte du plan
- plan_type:
  - `no plan` : aucun plan n’a été capturé
  - `estimate` : plan capturé avec estimation des coûts
  - `actual` : plan capturé avec EXPLAIN ANALYZE
- plan_captured_time — Dernière fois qu’un plan a été capturé

Notes d’utilisation

`aurora_compute_plan_id` doit être activé et `pg_stat_statements` doit être dans `shared_preload_libraries` pour que les plans puissent être suivis.

Le nombre de plans disponibles est contrôlé par la valeur définie dans le paramètre `pg_stat_statements.max`. Lorsque `aurora_compute_plan_id` est activé, vous pouvez suivre les plans jusqu’à cette valeur spécifiée dans `aurora_stat_plans`.

Cette fonction est disponible à partir des versions d’Aurora PostgreSQL 14.10, 15.5 et pour toutes les autres versions ultérieures.

Exemples

Dans l’exemple ci-dessous, les deux plans relatifs à l’identifiant de requête -5471422286312252535 sont capturés et les statistiques des instructions sont suivies par le planid.

```
db1=# select calls, total_exec_time, planid, plan_captured_time, explain_plan
db1-# from aurora_stat_plans(true)
db1-# where queryid = '-5471422286312252535'

calls    |  total_exec_time   |   planid    |      plan_captured_time       |
               explain_plan
---------+--------------------+-------------+-------------------------------
+-----------------------------------------------------------------
 1532632 |  3209846.097107853 |  1602979607 | 2023-10-31 03:27:16.925497+00 | Update on
 pgbench_branches                                      +
         |                    |             |                               |   ->
 Bitmap Heap Scan on pgbench_branches                     +
         |                    |             |                               |
 Recheck Cond: (bid = 76)                                   +
         |                    |             |                               |         -
>  Bitmap Index Scan on pgbench_branches_pkey          +
         |                    |             |                               |
     Index Cond: (bid = 76)
   61365 | 124078.18012200127 | -2054628807 | 2023-10-31 03:20:09.85429+00  | Update on
 pgbench_branches                                      +
         |                    |             |                               |   ->
 Index Scan using pgbench_branches_pkey on pgbench_branches+
         |                    |             |                               |
 Index Cond: (bid = 17)
```

aurora_stat_reset_wal_cache

Réinitialise le compteur du cache wal logique.

Syntaxe

Pour réinitialiser un emplacement spécifique

```
SELECT * FROM aurora_stat_reset_wal_cache('slot_name')
```

Pour réinitialiser tous les emplacements

```
SELECT * FROM aurora_stat_reset_wal_cache(NULL)
```

Arguments

`NULL` ou `slot_name`

Type de retour

Message d'état, chaîne de texte

- Réinitialisation du compteur de cache wal logique : message de réussite. Ce texte est renvoyé lorsque la fonction réussit.
- Emplacement de réplication introuvable. Veuillez réessayer. – Message d'échec Ce texte est renvoyé lorsque la fonction échoue.

Notes d’utilisation

Cette fonction est disponible pour les versions suivantes.

- Aurora PostgreSQL 14.5 et versions supérieures
- Aurora PostgreSQL versions 13.8 et ultérieures
- Aurora PostgreSQL versions 12.12 et ultérieures
- Aurora PostgreSQL version 11.17 et ultérieures

Exemples

L'exemple suivant utilise la fonction `aurora_stat_reset_wal_cache` pour réinitialiser un emplacement nommé `test_results`, puis tente de réinitialiser un emplacement qui n'existe pas.

```
=> SELECT *
     FROM aurora_stat_reset_wal_cache('test_slot');
aurora_stat_reset_wal_cache
------------------------------------
 Reset the logical wal cache counter.
(1 row)
=> SELECT *
     FROM aurora_stat_reset_wal_cache('slot-not-exist');
aurora_stat_reset_wal_cache
---------------------------------------------
 Replication slot not found. Please try again.
(1 row)
```

aurora_stat_resource_usage

Indique l’utilisation des ressources en temps réel, ce qui comprend les métriques des ressources dorsales et l’utilisation du processeur pour tous les processus dorsaux d’Aurora PostgreSQL.

Syntaxe

```
aurora_stat_resource_usage()
```

Arguments

Aucun

Type de retour

Enregistrement SETOF avec colonnes :

- pid — identifiant du processus
- allocated_memory — mémoire totale allouée par le processus en octets
- used_memory — mémoire réellement utilisée par le processus en octets
- cpu_usage_percent — pourcentage d’utilisation du processeur par le processus

Notes d'utilisation

Cette fonction affiche l'utilisation des ressources dorsales pour chaque processus dorsal d'Aurora PostgreSQL.

Cette fonction est disponible à compter des versions suivantes d'Aurora PostgreSQL :

- Aurora PostgreSQL 17.5 et versions 17 ultérieures
- Aurora PostgreSQL 16.9 et versions 16 ultérieures
- Aurora PostgreSQL 15.13 et versions 15 ultérieures
- Aurora PostgreSQL 14.18 et versions 14 ultérieures
- Aurora PostgreSQL 13.21 et versions 13 ultérieures

Exemples

L'exemple suivant illustre la sortie de la fonction `aurora_stat_resource_usage`.

```
=> select * from aurora_stat_resource_usage();
 pid  | allocated_memory | used_memory |   cpu_usage_percent
------+------------------+-------------+-----------------------
  666 |          1074032 |      333544 |   0.00729274882897963
  667 |           787312 |      287360 | 0.0029263928146372746
  668 |          3076776 |     1563488 |  0.006013116835953961
  684 |           803744 |      307480 |  0.002226855426881142
 2401 |          1232992 |      943144 |                     0
  647 |             8000 |         944 |   0.48853387812429855
  659 |           319344 |      243000 | 0.0004135602076683591
  663 |           262000 |      185736 |  0.008181301476644002
  664 |             9024 |        1216 |   0.10992313082653653
(9 rows)
```

aurora_stat_statements

Affiche toutes les colonnes `pg_stat_statements` et en ajoute d'autres à la fin.

Syntaxe

```
aurora_stat_statements(showtext boolean)
```

Arguments

showtext boolean

Type de retour

Registre SETOF contenant toutes les colonnes `pg_stat_statements` et les colonnes supplémentaires suivantes. Pour plus d'informations sur les colonnes `pg_stat_statements`, consultez `pg_stat_statements`.

Vous pouvez réinitialiser les statistiques de cette fonctionnalité en utilisant `pg_stat_statements_reset()`.

- `storage_blks_read` : nombre total de blocs partagés lus à partir du stockage Aurora par cette instruction.
- `orcache_blks_hit` : nombre total d'accès au cache Optimized reads par cette instruction.
- `storage_blk_read_time` : si `track_io_timing` est activé, le temps total que l'instruction a passé à lire des blocs partagés à partir du stockage Aurora est enregistré en millisecondes. Sinon, la valeur est nulle. Pour plus d'informations, consultez track_io_timing.
- `local_blk_read_time` : si `track_io_timing` est activé, le temps total que l'instruction a passé à lire des blocs locaux est enregistré en millisecondes. Sinon, la valeur est nulle. Pour plus d'informations, consultez track_io_timing.
- `orcache_blk_read_time` : si `track_io_timing` est activé, le temps total que l'instruction a passé à lire des blocs partagés à partir du cache Optimized Reads est enregistré en millisecondes. Sinon, la valeur est nulle. Pour plus d'informations, consultez track_io_timing.
- `total_plan_peakmem` : somme totale des valeurs de pic de mémoire pendant la phase de planification pour tous les appels envoyés à cette instruction. Pour connaître le pic de mémoire moyen lors de la planification de l'instruction, divisez cette valeur par le nombre d'appels.
- `min_plan_peakmem` : valeur de pic de mémoire la plus faible observée lors de la planification parmi tous les appels transmis à cette instruction.
- `max_plan_peakmem` : valeur de pic de mémoire la plus élevée observée lors de la planification parmi tous les appels transmis à cette instruction.
- `total_exec_peakmem` : somme totale des valeurs de pic de mémoire pendant la phase d'exécution pour tous les appels envoyés à cette instruction. Pour connaître le pic de mémoire moyen pendant l'exécution de l'instruction, divisez cette valeur par le nombre d'appels.
- `min_exec_peakmem` : valeur de pic de mémoire la plus faible, en octets, observée lors de l'exécution parmi tous les appels transmis à cette instruction.

- `max_exec_peakmem` : valeur de pic de mémoire la plus élevée, en octets, observée lors de l'exécution parmi tous les appels transmis à cette instruction.

> **Note**
>
> `total_plan_peakmen`, `min_plan_peakmem` et `max_plan_peakmem` ne sont surveillés que lorsque le paramètre `pg_stat_statements.track_planning` est activé.

Notes d'utilisation

Pour utiliser la fonction aurora_stat_statements(), vous devez inclure l'extension `pg_stat_statements` dans le paramètre `shared_preload_libraries`.

Cette fonctionnalité est disponible dans les versions suivantes d'Aurora PostgreSQL :

- 15.4 et versions 15 ultérieures
- 14.9 et versions 14 ultérieures

Les colonnes indiquant les pics de mémoire sont disponibles dans les versions suivantes :

- 16.3 et versions ultérieures
- 15.7 et versions ultérieures
- 14.12 et versions ultérieures

Exemples

L'exemple suivant montre comment elle transporte toutes les colonnes de l'instruction pg_stat_statements et en ajoute 11 nouvelles à la fin :

```
=> select * from aurora_stat_statements(true) where query like 'with window_max%';
-[ RECORD 1 ]----------
+-----------------------------------------------------------------------------------------------
userid                 | 16409
dbid                   | 5
toplevel               | t
queryid                | -8347523682669847482
query                  | with window_max as (select custid, max(scratch) over (order by
 scratch rows between $1 preceding
```

```
and $2 following) wmax from ts) select sum(wmax), max(custid) from window_max
plans                   | 0
total_plan_time         | 0
min_plan_time           | 0
max_plan_time           | 0
mean_plan_time          | 0
stddev_plan_time        | 0
calls                   | 4
total_exec_time         | 254.105121
min_exec_time           | 57.503164000000005
max_exec_time           | 68.687418
mean_exec_time          | 63.52628025
stddev_exec_time        | 5.150765359979643
rows                    | 4
shared_blks_hit         | 200192
shared_blks_read        | 0
shared_blks_dirtied     | 0
shared_blks_written     | 0
local_blks_hit          | 0
local_blks_read         | 0
local_blks_dirtied      | 0
local_blks_written      | 0
temp_blks_read          | 0
temp_blks_written       | 0
blk_read_time           | 0
blk_write_time          | 0
temp_blk_read_time      | 0
temp_blk_write_time     | 0
wal_records             | 0
wal_fpi                 | 0
wal_bytes               | 0
jit_functions           | 0
jit_generation_time     | 0
jit_inlining_count      | 0
jit_inlining_time       | 0
jit_optimization_count  | 0
jit_optimization_time   | 0
jit_emission_count      | 0
jit_emission_time       | 0
storage_blks_read       | 0
orcache_blks_hit        | 0
storage_blk_read_time   | 0
local_blk_read_time     | 0
orcache_blk_read_time   | 0
```

```
total_plan_peakmem     | 0
min_plan_peakmem       | 0
max_plan_peakmem       | 0
total_exec_peakmem     | 6356224
min_exec_peakmem       | 1589056
max_exec_peakmem       | 1589056
```

aurora_stat_system_waits

Indique les informations sur les événements d'attente pour l'instance de base de données PostgreSQL Aurora.

Syntaxe

```
aurora_stat_system_waits()
```

Arguments

Aucun

Type de retour

Registre SETOF

Notes d’utilisation

Cette fonction renvoie le nombre cumulé d'attente et le temps d'attente cumulé pour chaque événement d'attente généré par l'instance de base de données à laquelle vous êtes actuellement connecté.

Le jeu d'enregistrements comprend les champs suivants :

- `type_id` – ID du type d'événement d'attente.
- `event_id` – ID de l'événement d'attente.
- `waits` – Nombre de fois où l'événement d'attente s'est produit.
- `wait_time` – Temps total en microsecondes passé à attendre cet événement.

Les statistiques renvoyées par cette fonction sont réinitialisées lorsqu'une instance de base de données redémarre.

Exemples

L'exemple suivant montre les résultats de l'appel de la fonction `aurora_stat_system_waits`.

```
=> SELECT *
    FROM aurora_stat_system_waits();
 type_id | event_id  |   waits   |  wait_time
---------+-----------+-----------+--------------
       1 |  16777219 |        11 |        12864
       1 |  16777220 |       501 |       174473
       1 |  16777270 |     53171 |     23641847
       1 |  16777271 |        23 |       319668
       1 |  16777274 |        60 |        12759
.
.
.
      10 | 167772231 |    204596 |    790945212
      10 | 167772232 |         2 |        47729
      10 | 167772234 |         1 |          888
      10 | 167772235 |         2 |           64
```

L'exemple suivant montre comment utiliser cette fonction avec `aurora_stat_wait_event` et `aurora_stat_wait_type` pour produire des résultats plus lisibles.

```
=> SELECT type_name,
            event_name,
            waits,
            wait_time
        FROM aurora_stat_system_waits()
NATURAL JOIN aurora_stat_wait_event()
NATURAL JOIN aurora_stat_wait_type();

 type_name |       event_name       |  waits   |  wait_time
-----------+------------------------+----------+--------------
 LWLock    | XidGenLock             |       11 |        12864
 LWLock    | ProcArrayLock          |      501 |       174473
 LWLock    | buffer_content         |    53171 |     23641847
 LWLock    | rdsutils               |        2 |        12764
 Lock      | tuple                  |    75686 |   2033956052
 Lock      | transactionid          |  1765147 |  47267583409
 Activity  | AutoVacuumMain         |   136868 |  56305604538
 Activity  | BgWriterHibernate      |     7486 |  55266949471
 Activity  | BgWriterMain           |     7487 |   1508909964
```

```
.
.
.
 IO        | SLRURead               |        3 |        11756
 IO        | WALWrite               | 52544463 |    388850428
 IO        | XactSync               |   187073 |    597041642
 IO        | ClogRead               |        2 |        47729
 IO        | OutboundCtrlRead       |        1 |          888
 IO        | OutboundCtrlWrite      |        2 |           64
```

aurora_stat_wait_event

Répertorie tous les événements d'attente pris en charge pour Aurora PostgreSQL. Pour obtenir des informations sur les événements d'attente Aurora PostgreSQL, consultez Événements d’attente Amazon Aurora PostgreSQL.

Syntaxe

```
aurora_stat_wait_event()
```

Arguments

Aucun

Type de retour

Registre SETOF comprenant les colonnes suivantes :

- type_id – ID du type d'événement d'attente.
- event_id – ID de l'événement d'attente.
- type_name : nom du type d'attente
- event_name : nom de l'événement d'attente

Notes d’utilisation

Pour voir les noms d'événements avec le type d'événement au lieu de l'ID, utilisez cette fonction avec d'autres fonctions telles que `aurora_stat_wait_type` et `aurora_stat_system_waits`. Les noms des événements d'attente renvoyés par cette fonction sont les mêmes que ceux renvoyés par la fonction `aurora_wait_report`.

Exemples

L'exemple suivant montre les résultats de l'appel de la fonction `aurora_stat_wait_event`.

```
=>  SELECT *
    FROM aurora_stat_wait_event();

 type_id | event_id  |               event_name
---------+-----------+-----------------------------------------
       1 |  16777216 | <unassigned:0>
       1 |  16777217 | ShmemIndexLock
       1 |  16777218 | OidGenLock
       1 |  16777219 | XidGenLock
.
.
.
       9 | 150994945 | PgSleep
       9 | 150994946 | RecoveryApplyDelay
      10 | 167772160 | BufFileRead
      10 | 167772161 | BufFileWrite
      10 | 167772162 | ControlFileRead
.
.
.
      10 | 167772226 | WALInitWrite
      10 | 167772227 | WALRead
      10 | 167772228 | WALSync
      10 | 167772229 | WALSyncMethodAssign
      10 | 167772230 | WALWrite
      10 | 167772231 | XactSync
.
.
.
      11 | 184549377 | LsnAllocate
```

L'exemple suivant joint `aurora_stat_wait_type` et `aurora_stat_wait_event` pour renvoyer les noms de type et les noms d'événement pour une meilleure lisibilité.

```
=> SELECT *
    FROM aurora_stat_wait_type() t
    JOIN aurora_stat_wait_event() e
      ON t.type_id = e.type_id;
```

```
 type_id | type_name | type_id | event_id  |                event_name
---------+-----------+---------+-----------+-------------------------------------------
       1 | LWLock    |       1 |  16777216 | <unassigned:0>
       1 | LWLock    |       1 |  16777217 | ShmemIndexLock
       1 | LWLock    |       1 |  16777218 | OidGenLock
       1 | LWLock    |       1 |  16777219 | XidGenLock
       1 | LWLock    |       1 |  16777220 | ProcArrayLock
.
.
.
       3 | Lock      |       3 |  50331648 | relation
       3 | Lock      |       3 |  50331649 | extend
       3 | Lock      |       3 |  50331650 | page
       3 | Lock      |       3 |  50331651 | tuple
.
.
.
      10 | IO        |      10 | 167772214 | TimelineHistorySync
      10 | IO        |      10 | 167772215 | TimelineHistoryWrite
      10 | IO        |      10 | 167772216 | TwophaseFileRead
      10 | IO        |      10 | 167772217 | TwophaseFileSync
.
.
.
      11 | LSN       |      11 | 184549376 | LsnDurable
```

aurora_stat_wait_type

Répertorie tous les types d'attentes pris en charge pour Aurora PostgreSQL.

Syntaxe

```
aurora_stat_wait_type()
```

Arguments

Aucun

Type de retour

Registre SETOF comprenant les colonnes suivantes :

- type_id – ID du type d'événement d'attente.

- type_name – nom du type d'attente.

Notes d’utilisation

Pour voir les noms d'événements d'attente avec le type d'événement d'attente au lieu de l'ID, utilisez cette fonction avec d'autres fonctions telles que `aurora_stat_wait_event` et `aurora_stat_system_waits`. Les noms des types d'attentes renvoyés par cette fonction sont les mêmes que ceux renvoyés par la fonction `aurora_wait_report`.

Exemples

L'exemple suivant montre les résultats de l'appel de la fonction `aurora_stat_wait_type`.

```
=> SELECT *
     FROM aurora_stat_wait_type();
 type_id | type_name
---------+-----------
       1 | LWLock
       3 | Lock
       4 | BufferPin
       5 | Activity
       6 | Client
       7 | Extension
       8 | IPC
       9 | Timeout
      10 | IO
      11 | LSN
```

aurora_version

Renvoie la valeur de chaîne du numéro de version de l'Édition compatible avec PostgreSQL d'Amazon Aurora.

Syntaxe

```
aurora_version()
```

Arguments

Aucun

Type de retour

Chaîne CHAR ou VARCHAR

Notes d’utilisation

Cette fonction affiche la version du moteur de base de données Amazon Aurora Édition compatible avec PostgreSQL. Le numéro de version est renvoyé sous la forme d'une chaîne de caractères formatée comme *major*.*minor*.*patch*. Pour obtenir plus d'informations sur les numéros de versions d'Aurora PostgreSQL, consultez Numéro de version Aurora.

Vous pouvez choisir quand appliquer les mises à niveau de versions mineures en définissant la fenêtre de maintenance pour votre cluster de base de données Aurora PostgreSQL. Pour savoir comment procéder, veuillez consulter la section Entretien d’un cluster de bases de données Amazon Aurora.

À partir de la sortie d'Aurora PostgreSQL versions 13.3, 12.8, 11.13, 10.18, et pour toutes les autres versions ultérieures, les numéros de version d'Aurora suivent les numéros de version de PostgreSQL. Pour obtenir plus d'informations sur toutes les versions d'Aurora PostgreSQL, consultez Amazon Aurora PostgreSQL updates (Mises à jour d'Amazon Aurora PostgreSQL) dans les Notes de mise à jour d'Aurora PostgreSQL.

Exemples

L'exemple suivant montre les résultats de l'appel de la fonction `aurora_version` sur un cluster de base de données Aurora PostgreSQL exécutant PostgreSQL 12.7, Aurora PostgreSQL version 4.2, puis de l'exécution de la même fonction sur un cluster exécutant Aurora PostgreSQL version 13.3.

```
=> SELECT * FROM aurora_version();
aurora_version
----------------
 4.2.2
SELECT * FROM aurora_version();
aurora_version
----------------
 13.3.0
```

Cet exemple montre comment utiliser la fonction avec diverses options pour obtenir plus de détails sur la version Aurora PostgreSQL. Cet exemple présente un numéro de version d'Aurora distinct du numéro de version de PostgreSQL.

```
=> SHOW SERVER_VERSION;
 server_version
----------------
 12.7
(1 row)

=> SELECT * FROM aurora_version();
aurora_version
----------------
 4.2.2
(1 row)

=> SELECT current_setting('server_version') AS "PostgreSQL Compatiblility";
PostgreSQL Compatiblility
--------------------------
 12.7
(1 row)

=> SELECT version() AS "PostgreSQL Compatiblility Full String";
PostgreSQL Compatiblility Full String
-------------------------------------------------------------------------------------------------
 PostgreSQL 12.7 on aarch64-unknown-linux-gnu, compiled by aarch64-unknown-linux-gnu-
gcc (GCC) 7.4.0, 64-bit
(1 row)

=> SELECT 'Aurora: '
    || aurora_version()
    || ' Compatible with PostgreSQL: '
    || current_setting('server_version') AS "Instance Version";
Instance Version
-----------------------------------------------
 Aurora: 4.2.2 Compatible with PostgreSQL: 12.7
(1 row)
```

L'exemple suivant utilise la fonction avec les mêmes options que dans l'exemple précédent. Cet exemple ne présente pas de numéro de version Aurora distinct du numéro de version PostgreSQL.

```
=> SHOW SERVER_VERSION;
server_version
----------------
 13.3
```

```
=> SELECT * FROM aurora_version();
aurora_version
---------------
 13.3.0
=> SELECT current_setting('server_version') AS "PostgreSQL Compatiblility";
PostgreSQL Compatiblility
--------------------------
 13.3

=> SELECT version() AS "PostgreSQL Compatiblility Full String";
PostgreSQL Compatiblility Full String
-------------------------------------------------------------------------------------------------------
 PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by x86_64-pc-linux-gnu-gcc (GCC)
 7.4.0, 64-bit
=> SELECT 'Aurora: '
    || aurora_version()
    || ' Compatible with PostgreSQL: '
    || current_setting('server_version') AS "Instance Version";
Instance Version
-----------------------------------------------------
 Aurora: 13.3.0 Compatible with PostgreSQL: 13.3
```

aurora_volume_logical_start_lsn

Renvoie le numéro de séquence du journal (LSN) utilisé pour identifier le début d'un enregistrement dans le flux de journal d'écriture anticipée (WAL) logique du volume du cluster Aurora.

Syntaxe

```
aurora_volume_logical_start_lsn()
```

Arguments

Aucun

Type de retour

`pg_lsn`

Notes d’utilisation

Cette fonction identifie le début de l'enregistrement dans le flux WAL logique pour un volume de cluster Aurora donné. Vous pouvez utiliser cette fonction lors de la mise à niveau d'une version majeure à l'aide de la réplication logique et du clonage rapide Aurora afin de déterminer le LSN auquel un instantané ou un clone de base de données est pris. Vous pouvez ensuite utiliser la réplication logique pour diffuser en continu les nouvelles données enregistrées après le numéro LSN et synchroniser les modifications du diffuseur de publication à l'abonné.

Pour plus d'informations sur l'utilisation de la réplication logique pour une mise à niveau de version majeure, consultez Utilisation de la réplication logique pour effectuer une mise à niveau de version majeure pour Aurora PostgreSQL.

Cette fonction est disponible dans les versions suivantes d'Aurora PostgreSQL :

- Versions 15.2 et 15 ultérieures
- Versions 14.3 et 14 ultérieures
- Version 13.6 et versions 13 ultérieures
- Version 12.10 et versions 12 ultérieures
- Version 11.15 et versions 11 ultérieures
- Version 10.20 et versions 10 ultérieures

Exemples

Vous pouvez obtenir le numéro de séquence du journal (LSN) à l'aide de la requête suivante :

```
postgres=> SELECT aurora_volume_logical_start_lsn();

aurora_volume_logical_start_lsn
---------------
0/402E2F0
(1 row)
```

aurora_wait_report

Cette fonction affiche l’activité des événements d’attente sur une période.

Syntaxe

```
aurora_wait_report([time])
```

Arguments

time (optional) (temps (facultatif))

Le temps en secondes. La valeur par défaut est de 10 secondes.

Type de retour

Registre SETOF comprenant les colonnes suivantes :

- type_name : nom du type d'attente
- event_name : nom de l'événement d'attente
- wait : nombre d'attentes
- wait_time : temps d'attente en millisecondes
- ms_per_wait : moyenne en millisecondes par nombre d'attentes
- waits_per_xact : nombre moyen d'attentes sur le nombre de transactions
- ms_per_wait — moyenne en millisecondes par nombre de transactions

Notes d'utilisation

Cette fonction est disponible à partir de la version 1.1 d'Aurora PostgreSQL compatible avec PostgreSQL 9.6.6 et les versions ultérieures.

Pour utiliser cette fonction, vous devez d'abord créer l'extension Aurora PostgreSQL `aurora_stat_utils` comme suit :

```
=> CREATE extension aurora_stat_utils;
CREATE EXTENSION
```

Pour plus d'informations sur les versions d'extension d'Aurora PostgreSQL disponibles, consultez Versions d'extension d'Amazon Aurora PostgreSQL dans Notes de mise à jour d'Aurora PostgreSQL.

Cette fonction calcule les événements d'attente au niveau de l'instance en comparant deux instantanés de données statistiques provenant de la fonction aurora_stat_system_waits() et des vues des statistiques PostgreSQL pg_stat_database.

Pour plus d'informations concernant `aurora_stat_system_waits()` et `pg_stat_database`, consultez The Statistics Collector (Collecteur de statistiques) dans la documentation de PostgreSQL.

Lorsqu'elle est exécutée, cette fonction prend un instantané initial, attend le nombre de secondes spécifié, puis prend un deuxième instantané. La fonction compare les deux instantanés et renvoie la différence. Cette différence représente l'activité de l'instance pour cet intervalle de temps.

Sur l'instance d'enregistreur, la fonction affiche également le nombre de transactions validées et la valeur TPS (transactions par seconde). Cette fonction renvoie des informations au niveau de l'instance et inclut toutes les bases de données sur l'instance.

Exemples

Cet exemple montre comment créer l'extension aurora_stat_utils pour utiliser la fonction aurora_wait_report.

```
=> CREATE extension aurora_stat_utils;
CREATE EXTENSION
```

Cet exemple montre comment consulter le rapport d'attente pour 10 secondes.

```
=> SELECT *
     FROM aurora_wait_report();
NOTICE:  committed 34 transactions in 10 seconds (tps 3)
 type_name |    event_name     | waits | wait_time | ms_per_wait | waits_per_xact |
 ms_per_xact
-----------+-------------------+-------+-----------+-------------+----------------
+-------------
 Client    | ClientRead        |    26 |  30003.00 |    1153.961 |           0.76 |
 882.441
 Activity  | WalWriterMain     |    50 |  10051.32 |     201.026 |           1.47 |
 295.627
 Timeout   | PgSleep           |     1 |  10049.52 |   10049.516 |           0.03 |
 295.574
 Activity  | BgWriterHibernate |     1 |  10048.15 |   10048.153 |           0.03 |
 295.534
 Activity  | AutoVacuumMain    |    18 |   9941.66 |     552.314 |           0.53 |
 292.402
 Activity  | BgWriterMain      |     1 |    201.09 |     201.085 |           0.03 |
   5.914
 IO        | XactSync          |    15 |     25.34 |       1.690 |           0.44 |
   0.745
```

```
 IO        | RelationMapRead   |    12 |      0.54 |       0.045 |           0.35 |
   0.016
 IO        | WALWrite          |    84 |      0.21 |       0.002 |           2.47 |
   0.006
 IO        | DataFileExtend    |     1 |      0.02 |       0.018 |           0.03 |
   0.001
```

Cet exemple montre comment consulter le rapport d'attente pour 60 secondes.

```
=> SELECT *
     FROM aurora_wait_report(60);
NOTICE:  committed 1544 transactions in 60 seconds (tps 25)
 type_name |       event_name       |  waits  | wait_time | ms_per_wait |
 waits_per_xact | ms_per_xact
-----------+------------------------+---------+-----------+-------------
+----------------+-------------
 Lock      | transactionid          |    6422 | 477000.53 |      74.276 |
 4.16 |     308.938
 Client    | ClientRead             |    8265 | 270752.99 |      32.759 |
 5.35 |     175.358
 Activity  | CheckpointerMain       |       1 |  60100.25 |   60100.246 |
 0.00 |      38.925
 Timeout   | PgSleep                |       1 |  60098.49 |   60098.493 |
 0.00 |      38.924
 Activity  | WalWriterMain          |     296 |  60010.99 |     202.740 |
 0.19 |      38.867
 Activity  | AutoVacuumMain         |     107 |  59827.84 |     559.139 |
 0.07 |      38.749
 Activity  | BgWriterMain           |     290 |  58821.83 |     202.834 |
 0.19 |      38.097
 IO        | XactSync               |    1295 |  55220.13 |      42.641 |
 0.84 |      35.764
 IO        | WALWrite               | 6602259 |  47810.94 |       0.007 |
 4276.07 |      30.966
 Lock      | tuple                  |     473 |  29880.67 |      63.173 |
 0.31 |      19.353
 LWLock    | buffer_mapping         |     142 |   3540.13 |      24.930 |
 0.09 |       2.293
 Activity  | BgWriterHibernate      |     290 |   1124.15 |       3.876 |
 0.19 |       0.728
 IO        | BufFileRead            |    7615 |    618.45 |       0.081 |
 4.93 |       0.401
```

```
LWLock    | buffer_content         |      73 |    345.93 |       4.739 |
0.05 |       0.224
LWLock    | lock_manager           |      62 |    191.44 |       3.088 |
0.04 |       0.124
IO        | RelationMapRead        |      72 |      5.16 |       0.072 |
0.05 |       0.003
LWLock    | ProcArrayLock          |       1 |      2.01 |       2.008 |
0.00 |       0.001
IO        | ControlFileWriteUpdate |       2 |      0.03 |       0.013 |
0.00 |       0.000
IO        | DataFileExtend         |       1 |      0.02 |       0.018 |
0.00 |       0.000
IO        | ControlFileSyncUpdate  |       1 |      0.00 |       0.000 |
0.00 |       0.000
```

Paramètres Amazon Aurora PostgreSQL.

Vous gérez votre cluster de bases de données Amazon Aurora PostgreSQL de la même façon que les instances de base de données Amazon RDS, en utilisant les paramètres d'un groupe de paramètres de base de données. Cependant, Amazon Aurora diffère d'Amazon RDS en ce qu'un cluster de bases de données Aurora possède plusieurs instances de base de données. Certains des paramètres que vous utilisez pour gérer votre cluster de bases de données Amazon Aurora s'appliquent à la totalité du cluster, tandis que les autres paramètres s'appliquent uniquement à une instance de base de données spécifique du cluster de bases de données, comme suit :

- Groupe de paramètres de cluster de bases de données : un groupe de paramètres de cluster de bases de données contient l'ensemble des paramètres de configuration du moteur qui s'appliquent à l'ensemble du cluster de bases de données Aurora. Par exemple, la gestion des caches de clusters est une fonction d'un cluster de bases de données Aurora contrôlée par le paramètre `apg_ccm_enabled`, qui fait partie du groupe de paramètres de cluster de bases de données. Le groupe de paramètres de cluster de bases de données contient également les paramètres par défaut du groupe de paramètres de base de données pour les instances de base de données qui composent le cluster.
- Groupe de paramètres de base de données : un groupe de paramètres de base de données est l'ensemble de valeurs de configuration du moteur qui s'appliquent à une instance de base de données spécifique de ce type de moteur. Les groupes de paramètres de base de données du moteur de base de données PostgreSQL sont utilisés par une instance de base de données RDS pour PostgreSQL et un cluster de bases de données Aurora PostgreSQL. Ces paramètres

de configuration s'appliquent à des propriétés qui peuvent varier entre les instances de base de données d'un cluster Aurora comme, par exemple, les tailles des mémoires tampons.

Vous gérez les paramètres de niveau cluster dans les groupes de paramètres de cluster de bases de données. Vous gérez les paramètres de niveau instance dans les groupes de paramètres de base de données. Vous pouvez gérer les paramètres à l'aide de la console Amazon RDS, de la AWS CLI ou de l'API Amazon RDS. Il existe des commandes distinctes pour la gestion des paramètres de niveau cluster et des paramètres de niveau instance.

- Pour gérer les paramètres de niveau cluster dans un groupe de paramètres de cluster de bases de données, utilisez la commande AWS CLI modify-db-cluster-parameter-group.
- Pour gérer les paramètres de niveau instance d'un groupe de paramètres de base de données pour une instance de base de données dans un cluster de bases de données, utilisez la commande AWS CLI modify-db-parameter-group.

Pour en savoir plus sur la AWS CLI, consultez Utilisation de la AWS CLI dans le Guide de l'utilisateur AWS Command Line Interface.

Pour plus d'informations sur les groupes de paramètres, consultez Groupes de paramètres pour Amazon Aurora.

Affichage des paramètres de cluster de bases de données Aurora PostgreSQL et de base de données

Vous pouvez afficher tous les groupes de paramètres par défaut disponibles pour RDS pour les instances de base de données PostgreSQL et pour les clusters de bases de données Aurora PostgreSQL dans la AWS Management Console. Les groupes de paramètres par défaut pour l'ensemble des types et des versions des moteurs et des clusters de bases de données sont répertoriés pour chaque région AWS. Tous les groupes de paramètres personnalisés sont également répertoriés.

Plutôt que de les afficher dans la AWS Management Console, vous pouvez également répertorier les paramètres contenus dans des groupes de paramètres de cluster de bases de données et des groupes de paramètres de base de données à l'aide de la AWS CLI ou de l'API Amazon RDS. Par exemple, pour répertorier les paramètres d'un groupe de paramètres de cluster de bases de données, utilisez la commande describe-db-cluster-parameters de la AWS CLI.

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12
```

La commande renvoie des descriptions JSON détaillées de chaque paramètre. Pour réduire la quantité d'informations renvoyées, vous pouvez spécifier ce que vous voulez à l'aide de l'option `--query`. Par exemple, vous pouvez obtenir le nom du paramètre, sa description et les valeurs autorisées pour le groupe de paramètres de cluster de bases de données Aurora PostgreSQL 12 par défaut comme suit :

Pour Linux, macOS ou Unix :

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12 \
    --query 'Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Pour Windows :

```
aws rds describe-db-cluster-parameters --db-cluster-parameter-group-name
 default.aurora-postgresql12 ^
    --query "Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Un groupe de paramètres de cluster de bases de données Aurora inclut le groupe de paramètres d'instance de base de données et les valeurs par défaut d'un moteur de base de données Aurora spécifique. Vous pouvez obtenir la liste des paramètres de base de données à partir du même groupe de paramètres Aurora PostgreSQL par défaut à l'aide de la commande describe-db-parameters de la AWS CLI, comme illustré ci-dessous.

Pour Linux, macOS ou Unix :

```
aws rds describe-db-parameters --db-parameter-group-name default.aurora-postgresql12 \
    --query 'Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Pour Windows :

```
aws rds describe-db-parameters --db-parameter-group-name default.aurora-postgresql12 ^
```

```
    --query "Parameters[].
[{ParameterName:ParameterName,Description:Description,ApplyType:ApplyType,AllowedValues:Allowed
```

Les commandes précédentes renvoient des listes de paramètres du cluster de bases de données ou du groupe de paramètres de base de données avec des descriptions et d'autres détails spécifiés dans la requête. Voici un exemple de réponse.

```
[
    [
        {
            "ParameterName": "apg_enable_batch_mode_function_execution",
            "ApplyType": "dynamic",
            "Description": "Enables batch-mode functions to process sets of rows at a
 time.",
            "AllowedValues": "0,1"
        }
    ],
    [
        {
            "ParameterName": "apg_enable_correlated_any_transform",
            "ApplyType": "dynamic",
            "Description": "Enables the planner to transform correlated ANY Sublink
 (IN/NOT IN subquery) to JOIN when possible.",
            "AllowedValues": "0,1"
        }
    ],...
```

Vous trouverez ci-dessous des tableaux contenant les valeurs du paramètre de cluster de bases de données par défaut et du paramètre de base de données pour Aurora PostgreSQL version 14.

Paramètres de niveau cluster d'Aurora PostgreSQL

Vous pouvez consulter les paramètres de niveau cluster pour une version d'Aurora PostgreSQL spécifique à l'aide de la console de gestion AWS, de l'interface de ligne de commande AWS ou de l'API Amazon RDS. Pour obtenir des informations sur l'affichage des paramètres dans des groupes de paramètres de cluster de bases de données Aurora PostgreSQL dans la console RDS, consultez Affichage des valeurs de paramètres pour un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

Certains paramètres de niveau cluster ne sont pas disponibles dans toutes les versions et d'autres sont obsolètes. Pour obtenir des informations sur l'affichage des paramètres d'une version spécifique d'Aurora PostgreSQL, consultez Affichage des paramètres de cluster de bases de données Aurora PostgreSQL et de base de données.

Par exemple, le tableau suivant répertorie les paramètres disponibles dans le groupe de paramètres de cluster de bases de données par défaut pour Aurora PostgreSQL version 14. Si vous créez un cluster de bases de données Aurora PostgreSQL sans spécifier votre propre groupe de paramètres de base de données personnalisé, votre cluster de bases de données est créé à l'aide du groupe de paramètres de cluster de bases de données Aurora par défaut pour la version choisie, par exemple `default.aurora-postgresql14`, `default.aurora-postgresql13`, etc.

Pour obtenir une liste des paramètres d'instance de la base de données pour ce même groupe de paramètres de cluster de bases de données par défaut, consultez Paramètres de niveau instance d'Aurora PostgreSQL.

Nom du paramètre	Description	Par défaut
ansi_constraint_trigger_ordering	Modifiez l'ordre de déclenchement des déclencheurs de contrainte pour qu'ils soient compatibles avec la norme ANSI SQL.	–
ansi_force_foreign_key_checks	Assurez-vous que les actions référentielles telles que la suppression en cascade ou la mise à jour en cascade se produisent toujours, quels que soient les contextes de déclencheur existants pour l'action.	–
ansi_qualified_update_set_target	Qualificateurs de table et de schéma de support dans les instructions UPDATE... SET.	–

Nom du paramètre	Description	Par défaut
apg_ccm_enabled	Activez ou désactivez la gestion des caches de clusters pour le cluster.	–
apg_enable_batch_mode_function_execution	Permet aux fonctions en mode traitement par lots de traiter plusieurs ensembles de lignes à la fois.	–
apg_enable_correlated_any_transform	Permet au planificateur de transformer le sous-lien ANY corrélé (sous-requête IN/NOT IN) en JOIN lorsque c'est possible.	–
apg_enable_function_migration	Permet au planificateur de migrer les fonctions scalaires éligibles vers la clause FROM.	–
apg_enable_not_in_transform	Permet au planificateur de transformer la sous-requête NOT IN en ANTI JOIN lorsque c'est possible.	–
apg_enable_remove_redundant_inner_joins	Permet au planificateur de supprimer les jointures internes redondantes.	–
apg_enable_semijoin_push_down	Permet l'utilisation de filtres de semi-jointure pour les jointures de hachage.	–
apg_plan_mgmt.capture_plan_baselines	Mode capture de référence de plans. manuel – active la capture de plans pour n'importe quelle instruction SQL, désactivé – désactive la capture de plans, automatique – active la capture de plans pour les instructions dans pg_stat_statements qui satisfont aux critères d'éligibilité.	off
apg_plan_mgmt.max_databases	Définit le nombre maximal de bases de données pouvant gérer des requêtes à l'aide de apg_plan_mgmt.	10

Nom du paramètre	Description	Par défaut
apg_plan_mgmt.max_plans	Définit le nombre maximal de plans pouvant être mis en cache par apg_plan_mgmt.	10 000
apg_plan_mgmt.plan_retention_period	Nombre maximal de jours écoulés depuis qu'un plan a été utilisé avant qu'un plan soit automatiquement supprimé.	32
apg_plan_mgmt.unapproved_plan_execution_threshold	Coût total estimé du plan en dessous duquel un plan non approuvé sera exécuté.	0
apg_plan_mgmt.use_plan_baselines	Utilisez uniquement des plans approuvés ou fixes pour les instructions gérées.	false
application_name	Définit le nom de l'application à indiquer dans les statistiques et les journaux.	–
array_nulls	Autorisez l'entrée d'éléments NULL dans les tableaux.	–
aurora_compute_plan_id	Surveille les plans d'exécution des requêtes pour détecter ceux qui contribuent à la charge actuelle de la base de données et pour suivre leurs statistiques de performance au fil du temps. Pour plus d'informations, consultez Surveillance des plans d'exécution des requêtes pour Aurora PostgreSQL.	on
authentication_timeout	(s) Définit le délai maximum autorisé pour procéder à l'authentification du client.	–
auto_explain.log_analyze	Utilisez EXPLAIN ANALYZE pour la journalisation des plans.	–
auto_explain.log_buffers	Utilisation des tampons de journaux.	–

Nom du paramètre	Description	Par défaut
auto_explain.log_format	Format EXPLAIN à utiliser pour la journalisation des plans.	–
auto_explain.log_min_duration	Définit la durée minimum d'exécution au-delà de laquelle les plans seront journalisés.	–
auto_explain.log_nested_statements	Journalisez les instructions imbriquées.	–
auto_explain.log_timing	Collectez des données temporelles et non uniquement le nombre de lignes.	–
auto_explain.log_triggers	Incluez des statistiques de déclenchement dans les plans.	–
auto_explain.log_verbose	Utilisez EXPLAIN VERBOSE pour la journalisation des plans.	–
auto_explain.sample_rate	Partie de requêtes à traiter.	–
autovacuum	Démarre le sous-processus autovacuum.	–
autovacuum_analyze_scale_factor	Nombre d'insertions, de mises à jour ou de suppressions de tuples avant l'analyse en tant que partie de reltuples.	0,05
autovacuum_analyze_threshold	Nombre minimum d'insertions de tuples mis à jour ou supprimés avant l'analyse.	–
autovacuum_freeze_max_age	Âge auquel lancer le processus autovacuum sur une table pour empêcher le bouclage de l'ID de transaction.	–
autovacuum_max_workers	Définit le nombre maximum de processus de travail autovacuum qui peuvent être exécutés simultanément.	GREATEST(DBInstanceClassMemory/64371566592,3)

Nom du paramètre	Description	Par défaut
autovacuum_multixact_freeze_max_age	Âge multixact auquel lancer le processus autovacuum sur une table pour empêcher le processus multixact de bouclage.	–
autovacuum_naptime	(s) Temps de repos entre les exécutions autovacuum.	5
autovacuum_vacuum_cost_delay	(ms) Valeur du coût de retard du processus vacuum en millisecondes, pour le processus autovacuum.	5
autovacuum_vacuum_cost_limit	Coût cumulé qui provoque l'endormissement du processus vacuum, pour le processus autovacuum.	GREATEST(log(DBInstanceClassMemory/21474836480)*600,200)
autovacuum_vacuum_insert_scale_factor	Nombre d'insertions de tuples avant le processus vacuum en tant que partie de reltuples.	–
autovacuum_vacuum_insert_threshold	Nombre minimum d'insertions de tuples avant le processus vacuum ou -1 pour désactiver les insertions de vacuum.	–
autovacuum_vacuum_scale_factor	Nombre de tuples mis à jour ou supprimés avant le processus vacuum en tant que partie de reltuples.	0.1
autovacuum_vacuum_threshold	Nombre de tuples mis à jour ou supprimés avant le processus vacuum.	–
autovacuum_work_mem	(kB) Définit la quantité maximum de mémoire que peut utiliser chaque processus de travail autovacuum.	GREATEST(DBInstanceClassMemory/32768,131072)
babelfishpg_tds.default_server_name	Nom par défaut du serveur Babelfish	Microsoft SQL Server

Nom du paramètre	Description	Par défaut
babelfishpg_tds.listen_addresses	Définit le nom d'hôte ou les adresses IP sur lesquelles écouter TDS.	*
babelfishpg_tds.port	Définit le port TCP TDS sur lequel le serveur écoute.	1433
babelfishpg_tds.tds_debug_log_level	Définit le niveau de journalisation dans TDS, 0 désactive la journalisation	1
babelfishpg_tds.tds_default_numeric_precision	Définit la précision par défaut du type numérique à envoyer dans les métadonnées de la colonne TDS si le moteur n'en spécifie pas.	38
babelfishpg_tds.tds_default_numeric_scale	Définit l'échelle par défaut du type numérique à envoyer dans les métadonnées de la colonne TDS si le moteur n'en spécifie pas.	8
babelfishpg_tds.tds_default_packet_size	Définit la taille par défaut des paquets pour tous les clients SQL Server connectés	4096
babelfishpg_tds.tds_default_protocol_version	Définit une version de protocole TDS par défaut pour tous les clients connectés	DEFAULT
babelfishpg_tds.tds_ssl_encrypt	Définit l'option de chiffrement SSL	0
babelfishpg_tds.tds_ssl_max_protocol_version	Définit la version maximale du protocole SSL/TLS à utiliser pour la session tds.	TLSv1.2
babelfishpg_tds.tds_ssl_min_protocol_version	Définit la version minimale du protocole SSL/TLS à utiliser pour la session tds.	TLSv1.2 à partir d'Aurora PostgreSQL 16, TLSv1 pour les versions antérieures à Aurora PostgreSQL 16

Nom du paramètre	Description	Par défaut
babelfishpg_tsql.default_locale	Paramètres régionaux par défaut à utiliser pour les classements créés par CREATE COLLATION.	en-US
babelfishpg_tsql.migration_mode	Détermine si plusieurs bases de données utilisateur sont prises en charge.	multi-db à partir d'Aurora PostgreSQL 16, single-db pour les versions antérieures à Aurora PostgreSQL 16
babelfishpg_tsql.server_collation_name	Nom du classement de serveur par défaut	sql_latin1_general_cp1_ci_as
babelfishpg_tsql.version	Définit la sortie de la variable @@VERSION	default
backend_flush_after	(8Kb) Nombre de pages après lesquelles les écritures précédemment effectuées sont vidées sur le disque.	–
backslash_quote	Définit si \\ peut être utilisée dans les littéraux de chaîne.	–
backtrace_functions	Enregistre le retour sur trace pour détecter les erreurs dans ces fonctions.	–
bytea_output	Définit le format de sortie pour les valeurs de type octets.	–
check_function_bodies	Vérifie les corps des fonctions pendant la fonction CREATE FUNCTION.	–
client_connection_check_interval	Définit l'intervalle de temps entre les vérifications de déconnexion lors de l'exécution des requêtes.	–

Nom du paramètre	Description	Par défaut
client_encoding	Définit l'encodage du jeu de caractères des clients.	UTF8
client_min_messages	Définit les niveaux des messages envoyés au client.	–
compute_query_id	Calcule les identifiants de requêtes.	auto
config_file	Définit le fichier de configuration principal du serveur.	/rdsdbdata/config/postgresql.conf
constraint_exclusion	Autorise le planificateur à utiliser des contraint es pour optimiser les requêtes.	–
cpu_index_tuple_cost	Définit l'estimation faite par le planificateur du coût de traitement de chaque entrée d'index pendant la vérification d'un index.	–
cpu_operator_cost	Définit l'estimation faite par le planificateur du coût de traitement de chaque opérateur ou appel de fonction.	–
cpu_tuple_cost	Définit l'estimation faite par le planificateur du coût de traitement de chaque tuple (ligne).	–
cron.database_name	Définit la base de données pour stocker les tables de métadonnées pg_cron	postgres
cron.log_run	Journaliser toutes les exécutions de tâches dans la table job_run_details	on
cron.log_statement	Consignez toutes les instructions cron avant exécution.	off
cron.max_running_j obs	Nombre maximal de tâches pouvant être exécutées simultanément.	5

Nom du paramètre	Description	Par défaut
cron.use_background_workers	Active les applications de travail en arrière-plan pour pg_cron	on
cursor_tuple_fraction	Définit l'estimation faite par le planificateur de la fraction des lignes d'un curseur qui sera récupérée.	–
data_directory	Définit le répertoire de données du serveur.	/rdsdbdata/db
datestyle	Définit le format d'affichage des valeurs de type date et heure.	–
db_user_namespace	Active les noms d'utilisateurs par base de données.	–
deadlock_timeout	(ms) Définit le délai d'attente au niveau d'un verrou avant blocage.	–
debug_pretty_print	Indente les affichages des arborescences d'analyse et de planification.	–
debug_print_parse	Journalise l'arborescence d'analyse de chaque requête.	–
debug_print_plan	Journalise le plan d'exécution de chaque requête.	–
debug_print_rewritten	Journalise l'arbre d'interprétation réécrit de chaque requête.	–
default_statistics_target	Définit la cible des statistiques par défaut.	–
default_tablespace	Définit l'espace de table par défaut dans lequel créer des tables et des index.	–
default_toast_compression	Définit la méthode de compression par défaut pour les valeurs compressibles.	–

Nom du paramètre	Description	Par défaut
default_transaction_deferrable	Définit le statut reportable des nouvelles transactions.	–
default_transaction_isolation	Définit le niveau d'isolement de transaction de chaque nouvelle transaction.	–
default_transaction_read_only	Définit le statut en lecture seule des nouvelles transactions.	–
effective_cache_size	(8kB) Définit l'estimation faite par le planificateur de la taille du cache du disque.	SUM(DBInstanceClassMemory/12038,-50003)
effective_io_concurrency	Nombre de demandes simultanées pouvant être traitées de manière efficace par le sous-système du disque.	–
enable_async_append	Active l'utilisation de plans d'ajout asynchrones par le planificateur.	–
enable_bitmapscan	Active l'utilisation de plans de parcours de bitmap par le planificateur.	–
enable_gathermerge	Active l'utilisation de plans de fusions de collecte par le planificateur.	–
enable_hashagg	Active l'utilisation de plans d'agrégation hachée par le planificateur.	–
enable_hashjoin	Active l'utilisation de plans de jointures de hachage par le planificateur.	–
enable_incremental_sort	Active l'utilisation d'étapes incrémentielles de tri par le planificateur.	–
enable_indexonlyscan	Active l'utilisation de plans de parcours d'index uniquement par le planificateur.	–

Nom du paramètre	Description	Par défaut
enable_indexscan	Active l'utilisation de plans de parcours d'index par le planificateur.	–
enable_material	Active l'utilisation de la matérialisation par le planificateur.	–
enable_memoize	Active l'utilisation de la mémorisation par le planificateur.	–
enable_mergejoin	Active l'utilisation de plans de jointures de fusion par le planificateur.	–
enable_nestloop	Active l'utilisation de plans de jointures de boucles imbriquées par le planificateur.	–
enable_parallel_append	Active l'utilisation de plans d'ajout parallèles par le planificateur.	–
enable_parallel_hash	Active l'utilisation de plans de hachage parallèles par le planificateur.	–
enable_partition_pruning	Active l'élagage des partitions de planification et d'exécution.	–
enable_partitionwise_aggregate	Active l'agrégation et le regroupement entre partitions.	–
enable_partitionwise_join	Active la jointure entre partitions.	–
enable_seqscan	Active l'utilisation de plans de parcours séquentiels par le planificateur.	–
enable_sort	Active l'utilisation des étapes de tri explicite par le planificateur.	–
enable_tidscan	Active l'utilisation de plans de parcours de TID par le planificateur.	–

Nom du paramètre	Description	Par défaut
escape_string_warning	Avertit sur l'utilisation des barres obliques inverses dans des littéraux de chaîne ordinaires.	–
exit_on_error	Résilie la session en cas d'erreur.	–
extra_float_digits	Définit le nombre de chiffres affichés pour les valeurs à virgule flottante.	–
force_parallel_mode	Force l'utilisation d'installations de requêtes parallèles.	–
from_collapse_limit	Définit la taille FROM-list au-delà de laquelle les sous-requêtes ne sont pas regroupées.	–
geqo	Active l'optimisation génétique des requêtes.	–
geqo_effort	geqo_effort est utilisé pour définir la valeur par défaut pour les autres paramètres GEQO.	–
geqo_generations	GEQO : nombre d'itérations de l'algorithme.	–
geqo_pool_size	GEQO : nombre d'individus au sein d'une population.	–
geqo_seed	GEQO : valeur initiale pour la sélection des chemins au hasard.	–
geqo_selection_bias	GEQO : pression de sélectivité au sein de la population.	–
geqo_threshold	Définit le seuil d'éléments FROM au-delà duquel GEQO est utilisé.	–
gin_fuzzy_search_limit	Définit le résultat maximum autorisé pour la recherche exacte par GIN.	–

Nom du paramètre	Description	Par défaut
gin_pending_list_limit	(kB) Définit la taille maximale de la liste en attente pour l'index GIN.	–
hash_mem_multiplier	Multiple de work_mem à utiliser pour les tables de hachage.	–
hba_file	Définit le fichier de configuration hba du serveur.	/rdsdbdata/config/pg_hba.conf
hot_standby_feedback	Autorise le commentaire d'une instance de secours vers l'instance principale, ce qui évitera les conflits de requêtes.	on
huge_pages	Réduit la surcharge lorsqu'une instance de base de données fonctionne avec de gros morceaux de mémoire contigus, tels que ceux utilisés par les tampons partagés. Le paramètre est activé par défaut pour toutes les classes d'instance de base de données autres que les classes d'instance t3.medium, db.t3.large, db.t4g.medium, db.t4g.large.	on
ident_file	Définit le fichier de configuration ident du serveur.	/rdsdbdata/config/pg_ident.conf
idle_in_transaction_session_timeout	(ms) Définit la durée maximale de toute transaction inactive.	86400000
idle_session_timeout	Interrompt toute session qui est restée inactive (c'est-à-dire en attente d'une requête du client), mais qui ne figure pas dans le cadre d'une transaction ouverte, pendant plus longtemps que la durée spécifiée	–
intervalstyle	Définit le format d'affichage des valeurs de type intervalle.	–

Nom du paramètre	Description	Par défaut
join_collapse_limit	Définit la taille FROM-list au-delà de laquelle les constructions JOIN ne sont pas mises à plat.	–
krb_caseins_users	Définit si les noms d'utilisateur GSSAPI (Generic Security Service API) doivent être traités sans tenir compte de la casse (true) ou non. Par défaut, ce paramètre est défini sur false, de sorte que Kerberos s'attend à ce que les noms d'utilisateur soient sensibles à la casse. Pour plus d'informations, consultez GSSAPI Authentication (Authentification GSSAPI) dans la documentation de PostgreSQL.	false
lc_messages	Définit la langue d'affichage des messages.	–
lc_monetary	Définit la locale à utiliser pour le formatage des montants monétaires.	–
lc_numeric	Définit la locale à utiliser pour le formatage des nombres.	–
lc_time	Définit la locale à utiliser pour le formatage des valeurs de date et d'heure.	–
listen_addresses	Définit le nom d'hôte ou les adresses IP à écouter.	*
lo_compat_privileges	Active le mode de compatibilité descendante pour les vérifications de privilèges sur de larges objets.	0
log_autovacuum_min_duration	(ms) Définit la durée minimum d'exécution au-delà de laquelle les actions autovacuum seront journalisées.	10 000

Nom du paramètre	Description	Par défaut
log_connections	Enregistre toutes les connexions réussies.	–
log_destination	Définit la destination pour la sortie du journal de serveur.	stderr
log_directory	Définit le répertoire de destination des fichiers journaux.	/rdsdbdata/log/error
log_disconnections	Enregistre la fin d'une session, y compris sa durée.	–
log_duration	Enregistre la durée de chaque instruction SQL terminée.	–
log_error_verbosity	Définit la quantité de détails dans les messages enregistrés.	–
log_executor_stats	Ecrit les statistiques de performance de l'exécuteur dans le journal du serveur.	–
log_file_mode	Définit les autorisations de fichier pour les fichiers journaux.	0644
log_filename	Définit le modèle de nom de fichier pour les fichiers journaux.	postgresql.log.%Y-%m-%d-%H%M
logging_collector	Démarrez un sous-processus pour capturer la sortie stderr et/ou csvlogs dans des fichiers journaux.	1
log_hostname	Enregistre le nom de l'hôte dans les journaux de connexion.	0
logical_decoding_work_mem	(kB) Cette quantité de mémoire peut être utilisée par chaque tampon interne de réorganisation avant le déversement sur le disque.	–

Nom du paramètre	Description	Par défaut
log_line_prefix	Contrôle les informations préfixées à chaque ligne de journal.	%t:%r:%u@%d:%p]:
log_lock_waits	Enregistre les longs temps d'attente pour l'acquisition d'un verrou.	–
log_min_duration_sample	(ms) Définit la durée minimum d'exécution au-delà de laquelle les instructions seront journalisées. L'échantillonnage est déterminé par log_statement_sample_rate.	–
log_min_duration_statement	(ms) Définit la durée minimum d'exécution au-delà de laquelle les instructions seront journalisées.	–
log_min_error_statement	Déclenche l'enregistrement de toutes les instructions générant une erreur à ce niveau ou à un niveau supérieur.	–
log_min_messages	Définit les niveaux des messages qui sont enregistrés.	–
log_parameter_max_length	(B) Lors de la journalisation des instructions, limitez les valeurs des paramètres enregistrés aux N premiers octets.	–
log_parameter_max_length_on_error	(B) Lorsque vous signalez une erreur, limitez les valeurs des paramètres enregistrés aux N premiers octets.	–
log_parser_stats	Ecrit les statistiques de performance de l'analyseur dans le journal du serveur.	–
log_planner_stats	Ecrit les statistiques de performance du planificateur dans le journal du serveur.	–

Nom du paramètre	Description	Par défaut
log_replication_commands	Journalise chaque commande de réplication.	–
log_rotation_age	(min) Déclenchement de la rotation de fichier journal automatique au-delà d'un délai de N minutes.	60
log_rotation_size	(kB) Déclenchement de la rotation de fichier journal automatique au-delà de N kilo-octets.	100 000
log_statement	Définit le type d'instructions enregistrées.	–
log_statement_sample_rate	Partie d'instructions dépassant log_min_duration_sample à journaliser.	–
log_statement_stats	Ecrit les statistiques de performance cumulées dans le journal du serveur.	–
log_temp_files	(kB) Journalise l'utilisation des fichiers temporaires dont la taille est supérieure à cette taille en kilo-octets.	–
log_timezone	Définit le fuseau horaire à utiliser dans les messages de journaux.	UTC
log_transaction_sample_rate	Définissez la partie des transactions à journaliser pour les nouvelles transactions.	–
log_truncate_on_rotation	Tronquez les fichiers journaux existants du même nom pendant la rotation des journaux.	0
maintenance_io_concurrency	Variante de effective_io_concurrency utilisée pour les tâches de maintenance.	1

Nom du paramètre	Description	Par défaut
maintenance_work_mem	(kB) Définit la quantité maximum de mémoire que peuvent utiliser les opérations de maintenance.	GREATEST(DBInstanceClassMemory/63963136*1024, 65536)
max_connections	Définit le nombre maximum de connexions simultanées.	LEAST(DBInstanceClassMemory/9531392, 5000)
max_files_per_process	Définit le nombre maximum de fichiers ouverts simultanément pour chaque processus serveur.	–
max_locks_per_transaction	Définit le nombre maximum de verrous par transaction.	64
max_logical_replication_workers	Nombre maximal de processus de travail de réplication logique.	–
max_parallel_maintenance_workers	Définit le nombre maximal de processus parallèles par opération de maintenance.	–
max_parallel_workers	Définit le nombre maximal d'applications de travail parallèles pouvant être actives simultanément.	GREATEST($DBInstanceVCPU/2, 8)
max_parallel_workers_per_gather	Définit le nombre maximal de processus parallèles par nœud d'exécuteur.	–
max_pred_locks_per_page	Définit le nombre maximal de tuples verrouillés par prédicat par page.	–
max_pred_locks_per_relation	Définit le nombre maximum de pages et de tuples verrouillés par prédicat par relation.	–
max_pred_locks_per_transaction	Définit le nombre maximum de verrous de prédicat par transaction.	–

Nom du paramètre	Description	Par défaut
max_prepared_transactions	Définit le nombre maximum de transactions préparées simultanément.	0
max_replication_slots	Définit le nombre maximum d'emplacements de réplication que le serveur peut prendre en charge.	20
max_slot_wal_keep_size	(MB) Les emplacements de réplication seront marqués comme ayant échoué et les segments seront libérés pour suppression ou recyclage si cette quantité d'espace est occupée par WAL sur le disque.	–
max_stack_depth	(kB) Définit la profondeur maximum de la pile, en kilo-octets.	6144
max_standby_streaming_delay	(ms) Définit le délai maximum avant l'annulation des requêtes lorsqu'un serveur de secours traite des données WAL diffusées.	14000
max_sync_workers_per_subscription	Nombre maximum d'applications de travail de synchronisation par abonnement	2
max_wal_senders	Définit le nombre maximum de processus d'expéditeur WAL qui peuvent être exécutés simultanément.	10
max_worker_processes	Définit le nombre maximum de processus de travail simultanés.	GREATEST($DBInstanceVCPU*2, 8)
min_dynamic_shared_memory	(Mo) Quantité de mémoire partagée dynamique réservée au démarrage.	–
min_parallel_index_scan_size	(8kB) Définit la quantité minimum de données d'index pour une analyse parallèle.	–

Nom du paramètre	Description	Par défaut
min_parallel_table_scan_size	(8kB) Définit la quantité minimum de données de table pour une analyse parallèle.	–
old_snapshot_threshold	(min) Délai avant qu'un instantané soit trop ancien pour lire les pages modifiées après la prise de l'instantané.	–
orafce.nls_date_format	Émule le comportement de sortie des données d'Oracle.	–
orafce.timezone	Spécifie le fuseau horaire utilisé pour la fonction sysdate.	–
parallel_leader_participation	Contrôle si Gather et Gather Merge exécutent également des sous-plans.	–
parallel_setup_cost	Définit l'estimation du planificateur du coût de démarrage des processus de travail pour les requêtes parallèles.	–
parallel_tuple_cost	Définit l'estimation du planificateur du coût de transmission de chaque tuple (ligne) de l'application de travail vers le système dorsal principal.	–
password_encryption	Chiffrez les mots de passe.	–
pgaudit.log	Spécifie quelles classes d'instructions seront journalisées par la journalisation de l'audit de session.	–
pgaudit.log_catalog	Spécifie que la journalisation de session doit être activée dans le cas où toutes les relations d'une instruction se trouvent dans pg_catalog.	–
pgaudit.log_level	Spécifie le niveau de journal qui sera utilisé pour les entrées de journal.	–

Nom du paramètre	Description	Par défaut
pgaudit.log_parameter	Spécifie que la journalisation de l’audit doit inclure les paramètres transmis avec l’instruction.	–
pgaudit.log_relation	Spécifie si la journalisation de l’audit de session doit créer une entrée de journal distincte pour chaque relation (TABLE, VIEW, etc.) référencée dans une instruction SELECT ou DML.	–
pgaudit.log_statement_once	Spécifie si la journalisation inclura le texte de l’instruction et les paramètres avec la première entrée de journal pour une combinaison instruction/sous-instruction ou avec chaque entrée.	–
pgaudit.role	Spécifie le rôle principal à utiliser pour la journalisation de l’audit des objets.	–
pg_bigm.enable_recheck	Spécifie s’il faut effectuer une revérification qui est un processus interne de recherche en texte intégral.	on
pg_bigm.gin_key_limit	Spécifie le nombre maximum de 2 grammes du mot-clé de recherche à utiliser pour la recherche en texte intégral.	0
pg_bigm.last_update	Indique la dernière date de mise à jour du module pg_bigm.	22.11.2013
pg_bigm.similarity_limit	Spécifie le seuil minimal utilisé par la recherche de similarité.	0.3
pg_hint_plan.debug_print	Journalise les résultats de l’analyse des indices.	–

Nom du paramètre	Description	Par défaut
pg_hint_plan.enable_hint	Force le planificateur à utiliser les plans spécifiés dans le commentaire d'indice précédant la requête.	–
pg_hint_plan.enable_hint_table	Force le planificateur à ne pas obtenir d'indice à l'aide des recherches de table.	–
pg_hint_plan.message_level	Niveau de message des messages de débogage.	–
pg_hint_plan.parse_messages	Niveau de message des erreurs d'analyse.	–
pglogical.batch_inserts	Insertions de lots si possible	–
pglogical.conflict_log_level	Définit le niveau de journal utilisé pour la journalisation des conflits résolus.	–
pglogical.conflict_resolution	Définit la méthode utilisée pour la résolution des conflits résolubles.	–
pglogical.extra_connection_options	options de connexion à ajouter à toutes les connexions de nœuds de pairs	–
pglogical.synchronous_commit	valeur de validation synchrone spécifique pglogical	–
pglogical.use_spi	Utiliser une SPI au lieu d'une API de bas niveau pour appliquer les modifications	–
pgtle.clientauth_databases_to_skip	Liste des bases de données à ignorer pour la fonctionnalité clientauth.	–
pgtle.clientauth_db_name	Contrôle la base de données utilisée pour la fonctionnalité clientauth.	–

Nom du paramètre	Description	Par défaut
pgtle.clientauth_num_parallel_workers	Nombre d'applications de travail en arrière-plan utilisés pour la fonctionnalité clientauth.	–
pgtle.clientauth_users_to_skip	Liste des utilisateurs à ignorer pour la fonctionnalité clientauth.	–
pgtle.enable_clientauth	Active la fonctionnalité clientauth.	–
pgtle.passcheck_db_name	Définit la base de données utilisée pour la fonctionnalité de vérification de mot de passe à l'échelle du cluster.	–
pg_prewarm.autoprewarm	Démarre l'application de travail autoprewarm.	–
pg_prewarm.autoprewarm_interval	Définit l'intervalle entre les vidages de tampons partagés	–
pg_similarity.block_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.block_threshold	Définit le seuil utilisé par la fonction de similarité Block.	–
pg_similarity.block_tokenizer	Définit le créateur de jetons pour la fonction de similarité Block.	–
pg_similarity.cosine_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.cosine_threshold	Définit le seuil utilisé par la fonction de similarité Cosine.	–
pg_similarity.cosine_tokenizer	Définit le créateur de jetons pour la fonction de similarité Cosine.	–

Nom du paramètre	Description	Par défaut
pg_similarity.dice_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.dice_threshold	Définit le seuil utilisé par la mesure de similarité Dice.	–
pg_similarity.dice_tokenizer	Définit le créateur de jetons pour la mesure de similarité Dice.	–
pg_similarity.euclidean_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.euclidean_threshold	Définit le seuil utilisé par la mesure de similarité Euclidean.	–
pg_similarity.euclidean_tokenizer	Définit le créateur de jetons pour la mesure de similarité Euclidean.	–
pg_similarity.hamming_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.hamming_threshold	Définit le seuil utilisé par la mesure de similarité Block.	–
pg_similarity.jaccard_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.jaccard_threshold	Définit le seuil utilisé par la mesure de similarité Jaccard.	–
pg_similarity.jaccard_tokenizer	Définit le créateur de jetons pour la mesure de similarité Jaccard.	–
pg_similarity.jaro_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.jaro_threshold	Définit le seuil utilisé par la mesure de similarité Jaro.	–

Nom du paramètre	Description	Par défaut
pg_similarity.jaro winkler_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.jaro winkler_threshold	Définit le seuil utilisé par la mesure de similarité Jarowinkler.	–
pg_similarity.leve nshtein_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.leve nshtein_threshold	Définit le seuil utilisé par la mesure de similarité Levenshtein.	–
pg_similarity.matc hing_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.matc hing_threshold	Définit le seuil utilisé par la mesure de similarité Matching Coefficient.	–
pg_similarity.matc hing_tokenizer	Définit le créateur de jetons pour la mesure de similarité Matching Coefficient.	–
pg_similarity.mong eelkan_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.mong eelkan_threshold	Définit le seuil utilisé par la mesure de similarité Monge-Elkan.	–
pg_similarity.mong eelkan_tokenizer	Définit le créateur de jetons pour la mesure de similarité Monge-Elkan.	–
pg_similarity.nw_g ap_penalty	Définit la pénalité d’écart utilisée par la mesure de similarité Needleman-Wunsch.	–
pg_similarity.nw_i s_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.nw_t hreshold	Définit le seuil utilisé par la mesure de similarité Needleman-Wunsch.	–

Nom du paramètre	Description	Par défaut
pg_similarity.overlap_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.overlap_threshold	Définit le seuil utilisé par la mesure de similarité Overlap Coefficient.	–
pg_similarity.overlap_tokenizer	Définit le créateur de jetons pour la mesure de similarité Overlap Coefficient.	–
pg_similarity.qgram_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.qgram_threshold	Définit le seuil utilisé par la mesure de similarité Q-Gram.	–
pg_similarity.qgram_tokenizer	Définit le créateur de jetons pour la mesure Q-Gram.	–
pg_similarity.swg_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.swg_threshold	Définit le seuil utilisé par la mesure de similarité Smith-Waterman-Gotoh.	–
pg_similarity.sw_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.sw_threshold	Définit le seuil utilisé par la mesure de similarité Smith-Waterman.	–
pg_stat_statements.max	Définit le nombre maximal d’instructions suivies par pg_stat_statements.	–
pg_stat_statements.save	Enregistre les statistiques pg_stat_statements sur les arrêts de serveur.	–
pg_stat_statements.track	Définit les instructions qui sont suivies par pg_stat_statements.	–

Nom du paramètre	Description	Par défaut
pg_stat_statements.track_planning	Définit si le délai de planification est suivi par pg_stat_statements.	–
pg_stat_statements.track_utility	Définit si les commandes d'utilitaire sont suivies par pg_stat_statements.	–
plan_cache_mode	Contrôle la sélection du planificateur d'un plan personnalisé ou générique.	–
port	Définit le port TCP sur lequel le serveur écoute.	EndPointPort
postgis.gdal_enabled_drivers	Activez ou désactivez les pilotes GDAL utilisés avec PostGIS dans Postgres 9.3.5 et versions ultérieures.	ENABLE_ALL
quote_all_identifiers	Lors de la génération de fragments SQL, ajoute des guillemets à tous les identifiants.	–
random_page_cost	Définit l'estimation faite par le planificateur du coût d'une page de disque extraite de façon non séquentielle.	–
rdkit.dice_threshold	Seuil inférieur de similarité Dice. Les molécules dont la similarité est inférieure au seuil ne sont pas similaires par l'opération #.	–
rdkit.do_chiral_sss	Si la stéréochimie doit être prise en compte dans la correspondance des sous-structures. Si la valeur est fausse, aucune information de stéréochimie n'est utilisée dans les correspondances des sous-structures.	–
rdkit.tanimoto_threshold	Seuil inférieur de similitude avec Tanimoto. Les molécules dont la similarité est inférieure au seuil ne sont pas similaires par l'opération %.	–

Nom du paramètre	Description	Par défaut
rds.accepted_password_auth_method	Force l’authentification des connexions avec un mot de passe stocké localement.	md5+scram
rds.adaptive_autovacuum	Paramètre RDS pour activer/désactiver l’autovacuum adaptatif.	1
rds.babelfish_status	Paramètre RDS pour activer/désactiver Babelfish pour Aurora PostgreSQL.	off
rds.enable_plan_management	Activez ou désactivez l’extension apg_plan_mgmt.	0

Nom du paramètre	Description	Par défaut
rds.extensions	Liste des extensions fournies par RDS	address_standardizer, address_standardizer_data_us, apg_plan_mgmt, aurora_stat_utils, amcheck, autoinc, aws_commons, aws_ml, aws_s3, aws_lambda, bool_plperl, bloom, btree_gin, btree_gist, citext, cube, dblink, dict_int, dict_xsyn, earthdistance, fuzzystrmatch, hll, hstore, hstore_plperl, insert_username, intagg, intarray, ip4r, isn, jsonb_plperl, lo, log_fdw, ltree, moddatetime, old_snapshot, oracle_fdw, orafce, pgaudit, pgcrypto, pglogical, pgrouting, pgrowlocks, pgstattuple, pgtap, pg_bigm, pg_buffercache, pg_cron, pg_freespacemap, pg_hint_plan, pg_partman, pg_prewarm, pg_proctab, pg_repack, pg_simila

Nom du paramètre	Description	Par défaut
		rity, pg_stat_s tatements, pg_trgm, pg_visibility, plcoffee, plls, plperl, plpgsql, plprofiler, pltcl, plv8, postgis, postgis_t iger_geocoder, postgis_raster, postgis_topology, postgres_fdw, prefix, rdkit, rds_tools, refint, sslinfo, tablefunc, tds_fdw, test_parser, tsm_system_rows, tsm_system_time, unaccent, uuid-ossp
rds.force_admin_lo gging_level	Consultez les messages de journalisation des actions d'utilisateur de l'administrateur RDS dans les bases de données clients.	–
rds.force_autovacu um_logging_level	Consultez les messages de journal relatifs aux opérations d'autovacuum.	WARNING
rds.force_ssl	Forcez les connexions SSL.	0

Nom du paramètre	Description	Par défaut
rds.global_db_rpo	(s) Seuil d'objectif de point de reprise en secondes qui bloque les validations de l'utilisateur lorsqu'il est violé. **Important** Ce paramètre est destiné aux bases de données globales basées sur Aurora PostgreSQL. Pour une base de données non globale, conservez la valeur par défaut. Pour en savoir plus sur l'utilisation de ce paramètre, consultez the section called "Gestion des RPO (Aurora PostgreSQL)".	–
rds.logical_replication	Active le décodage logique.	0
rds.logically_replicate_unlogged_tables	Les tables non journalisées sont répliquées de manière logique.	1
rds.log_retention_period	Amazon RDS supprimera les journaux PostgreSQL antérieurs à N minutes.	4320
rds.pg_stat_ramdisk_size	Taille du disque RAM des statistiques en Mo. Une valeur différente de zéro va configurer le disque RAM. Ce paramètre est disponible uniquement dans Aurora PostgreSQL 14 et les versions antérieures.	0
rds.rds_superuser_reserved_connections	Définit le nombre d'emplacements de connexion réservés pour rds_superusers. Ce paramètre est uniquement disponible dans les versions 15 et antérieures. Pour plus d'informations, consultez la documentation de PostgreSQL sur reserved_connections.	2

Nom du paramètre	Description	Par défaut
rds.restrict_password_commands	restreint les commandes liées au mot de passe aux membres de rds_password	–
rds.superuser_variables	Liste des variables réservées aux super-utilisateurs pour lesquelles nous augmentons les instructions de modification rds_superuser.	session_replication_role
recovery_init_sync_method	Définit la méthode de synchronisation du répertoire de données avant la reprise après incident.	syncfs
remove_temp_files_after_crash	Supprime les fichiers temporaires après l'arrêt du système dorsal.	0
restart_after_crash	Réinitialisez le serveur après l'arrêt du système dorsal.	–
row_security	Activez la sécurité au niveau des lignes.	–
search_path	Définit l'ordre de recherche des schémas pour les noms pour lesquels le schéma n'est pas précisé.	–
seq_page_cost	Définit l'estimation faite par le planificateur du coût d'une page de disque extraite de façon séquentielle.	–
session_replication_role	Définit le comportement des sessions concernant les déclencheurs et les règles de réécriture.	–
shared_buffers	(8kB) Définit le nombre de tampons de mémoire partagée utilisés par le serveur.	SUM(DBInstanceClassMemory/12038,-50003)
shared_preload_libraries	Répertorie les bibliothèques partagées à précharger sur le serveur.	pg_stat_statements

Nom du paramètre	Description	Par défaut
ssl	Active les connexions SSL.	1
ssl_ca_file	Emplacement du fichier d'autorité du serveur SSL.	/rdsdbdata/rds-metadata/ca-cert.pem
ssl_cert_file	Emplacement du fichier de certificat du serveur SSL.	/rdsdbdata/rds-metadata/server-cert.pem
ssl_ciphers	Définit la liste des chiffrements TLS autorisés à utiliser sur les connexions sécurisées.	–
ssl_crl_dir	Emplacement du répertoire de la liste de révocation des certificats SSL.	/rdsdbdata/rds-metadata/ssl_crl_dir/
ssl_key_file	Emplacement du fichier de clé privée du serveur SSL	/rdsdbdata/rds-metadata/server-key.pem
ssl_max_protocol_version	Définit la version maximale autorisée du protocole SSL/TLS	–
ssl_min_protocol_version	Définit la version minimale du protocole SSL/TLS	TLSv1.2
standard_conforming_strings	Entraîne les chaînes ... à traiter littéralement les barres obliques inverses.	–
statement_timeout	(ms) Définit la durée maximum de toute instruction.	–
stats_temp_directory	Écrit des fichiers de statistiques temporaires dans le répertoire spécifié.	/rdsdbdata/db/pg_stat_tmp
superuser_reserved_connections	Définit le nombre d'emplacements de connexion réservés aux super-utilisateurs.	3
synchronize_seqscans	Active les analyses séquentielles synchronisées.	–

Nom du paramètre	Description	Par défaut
synchronous_commit	Définit le niveau de synchronisation des transactions actuelles.	on
tcp_keepalives_count	Nombre maximum de paquets TCP keepalive.	–
tcp_keepalives_idle	(s) Délai entre les émissions de paquets TCP keepalive.	–
tcp_keepalives_interval	(s) Délai entre les envois de paquets TCP keepalive.	–
temp_buffers	(8kB) Définit le nombre maximum de tampons temporaires utilisés par chaque session.	–
temp_file_limit	Contraint l’espace disque total en kilo-octets qu’un processus PostgreSQL donné peut utiliser pour les fichiers temporaires, à l’exclusion de l’espace utilisé pour les tables temporaires explicites	-1
temp_tablespaces	Définit les espaces de table à utiliser pour les tables et fichiers de tri temporaires.	–
timezone	Définit le fuseau horaire pour l’affichage et l’interprétation de la date et de l’heure.	UTC
track_activities	Active la collecte d’informations sur l’exécution des commandes.	–
track_activity_query_size	Définit la taille réservée pour pg_stat_activity.current_query, en octets.	4096
track_commit_timestamp	Collecte l’heure de validation des transactions.	–
track_counts	Active la collecte de statistiques sur l’activité de la base de données.	–

Nom du paramètre	Description	Par défaut
track_functions	Active la collecte de statistiques au niveau de la fonction sur l'activité de la base de données.	pl
track_io_timing	Active la collecte de statistiques de durée sur l'activité I/O de la base de données.	1
track_wal_io_timing	Collecte des statistiques de durée sur l'activité I/O de WAL.	–
transform_null_equals	Traite l'expression =NULL en tant que IS NULL.	–
update_process_title	Met à jour le titre du processus pour indiquer la commande SQL active.	–
vacuum_cost_delay	(ms) Valeur du coût de délai du processus vacuum en millisecondes.	–
vacuum_cost_limit	Coût cumulé qui provoque l'endormissement du processus vacuum.	–
vacuum_cost_page_dirty	Coût du processus vacuum pour une page salie par le processus vacuum.	–
vacuum_cost_page_hit	Coût du processus vacuum pour une page trouvée dans le cache des tampons.	–
vacuum_cost_page_miss	Coût du processus vacuum pour une page non trouvée dans le cache des tampons.	0
vacuum_defer_cleanup_age	Nombre de transactions pendant lesquelles le processus VACUUM et le nettoyage HOT seront reportés à plus tard, le cas échéant.	–
vacuum_failsafe_age	Âge auquel le processus VACUUM doit déclencher la sécurité pour éviter une panne complète.	1200000000

Nom du paramètre	Description	Par défaut
vacuum_freeze_min_age	Âge limite auquel le processus VACUUM doit figer une ligne de tableau.	–
vacuum_freeze_table_age	Âge auquel le processus VACUUM effectue une analyse complète de la table pour figer des lignes.	–
vacuum_multixact_failsafe_age	Âge Multixact auquel le processus VACUUM doit déclencher la sécurité pour éviter une panne complète.	1200000000
vacuum_multixact_freeze_min_age	Âge limite auquel le processus VACUUM doit figer un MultiXactId dans une ligne de tableau.	–
vacuum_multixact_freeze_table_age	Âge Multixact auquel le processus VACUUM effectue une analyse complète de la table pour figer des lignes.	–
wal_buffers	(8kB) Définit le nombre de tampons de page de disque dans la mémoire partagée pour WAL.	–
wal_receiver_create_temp_slot	Définit si un récepteur WAL doit créer un emplacement de réplication temporaire lorsqu'aucun emplacement permanent n'est configuré.	0
wal_receiver_status_interval	(s) Définit l'intervalle maximal entre les rapports d'état du récepteur WAL et le principal.	–
wal_receiver_timeout	(ms) Définit le délai d'attente maximal pour recevoir les données du principal.	30 000
wal_sender_timeout	(ms) Définit le délai d'attente maximal de la réplication WAL.	–

Nom du paramètre	Description	Par défaut
work_mem	(kB) Définit la quantité maximum de mémoire que peuvent utiliser les espaces de travail des requêtes.	–
xmlbinary	Définit la façon dont les valeurs binaires doivent être codées en XML.	–
xmloption	Définit si des données XML dans des opération s d’analyse ou de sérialisation implicite s doivent être considérées comme des documents ou des fragments de contenu.	–

Paramètres de niveau instance d'Aurora PostgreSQL

Vous pouvez afficher les paramètres de niveau d'instance pour une version d'Aurora PostgreSQL spécifique à l'aide de la console de gestion AWS, de l'interface de ligne de commande AWS ou de l'API Amazon RDS. Pour obtenir des informations sur l'affichage des paramètres dans des groupes de paramètres de base de données Aurora PostgreSQL dans la console RDS, consultez Affichage des valeurs de paramètres pour un groupe de paramètres de base de données dans Amazon Aurora.

Certains paramètres de niveau d'instance ne sont pas disponibles dans toutes les versions et d'autres sont obsolètes. Pour obtenir des informations sur l'affichage des paramètres d'une version spécifique d'Aurora PostgreSQL, consultez Affichage des paramètres de cluster de bases de données Aurora PostgreSQL et de base de données.

Par exemple, le tableau suivant répertorie les paramètres qui s'appliquent à une instance de base de données spécifique dans un cluster de bases de données Aurora PostgreSQL. Cette liste a été générée en exécutant la commande describe-db-parameters de la AWS CLI avec `default.aurora-postgresql14` pour la valeur `--db-parameter-group-name`.

Pour obtenir une liste des paramètres de cluster de bases de données pour ce même groupe de paramètres de base de données par défaut, consultez Paramètres de niveau cluster d'Aurora PostgreSQL.

Nom du paramètre	Description	Par défaut
apg_enable_batch_mode_function_execution	Permet aux fonctions en mode traitement par lots de traiter plusieurs ensembles de lignes à la fois.	–
apg_enable_correlated_any_transform	Permet au planificateur de transformer le sous-lien ANY corrélé (sous-requête IN/NOT IN) en JOIN lorsque c'est possible.	–
apg_enable_function_migration	Permet au planificateur de migrer les fonctions scalaires éligibles vers la clause FROM.	–
apg_enable_not_in_transform	Permet au planificateur de transformer la sous-requête NOT IN en ANTI JOIN lorsque c'est possible.	–

Nom du paramètre	Description	Par défaut
apg_enable_remove_redundant_inner_joins	Permet au planificateur de supprimer les jointures internes redondantes.	–
apg_enable_semijoin_push_down	Permet l'utilisation de filtres de semi-jointure pour les jointures de hachage.	–
apg_plan_mgmt.capture_plan_baselines	Mode capture de référence de plans. manuel – active la capture de plans pour n'importe quelle instruction SQL, désactivé – désactive la capture de plans, automatique – active la capture de plans pour les instructions dans pg_stat_statements qui satisfont aux critères d'éligibilité.	off
apg_plan_mgmt.max_databases	Définit le nombre maximal de bases de données pouvant gérer des requêtes à l'aide de apg_plan_mgmt.	10
apg_plan_mgmt.max_plans	Définit le nombre maximal de plans pouvant être mis en cache par apg_plan_mgmt.	10 000
apg_plan_mgmt.plan_retention_period	Nombre maximal de jours écoulés depuis qu'un plan a été utilisé avant qu'un plan soit automatiquement supprimé.	32
apg_plan_mgmt.unapproved_plan_execution_threshold	Coût total estimé du plan en dessous duquel un plan non approuvé sera exécuté.	0
apg_plan_mgmt.use_plan_baselines	Utilisez uniquement des plans approuvés ou fixes pour les instructions gérées.	false
application_name	Définit le nom de l'application à indiquer dans les statistiques et les journaux.	–

Nom du paramètre	Description	Par défaut
aurora_compute_plan_id	Surveille les plans d'exécution des requêtes pour détecter ceux qui contribuent à la charge actuelle de la base de données et pour suivre leurs statistiques de performance au fil du temps. Pour plus d'informations, consultez Surveillance des plans d'exécution des requêtes pour Aurora PostgreSQL.	on
aurora_temp_space_size	(Mo) Définit la taille de l'espace alloué aux objets temporaires compatibles avec Optimized Reads sur les clusters optimisés pour les E/S Aurora avec les classes d'instance prises en charge.	DBInstanceClassMemory/524288
authentication_timeout	(s) Définit le délai maximum autorisé pour procéder à l'authentification du client.	–
auto_explain.log_analyze	Utilisez EXPLAIN ANALYZE pour la journalisation des plans.	–
auto_explain.log_buffers	Utilisation des tampons de journaux.	–
auto_explain.log_format	Format EXPLAIN à utiliser pour la journalisation des plans.	–
auto_explain.log_min_duration	Définit la durée minimum d'exécution au-delà de laquelle les plans seront journalisés.	–
auto_explain.log_nested_statements	Journalisez les instructions imbriquées.	–
auto_explain.log_timing	Collectez des données temporelles et non uniquement le nombre de lignes.	–
auto_explain.log_triggers	Incluez des statistiques de déclenchement dans les plans.	–

Nom du paramètre	Description	Par défaut
auto_explain.log_verbose	Utilisez EXPLAIN VERBOSE pour la journalisation des plans.	–
auto_explain.sample_rate	Partie de requêtes à traiter.	–
babelfishpg_tds.listen_addresses	Définit le nom d'hôte ou les adresses IP sur lesquelles écouter TDS.	*
babelfishpg_tds.tds_debug_log_level	Définit le niveau de journalisation dans TDS, 0 désactive la journalisation	1
backend_flush_after	(8kB) Nombre de pages après lesquelles les écritures précédemment effectuées sont vidées sur le disque.	–
bytea_output	Définit le format de sortie pour les valeurs de type octets.	–
check_function_bodies	Vérifie les corps des fonctions pendant la fonction CREATE FUNCTION.	–
client_connection_check_interval	Définit l'intervalle de temps entre les vérifications de déconnexion lors de l'exécution des requêtes.	–
client_min_messages	Définit les niveaux des messages envoyés au client.	–
config_file	Définit le fichier de configuration principal du serveur.	/rdsdbdata/config/postgresql.conf
constraint_exclusion	Autorise le planificateur à utiliser des contraintes pour optimiser les requêtes.	–

Nom du paramètre	Description	Par défaut
cpu_index_tuple_cost	Définit l’estimation faite par le planificateur du coût de traitement de chaque entrée d’index pendant la vérification d’un index.	–
cpu_operator_cost	Définit l’estimation faite par le planificateur du coût de traitement de chaque opérateur ou appel de fonction.	–
cpu_tuple_cost	Définit l’estimation faite par le planificateur du coût de traitement de chaque tuple (ligne).	–
cron.database_name	Définit la base de données pour stocker les tables de métadonnées pg_cron	postgres
cron.log_run	Journaliser toutes les exécutions de tâches dans la table job_run_details	on
cron.log_statement	Consignez toutes les instructions cron avant exécution.	off
cron.max_running_jobs	Nombre maximal de tâches pouvant être exécutées simultanément.	5
cron.use_background_workers	Active les applications de travail en arrière-plan pour pg_cron	on
cursor_tuple_fraction	Définit l’estimation faite par le planificateur de la fraction des lignes d’un curseur qui sera récupérée.	–
db_user_namespace	Active les noms d’utilisateurs par base de données.	–
deadlock_timeout	(ms) Définit le délai d’attente au niveau d’un verrou avant blocage.	–

Nom du paramètre	Description	Par défaut
debug_pretty_print	Indente les affichages des arborescences d'analyse et de planification.	–
debug_print_parse	Journalise l'arborescence d'analyse de chaque requête.	–
debug_print_plan	Journalise le plan d'exécution de chaque requête.	–
debug_print_rewritten	Journalise l'arbre d'interprétation réécrit de chaque requête.	–
default_statistics_target	Définit la cible des statistiques par défaut.	–
default_transaction_deferrable	Définit le statut reportable des nouvelles transactions.	–
default_transaction_isolation	Définit le niveau d'isolement de transaction de chaque nouvelle transaction.	–
default_transaction_read_only	Définit le statut en lecture seule des nouvelles transactions.	–
effective_cache_size	(8kB) Définit l'estimation faite par le planificateur de la taille du cache du disque.	SUM(DBInstanceClassMemory/12038,-50003
effective_io_concurrency	Nombre de demandes simultanées pouvant être traitées de manière efficace par le sous-système du disque.	–
enable_async_append	Active l'utilisation de plans d'ajout asynchrones par le planificateur.	–
enable_bitmapscan	Active l'utilisation de plans de parcours de bitmap par le planificateur.	–

Nom du paramètre	Description	Par défaut
enable_gathermerge	Active l'utilisation de plans de fusions de collecte par le planificateur.	–
enable_hashagg	Active l'utilisation de plans d'agrégation hachée par le planificateur.	–
enable_hashjoin	Active l'utilisation de plans de jointures de hachage par le planificateur.	–
enable_incremental_sort	Active l'utilisation d'étapes incrémentielles de tri par le planificateur.	–
enable_indexonlyscan	Active l'utilisation de plans de parcours d'index uniquement par le planificateur.	–
enable_indexscan	Active l'utilisation de plans de parcours d'index par le planificateur.	–
enable_material	Active l'utilisation de la matérialisation par le planificateur.	–
enable_memoize	Active l'utilisation de la mémorisation par le planificateur.	–
enable_mergejoin	Active l'utilisation de plans de jointures de fusion par le planificateur.	–
enable_nestloop	Active l'utilisation de plans de jointures de boucles imbriquées par le planificateur.	–
enable_parallel_append	Active l'utilisation de plans d'ajout parallèles par le planificateur.	–
enable_parallel_hash	Active l'utilisation de plans de hachage parallèles par le planificateur.	–
enable_partition_pruning	Active l'élagage des partitions de planification et d'exécution.	–

Nom du paramètre	Description	Par défaut
enable_partitionwise_aggregate	Active l’agrégation et le regroupement entre partitions.	–
enable_partitionwise_join	Active la jointure entre partitions.	–
enable_seqscan	Active l’utilisation de plans de parcours séquentiels par le planificateur.	–
enable_sort	Active l’utilisation des étapes de tri explicite par le planificateur.	–
enable_tidscan	Active l’utilisation de plans de parcours de TID par le planificateur.	–
escape_string_warning	Avertit sur l’utilisation des barres obliques inverses dans des littéraux de chaîne ordinaires.	–
exit_on_error	Résilie la session en cas d’erreur.	–
force_parallel_mode	Force l’utilisation d’installations de requêtes parallèles.	–
from_collapse_limit	Définit la taille FROM-list au-delà de laquelle les sous-requêtes ne sont pas regroupées.	–
geqo	Active l’optimisation génétique des requêtes.	–
geqo_effort	geqo_effort est utilisé pour définir la valeur par défaut pour les autres paramètres GEQO.	–
geqo_generations	GEQO : nombre d’itérations de l’algorithme.	–
geqo_pool_size	GEQO : nombre d’individus au sein d’une population.	–

Nom du paramètre	Description	Par défaut
geqo_seed	GEQO : valeur initiale pour la sélection des chemins au hasard.	–
geqo_selection_bias	GEQO : pression de sélectivité au sein de la population.	–
geqo_threshold	Définit le seuil d'éléments FROM au-delà duquel GEQO est utilisé.	–
gin_fuzzy_search_limit	Définit le résultat maximum autorisé pour la recherche exacte par GIN.	–
gin_pending_list_limit	(kB) Définit la taille maximale de la liste en attente pour l'index GIN.	–
hash_mem_multiplier	Multiple de work_mem à utiliser pour les tables de hachage.	–
hba_file	Définit le fichier de configuration hba du serveur.	/rdsdbdata/config/pg_hba.conf
hot_standby_feedback	Autorise le commentaire d'une instance de secours vers l'instance principale, ce qui évitera les conflits de requêtes.	on
ident_file	Définit le fichier de configuration ident du serveur.	/rdsdbdata/config/pg_ident.conf
idle_in_transaction_session_timeout	(ms) Définit la durée maximale de toute transaction inactive.	86400000
idle_session_timeout	Interrompt toute session qui est restée inactive (c'est-à-dire en attente d'une requête du client), mais qui ne figure pas dans le cadre d'une transaction ouverte, pendant plus longtemps que la durée spécifiée	–

Nom du paramètre	Description	Par défaut
join_collapse_limit	Définit la taille FROM-list au-delà de laquelle les constructions JOIN ne sont pas mises à plat.	–
lc_messages	Définit la langue d'affichage des messages.	–
listen_addresses	Définit le nom d'hôte ou l'adresse IP à écouter.	*
lo_compat_privileges	Active le mode de compatibilité descendante pour les vérifications de privilèges sur de larges objets.	0
log_connections	Enregistre toutes les connexions réussies.	–
log_destination	Définit la destination pour la sortie du journal de serveur.	stderr
log_directory	Définit le répertoire de destination des fichiers journaux.	/rdsdbdata/log/error
log_disconnections	Enregistre la fin d'une session, y compris sa durée.	–
log_duration	Enregistre la durée de chaque instruction SQL terminée.	–
log_error_verbosity	Définit la quantité de détails dans les messages enregistrés.	–
log_executor_stats	Ecrit les statistiques de performance de l'exécuteur dans le journal du serveur.	–
log_file_mode	Définit les autorisations de fichier pour les fichiers journaux.	0644
log_filename	Définit le modèle de nom de fichier pour les fichiers journaux.	postgresql.log.%Y-%m-%d-%H%M

Nom du paramètre	Description	Par défaut
logging_collector	Démarrez un sous-processus pour capturer la sortie stderr et/ou csvlogs dans des fichiers journaux.	1
log_hostname	Enregistre le nom de l'hôte dans les journaux de connexion.	0
logical_decoding_work_mem	(kB) Cette quantité de mémoire peut être utilisée par chaque tampon interne de réorganisation avant le déversement sur le disque.	–
log_line_prefix	Contrôle les informations préfixées à chaque ligne de journal.	%t:%r:%u@%d:%p]:
log_lock_waits	Enregistre les longs temps d'attente pour l'acquisition d'un verrou.	–
log_min_duration_sample	(ms) Définit la durée minimum d'exécution au-delà de laquelle un exemple d'instructions sera journalisé. L'échantillonnage est déterminé par log_statement_sample_rate.	–
log_min_duration_statement	(ms) Définit la durée minimum d'exécution au-delà de laquelle les instructions seront journalisées.	–
log_min_error_statement	Déclenche l'enregistrement de toutes les instructions générant une erreur à ce niveau ou à un niveau supérieur.	–
log_min_messages	Définit les niveaux des messages qui sont enregistrés.	–
log_parameter_max_length	(B) Lors de la journalisation des instructions, limitez les valeurs des paramètres enregistrés aux N premiers octets.	–

Nom du paramètre	Description	Par défaut
log_parameter_max_length_on_error	(B) Lorsque vous signalez une erreur, limitez les valeurs des paramètres enregistrés aux N premiers octets.	–
log_parser_stats	Ecrit les statistiques de performance de l'analyseur dans le journal du serveur.	–
log_planner_stats	Ecrit les statistiques de performance du planificateur dans le journal du serveur.	–
log_replication_commands	Journalise chaque commande de réplication.	–
log_rotation_age	(min) Déclenchement de la rotation de fichier journal automatique au-delà d'un délai de N minutes.	60
log_rotation_size	(kB) Déclenchement de la rotation de fichier journal automatique au-delà de N kilo-octets.	100 000
log_statement	Définit le type d'instructions enregistrées.	–
log_statement_sample_rate	Partie d'instructions dépassant log_min_duration_sample à journaliser.	–
log_statement_stats	Ecrit les statistiques de performance cumulées dans le journal du serveur.	–
log_temp_files	(kB) Journalise l'utilisation des fichiers temporaires dont la taille est supérieure à cette taille en kilo-octets.	–
log_timezone	Définit le fuseau horaire à utiliser dans les messages de journaux.	UTC
log_truncate_on_rotation	Tronquez les fichiers journaux existants du même nom pendant la rotation des journaux.	0

Nom du paramètre	Description	Par défaut
maintenance_io_concurrency	Variante de effective_io_concurrency utilisée pour les tâches de maintenance.	1
maintenance_work_mem	(kB) Définit la quantité maximum de mémoire que peuvent utiliser les opérations de maintenance.	GREATEST(DBInstanceClassMemory/63963136*1024, 65536
max_connections	Définit le nombre maximum de connexions simultanées.	LEAST(DBInstanceClassMemory/9531392, 5000
max_files_per_process	Définit le nombre maximum de fichiers ouverts simultanément pour chaque processus serveur.	–
max_locks_per_transaction	Définit le nombre maximum de verrous par transaction.	64
max_parallel_maintenance_workers	Définit le nombre maximal de processus parallèles par opération de maintenance.	–
max_parallel_workers	Définit le nombre maximal d'applications de travail parallèles pouvant être actives simultanément.	GREATEST($DBInstanceVCPU/2, 8
max_parallel_workers_per_gather	Définit le nombre maximal de processus parallèles par nœud d'exécuteur.	–
max_pred_locks_per_page	Définit le nombre maximal de tuples verrouillés par prédicat par page.	–
max_pred_locks_per_relation	Définit le nombre maximum de pages et de tuples verrouillés par prédicat par relation.	–
max_pred_locks_per_transaction	Définit le nombre maximum de verrous de prédicat par transaction.	–

Nom du paramètre	Description	Par défaut
max_slot_wal_keep_size	(MB) Les emplacements de réplication seront marqués comme ayant échoué et les segments seront libérés à des fins de suppression ou de recyclage si cette quantité d'espace est occupée par WAL sur le disque.	–
max_stack_depth	(kB) Définit la profondeur maximum de la pile, en kilo-octets.	6144
max_standby_streaming_delay	(ms) Définit le délai maximum avant l'annulation des requêtes lorsqu'un serveur en zone hébergée traite des données WAL diffusées.	14000
max_worker_processes	Définit le nombre maximum de processus de travail simultanés.	GREATEST($DBInstanceVCPU*2, 8
min_dynamic_shared_memory	(MB) Quantité de mémoire partagée dynamique réservée au démarrage.	–
min_parallel_index_scan_size	(8kB) Définit la quantité minimum de données d'index pour une analyse parallèle.	–
min_parallel_table_scan_size	(8kB) Définit la quantité minimum de données de table pour une analyse parallèle.	–
old_snapshot_threshold	(min) Délai avant qu'un instantané soit trop ancien pour lire les pages modifiées après la prise de l'instantané.	–
parallel_leader_participation	Contrôle si Gather et Gather Merge exécutent également des sous-plans.	–
parallel_setup_cost	Définit l'estimation du planificateur du coût de démarrage des processus de travail pour les requêtes parallèles.	–

Nom du paramètre	Description	Par défaut
parallel_tuple_cost	Définit l'estimation du planificateur du coût de transmission de chaque tuple (ligne) de l'application de travail vers le système dorsal principal.	–
pgaudit.log	Spécifie quelles classes d'instructions seront journalisées par la journalisation de l'audit de session.	–
pgaudit.log_catalog	Spécifie que la journalisation de session doit être activée dans le cas où toutes les relations d'une instruction se trouvent dans pg_catalog.	–
pgaudit.log_level	Spécifie le niveau de journal qui sera utilisé pour les entrées de journal.	–
pgaudit.log_parameter	Spécifie que la journalisation de l'audit doit inclure les paramètres transmis avec l'instruc tion.	–
pgaudit.log_relation	Spécifie si la journalisation de l'audit de session doit créer une entrée de journal distincte pour chaque relation (TABLE, VIEW, etc.) référencé e dans une instruction SELECT ou DML.	–
pgaudit.log_statem ent_once	Spécifie si la journalisation inclura le texte de l'instruction et les paramètres avec la première entrée de journal pour une combinais on instruction/sous-instruction ou avec chaque entrée.	–
pgaudit.role	Spécifie le rôle principal à utiliser pour la journalisation de l'audit des objets.	–
pg_bigm.enable_rec heck	Spécifie s'il faut effectuer une revérification qui est un processus interne de recherche en texte intégral.	on

Nom du paramètre	Description	Par défaut
pg_bigm.gin_key_limit	Spécifie le nombre maximum de 2 grammes du mot-clé de recherche à utiliser pour la recherche en texte intégral.	0
pg_bigm.last_update	Indique la dernière date de mise à jour du module pg_bigm.	22.11.2013
pg_bigm.similarity_limit	Spécifie le seuil minimal utilisé par la recherche de similarité.	0.3
pg_hint_plan.debug_print	Journalise les résultats de l'analyse des indices.	–
pg_hint_plan.enable_hint	Force le planificateur à utiliser les plans spécifiés dans le commentaire d'indice précédant la requête.	–
pg_hint_plan.enable_hint_table	Force le planificateur à ne pas obtenir d'indice à l'aide des recherches de table.	–
pg_hint_plan.message_level	Niveau de message des messages de débogage.	–
pg_hint_plan.parse_messages	Niveau de message des erreurs d'analyse.	–
pglogical.batch_inserts	Insertions de lots si possible	–
pglogical.conflict_log_level	Définit le niveau de journal utilisé pour la journalisation des conflits résolus.	–
pglogical.conflict_resolution	Définit la méthode utilisée pour la résolution des conflits résolubles.	–
pglogical.extra_connection_options	options de connexion à ajouter à toutes les connexions de nœuds de pairs	–

Nom du paramètre	Description	Par défaut
pglogical.synchronous_commit	valeur de validation synchrone spécifique pglogical	–
pglogical.use_spi	Utiliser une SPI au lieu d'une API de bas niveau pour appliquer les modifications	–
pg_similarity.block_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.block_threshold	Définit le seuil utilisé par la fonction de similarité Block.	–
pg_similarity.block_tokenizer	Définit le créateur de jetons pour la fonction de similarité Block.	–
pg_similarity.cosine_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.cosine_threshold	Définit le seuil utilisé par la fonction de similarité Cosine.	–
pg_similarity.cosine_tokenizer	Définit le créateur de jetons pour la fonction de similarité Cosine.	–
pg_similarity.dice_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.dice_threshold	Définit le seuil utilisé par la mesure de similarité Dice.	–
pg_similarity.dice_tokenizer	Définit le créateur de jetons pour la mesure de similarité Dice.	–
pg_similarity.euclidean_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.euclidean_threshold	Définit le seuil utilisé par la mesure de similarité Euclidean.	–

Nom du paramètre	Description	Par défaut
pg_similarity.euclidean_tokenizer	Définit le créateur de jetons pour la mesure de similarité Euclidean.	–
pg_similarity.hamming_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.hamming_threshold	Définit le seuil utilisé par la mesure de similarité Block.	–
pg_similarity.jaccard_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.jaccard_threshold	Définit le seuil utilisé par la mesure de similarité Jaccard.	–
pg_similarity.jaccard_tokenizer	Définit le créateur de jetons pour la mesure de similarité Jaccard.	–
pg_similarity.jaro_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.jaro_threshold	Définit le seuil utilisé par la mesure de similarité Jaro.	–
pg_similarity.jarowinkler_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.jarowinkler_threshold	Définit le seuil utilisé par la mesure de similarité Jarowinkler.	–
pg_similarity.levenshtein_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.levenshtein_threshold	Définit le seuil utilisé par la mesure de similarité Levenshtein.	–
pg_similarity.matching_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–

Nom du paramètre	Description	Par défaut
pg_similarity.matching_threshold	Définit le seuil utilisé par la mesure de similarité Matching Coefficient.	–
pg_similarity.matching_tokenizer	Définit le créateur de jetons pour la mesure de similarité Matching Coefficient.	–
pg_similarity.mongeelkan_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.mongeelkan_threshold	Définit le seuil utilisé par la mesure de similarité Monge-Elkan.	–
pg_similarity.mongeelkan_tokenizer	Définit le créateur de jetons pour la mesure de similarité Monge-Elkan.	–
pg_similarity.nw_gap_penalty	Définit la pénalité d'écart utilisée par la mesure de similarité Needleman-Wunsch.	–
pg_similarity.nw_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.nw_threshold	Définit le seuil utilisé par la mesure de similarité Needleman-Wunsch.	–
pg_similarity.overlap_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.overlap_threshold	Définit le seuil utilisé par la mesure de similarité Overlap Coefficient.	–
pg_similarity.overlap_tokenizer	Définit le créateur de jetons pour la mesure de similarité Overlap Coefficient.	–
pg_similarity.qgram_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.qgram_threshold	Définit le seuil utilisé par la mesure de similarité Q-Gram.	–

Nom du paramètre	Description	Par défaut
pg_similarity.qgram_tokenizer	Définit le créateur de jetons pour la mesure Q-Gram.	–
pg_similarity.swg_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.swg_threshold	Définit le seuil utilisé par la mesure de similarité Smith-Waterman-Gotoh.	–
pg_similarity.sw_is_normalized	Définit si la valeur du résultat est normalisée ou non.	–
pg_similarity.sw_threshold	Définit le seuil utilisé par la mesure de similarité Smith-Waterman.	–
pg_stat_statements.max	Définit le nombre maximal d'instructions suivies par pg_stat_statements.	–
pg_stat_statements.save	Enregistre les statistiques pg_stat_statements sur les arrêts de serveur.	–
pg_stat_statements.track	Définit les instructions qui sont suivies par pg_stat_statements.	–
pg_stat_statements.track_planning	Définit si le délai de planification est suivi par pg_stat_statements.	–
pg_stat_statements.track_utility	Définit si les commandes d'utilitaire sont suivies par pg_stat_statements.	–
postgis.gdal_enabled_drivers	Activez ou désactivez les pilotes GDAL utilisés avec PostGIS dans Postgres 9.3.5 et versions ultérieures.	ENABLE_ALL
quote_all_identifiers	Lors de la génération de fragments SQL, ajoute des guillemets à tous les identifiants.	–

Nom du paramètre	Description	Par défaut
random_page_cost	Définit l'estimation faite par le planificateur du coût d'une page de disque extraite de façon non séquentielle.	–
rds.enable_memory_management	Améliore les fonctionnalités de gestion de mémoire dans Aurora PostgreSQL 12.17, 13.13, 14.10, 15.5 et les versions ultérieures, empêchant ainsi les problèmes de stabilité et les redémarrages de base de données provoqués par une mémoire disponible insuffisante. Pour plus d'informations, consultez Gestion de mémoire améliorée dans Aurora PostgreSQL.	True
rds.force_admin_logging_level	Consultez les messages de journalisation des actions d'utilisateur de l'administrateur RDS dans les bases de données clients.	–
rds.log_retention_period	Amazon RDS supprimera les journaux PostgreSQL antérieurs à N minutes.	4320
rds.memory_allocation_guard	Améliore les fonctionnalités de gestion de mémoire dans Aurora PostgreSQL 11.21, 12.16, 13.12, 14.9, 15.4 et les versions ultérieures, empêchant ainsi les problèmes de stabilité et les redémarrages de base de données provoqués par une mémoire disponible insuffisante. Pour plus d'informations, consultez Gestion de mémoire améliorée dans Aurora PostgreSQL.	False
rds.pg_stat_ramdisk_size	Taille du disque RAM des statistiques en Mo. Une valeur différente de zéro va configurer le disque RAM.	0

Nom du paramètre	Description	Par défaut
rds.rds_superuser_reserved_connections	Définit le nombre d'emplacements de connexion réservés pour rds_superusers. Ce paramètre est uniquement disponible dans les versions 15 et antérieures. Pour plus d'informations, consultez la documentation de PostgreSQL sur reserved_connections.	2
rds.superuser_variables	Liste des variables réservées aux super-utilisateurs pour lesquelles nous augmentons les instructions de modification rds_superuser.	session_replication_role
remove_temp_files_after_crash	Supprime les fichiers temporaires après l'arrêt du système dorsal.	0
restart_after_crash	Réinitialisez le serveur après l'arrêt du système dorsal.	–
row_security	Activez la sécurité au niveau des lignes.	–
search_path	Définit l'ordre de recherche des schémas pour les noms pour lesquels le schéma n'est pas précisé.	–
seq_page_cost	Définit l'estimation faite par le planificateur du coût d'une page de disque extraite de façon séquentielle.	–
session_replication_role	Définit le comportement des sessions concernant les déclencheurs et les règles de réécriture.	–
shared_buffers	(8kB) Définit le nombre de tampons de mémoire partagée utilisés par le serveur.	SUM(DBInstanceClassMemory/12038,-50003)
shared_preload_libraries	Répertorie les bibliothèques partagées à précharger sur le serveur.	pg_stat_statements

Nom du paramètre	Description	Par défaut
ssl_ca_file	Emplacement du fichier d'autorité du serveur SSL.	/rdsdbdata/rds-metadata/ca-cert.pem
ssl_cert_file	Emplacement du fichier de certificat du serveur SSL.	/rdsdbdata/rds-metadata/server-cert.pem
ssl_crl_dir	Emplacement du répertoire de la liste de révocation des certificats SSL.	/rdsdbdata/rds-metadata/ssl_crl_dir/
ssl_key_file	Emplacement du fichier de clé privée du serveur SSL	/rdsdbdata/rds-metadata/server-key.pem
standard_conforming_strings	Entraîne les chaînes ... à traiter littéralement les barres obliques inverses.	–
statement_timeout	(ms) Définit la durée maximum de toute instruction.	–
stats_temp_directory	Écrit des fichiers de statistiques temporaires dans le répertoire spécifié.	/rdsdbdata/db/pg_stat_tmp
superuser_reserved_connections	Définit le nombre d'emplacements de connexion réservés aux super-utilisateurs.	3
synchronize_seqscans	Active les analyses séquentielles synchronisées.	–
tcp_keepalives_count	Nombre maximum de paquets TCP keepalive.	–
tcp_keepalives_idle	(s) Délai entre les émissions de paquets TCP keepalive.	–
tcp_keepalives_interval	(s) Délai entre les envois de paquets TCP keepalive.	–
temp_buffers	(8kB) Définit le nombre maximum de tampons temporaires utilisés par chaque session.	–

Nom du paramètre	Description	Par défaut
temp_file_limit	Contraint l'espace disque total en kilo-octe ts qu'un processus PostgreSQL donné peut utiliser pour les fichiers temporaires, à l'exclusi on de l'espace utilisé pour les tables temporair es explicites	-1
temp_tablespaces	Définit les espaces de table à utiliser pour les tables et fichiers de tri temporaires.	–
track_activities	Active la collecte d'informations sur l'exécution des commandes.	–
track_activity_que ry_size	Définit la taille réservée pour pg_stat_activity.c urrent_query, en octets.	4096
track_counts	Active la collecte de statistiques sur l'activité de la base de données.	–
track_functions	Active la collecte de statistiques au niveau de la fonction sur l'activité de la base de données.	pl
track_io_timing	Active la collecte de statistiques de durée sur l'activité I/O de la base de données.	1
transform_–_equals	Traite expr=– en tant que IS –.	–
update_process_title	Met à jour le titre du processus pour indiquer la commande SQL active.	–
wal_receiver_statu s_interval	(s) Définit l'intervalle maximal entre les rapports d'état du récepteur WAL et le principal.	–
work_mem	(kB) Définit la quantité maximum de mémoire que peuvent utiliser les espaces de travail des requêtes.	–

Nom du paramètre	Description	Par défaut
xmlbinary	Définit la façon dont les valeurs binaires doivent être codées en XML.	–
xmloption	Définit si des données XML dans des opération s d'analyse ou de sérialisation implicite s doivent être considérées comme des documents ou des fragments de contenu.	–

Événements d'attente Amazon Aurora PostgreSQL

Voici les événements d'attente courants pour Aurora PostgreSQL. Pour en savoir plus sur les événements d'attente et le réglage de votre cluster de bases de données Aurora PostgreSQL, consultez Réglage des événements d'attente pour Aurora PostgreSQL.

Activité : ArchiverMain

Le processus d'archivage est en attente d'activité.

Activité : AutoVacuumMain

Le processus de lancement de l'autovacuum est en attente d'activité.

Activité : BgWriterHibernate

Le processus d'écriture en arrière-plan est en veille prolongée lors de l'attente d'activité.

Activité : BgWriterMain

Le processus d'écriture en arrière-plan est en attente d'activité.

Activité : CheckpointerMain

Le processus du pointeur de contrôle est en attente d'activité.

Activité : LogicalApplyMain

Le processus d'application de réplication logique est en attente d'activité.

Activité : LogicalLauncherMain

Le processus de lancement de réplication logique est en attente d'activité.

Activité : PgStatMain

Le processus de collecte de statistiques est en attente d'activité.

Activité : RecoveryWalAll

Un processus est en attente du journal d'écriture anticipée (WAL) à partir d'un flux lors de la récupération.

Activité : RecoveryWalStream

Le processus de démarrage est en attente de l'arrivée du journal d'écriture anticipée (WAL) pendant la récupération du streaming.

Activité : SysLoggerMain

Le processus syslogger est en attente d'activité.

Activité : WalReceiverMain

Le processus de réception du journal d'écriture anticipée (WAL) est en attente d'activité.

Activité : WalSenderMain

Le processus d'expédition WAL est en attente d'activité.

Activité : WalWriterMain

Le processus d'écriture WAL est en attente d'activité.

BufferPin:BufferPin

Un processus attend d'acquérir une connexion exclusive sur un tampon.

Client : GSSOpen serveur

Un processus attend de lire les données du client lors de l'établissement d'une session GSSAPI (Generic Security Service Application Program Interface).

Cliente : ClientRead

Un processus dorsal attend de recevoir des données d'un client PostgreSQL. Pour de plus amples informations, veuillez consulter Client:ClientRead.

Cliente : ClientWrite

Un processus dorsal attend d'envoyer davantage de données à un client PostgreSQL. Pour de plus amples informations, veuillez consulter Client:ClientWrite.

Client : Lib PQWal ReceiverConnect

Un processus est en attente dans le récepteur du journal d'écriture anticipée (WAL) d'établir la connexion au serveur distant.

Client : Lib PQWal ReceiverReceive

Un processus est en attente dans le récepteur WAL de réception des données du serveur distant.

Client : SSLOpen serveur

Un processus est en attente du protocole SSL pendant la tentative de connexion.

Cliente : WalReceiverWaitStart

Un processus attend que le processus de démarrage envoie des données initiales pour la réplication du streaming.

Client : WalSenderWaitFor WAL

Un processus attend le vidage du journal d'écriture anticipée (WAL) dans le processus d'expédition WAL.

Cliente : WalSenderWriteData

Un processus est en attente d'activité lors du traitement des réponses provenant du récepteur WAL du processus d'expédition WAL.

CPU

Un processus de backend est actif ou en attente du processeur. Pour plus d'informations, consultez CPU.

Extension:extension

Un processus dorsal est en attente d'une condition définie par une extension ou un module.

IO : AuroraEnhancedLogical WALRead

Un processus dorsal extrait les enregistrements du journal depuis le volume de capture des données modifiées (CDC).

IO : AuroraOptimizedReadsCacheRead

Un processus est en attente d'une lecture du cache à plusieurs niveaux Optimized Reads, car la page n'est pas disponible en mémoire partagée.

IO : AuroraOptimizedReads CacheSegmenttronquer

Un processus est en attente de la troncature d'un fichier de segments du cache à plusieurs niveaux Optimized Reads.

IO : AuroraOptimizedReadsCacheWrite

Le processus d'écriture en arrière-plan est en attente d'une écriture dans le cache à plusieurs niveaux Optimized Reads.

IO : AuroraStorageLogAllocate

Une session alloue des métadonnées et prépare l'écriture d'un journal de transactions.

IO : BufFileRead

Lorsque les opérations nécessitent plus de mémoire que la quantité définie par les paramètres de mémoire de travail, le moteur crée des fichiers temporaires sur le disque. Cet événement d'attente se produit lorsque les opérations sont lues à partir des fichiers temporaires. Pour de plus amples informations, veuillez consulter IO : BufFileRead et IO : BufFileWrite.

IO : BufFileWrite

Lorsque les opérations nécessitent plus de mémoire que la quantité définie par les paramètres de mémoire de travail, le moteur crée des fichiers temporaires sur le disque. Cet événement d'attente se produit lorsque les opérations sont écrites dans des fichiers temporaires. Pour de plus amples informations, veuillez consulter IO : BufFileRead et IO : BufFileWrite.

IO : ControlFileRead

Un processus est en attente d'une lecture du fichier `pg_control`.

IO : ControlFileSync

Un processus attend que le fichier `pg_control` atteigne un stockage durable.

IO : ControlFileSyncUpdate

Un processus est en attente qu'une mise à jour du fichier `pg_control` atteigne un stockage durable.

IO : ControlFileWrite

Un processus est en attente d'une écriture dans le fichier `pg_control`.

IO : ControlFileWriteUpdate

Un processus est en attente d'une écriture de mise à jour du fichier `pg_control`.

IO : CopyFileRead

Un processus est en attente de lecture pendant une opération de copie de fichier.

IO : CopyFileWrite

Un processus est en attente d'une écriture pendant une opération de copie de fichier.

IO : DataFileExtend

Un processus attend qu'un fichier de données de relation soit étendu.

IO : DataFileFlush

Un processus est en attente qu'un fichier de données de relation atteigne un stockage durable.

IO : DataFileImmediateSync

Un processus est en attente de synchronisation immédiate d'un fichier de données de relation sur un stockage durable.

IO : DataFilePrefetch

Un processus est en attente d'une récupération préalable asynchrone depuis un fichier de données de relation.

IO : DataFileSync

Un processus est en attente de modification d'un fichier de données de relation sur un stockage durable.

IO : DataFileRead

Un processus dorsal a essayé de trouver une page dans les tampons partagés, ne l'a pas trouvée et l'a donc lue depuis le stockage. Pour de plus amples informations, veuillez consulter IO:DataFileRead.

IO : DataFileTruncate

Un processus attend qu'un fichier de données de relation soit tronqué.

IO : DataFileWrite

Un processus est en attente d'une écriture dans un fichier de données de relation.

IO : DSMFill ZeroWrite

Un processus attend d'écrire zéro octet dans un fichier de sauvegarde dynamique de mémoire partagée.

IO : LockFileAddToDataDirRead

Un processus est en attente d’une lecture lors de l’ajout d’une ligne au fichier de verrouillage du répertoire de données.

IO : LockFileAddToDataDirSync

Un processus est en attente que les données atteignent un stockage durable lors de l’ajout d’une ligne au fichier de verrouillage du répertoire de données.

IO : LockFileAddToDataDirWrite

Un processus est en attente d’une écriture lors de l’ajout d’une ligne au fichier de verrouillage du répertoire de données.

IO : LockFileCreateRead

Un processus est en attente d’une lecture lors de la création du fichier de verrouillage du répertoire de données.

IO : LockFileCreateSync

Un processus est en attente que les données atteignent un stockage durable lors de la création du fichier de verrouillage du répertoire de données.

IO : LockFileCreateWrite

Un processus est en attente d’une écriture lors de la création du fichier de verrouillage du répertoire de données.

IO : LockFileReCheckDataDirRead

Un processus est en attente d’une lecture lors de la revérification du fichier de verrouillage du répertoire de données.

IO : LogicalRewriteCheckpointSync

Un processus attend que des mappages de réécriture logiques atteignent un stockage durable pendant un point de contrôle.

IO : LogicalRewriteMappingSync

Un processus attend que les données de mappage atteignent un stockage durable pendant une réécriture logique.

IO : LogicalRewriteMappingWrite

Un processus est en attente d’une écriture de données de mappage lors d’une réécriture logique.

IO : LogicalRewriteSync

Un processus attend que les mappages de réécriture logique atteignent un stockage durable.

IO : LogicalRewriteTruncate

Un processus est en attente de la troncature des données de mappage lors d'une réécriture logique.

IO : LogicalRewriteWrite

Un processus est en attente d'une écriture des mappages de réécriture logique.

IO : RelationMapRead

Un processus est en attente d'une lecture d'un fichier de mappage de relation.

IO : RelationMapSync

Un processus est en attente que le fichier de mappage de relation atteigne un stockage durable.

IO : RelationMapWrite

Un processus est en attente d'une écriture dans le fichier de mappage de relation.

IO : ReorderBufferRead

Un processus est en attente d'une lecture pendant la gestion du tampon de réorganisation.

IO : ReorderBufferWrite

Un processus est en attente d'une écriture pendant la gestion de la mémoire tampon de réorganisation.

IO : ReorderLogicalMappingRead

Un processus est en attente d'une lecture d'un mappage logique pendant la gestion de la mémoire tampon de réorganisation.

IO : ReplicationSlotRead

Un processus est en attente d'une lecture à partir d'un fichier de contrôle d'emplacement de réplication.

IO : ReplicationSlotRestoreSync

Un processus attend qu'un fichier de contrôle d'emplacement de réplication atteigne un stockage durable tout en le rétablissant en mémoire.

IO : ReplicationSlotSync

Un processus attend qu'un fichier de contrôle d'emplacement de réplication atteigne un stockage durable.

IO : ReplicationSlotWrite

Un processus est en attente d'une écriture sur un fichier de contrôle d'emplacement de réplication.

IO : SLRUFlush Synchronisation

Un processus attend que les données simples les moins récemment utilisées (SLRU) atteignent un stockage durable lors d'un point de contrôle ou de l'arrêt de la base de données.

IO : SLRURead

Un processus attend la lecture d'une page simple la moins récemment utilisée (SLRU).

IO : SLRUSync

Un processus attend que les données simples les moins récemment utilisées (SLRU) atteignent un stockage durable après l'écriture d'une page.

IO : SLRUWrite

Un processus attend l'écriture d'une page simple la moins récemment utilisée (SLRU).

IO : SnapbuildRead

Un processus attend la lecture d'un instantané de catalogue historique sérialisé.

IO : SnapbuildSync

Un processus attend qu'un instantané de catalogue historique sérialisé atteigne un stockage durable.

IO : SnapbuildWrite

Un processus attend l'écriture d'un instantané de catalogue historique sérialisé.

IO : TimelineHistoryFileSync

Un processus attend qu'un fichier d'historique de chronologie reçu via la réplication de streaming atteigne un stockage durable.

IO : TimelineHistoryFileWrite

Un processus attend qu'une écriture d'un fichier d'historique de chronologie soit reçu via la réplication de streaming.

IO : TimelineHistoryRead

Un processus est en attente de lecture d'un fichier d'historique de chronologie.

IO : TimelineHistorySync

Un processus attend qu'un fichier d'historique de chronologie nouvellement créé atteigne un stockage durable.

IO : TimelineHistoryWrite

Un processus est en attente d'écriture d'un fichier d'historique de chronologie nouvellement créé.

IO : TwophaseFileRead

Un processus attend la lecture d'un fichier d'état à deux phases.

IO : TwophaseFileSync

Un processus est en attente qu'un fichier d'état à deux phases atteigne un stockage durable.

IO : TwophaseFileWrite

Un processus attend une écriture d'un fichier d'état à deux phases.

IO : WALBootstrap Synchronisation

Un processus attend que le journal d'écriture anticipée (WAL) atteigne un stockage durable pendant l'action d'amorçage.

IO : WALBootstrap Écrivez

Un processus est en attente d'écriture d'une page d'un journal d'écriture anticipée (WAL) pendant l'action d'amorçage.

IO : WALCopy Lire

Un processus attend la lecture lors de la création d'un nouveau segment de journal d'écriture anticipée (WAL) en copiant un segment existant.

IO : WALCopy Synchronisation

Un processus attend qu'un nouveau segment WAL créé en copiant un segment existant atteigne un stockage durable.

IO : WALCopy Écrivez

Un processus attend une écriture lors de la création d'un nouveau segment WAL en copiant un segment existant.

IO : WALInit Synchronisation

Un processus attend qu'un fichier de journal d'écriture anticipée (WAL) nouvellement initialisé atteigne un stockage durable.

IO : WALInit Écrivez

Un processus est en attente d'une écriture lors de l'initialisation d'un nouveau fichier de journal d'écriture anticipée (WAL).

IO : WALRead

Un processus est en attente d'une lecture à partir d'un fichier de journal d'écriture anticipée (WAL).

IO : WALSender TimelineHistoryRead

Un processus est en attente d'une lecture à partir d'un fichier d'historique de chronologie pendant une commande de chronologie d'expédition WAL.

IO : WALSync

Un processus attend qu'un fichier WAL atteigne un stockage durable.

IO : WALSync MethodAssign

Un processus attend que les données atteignent un stockage durable lors de l'affectation d'une nouvelle méthode de synchronisation WAL.

IO : WALWrite

Un processus est en attente d'une écriture dans un fichier WAL.

IO : XactSync

Un processus dorsal attend que le sous-système de stockage Aurora confirme la validation d'une transaction standard, ou la validation ou restauration d'une transaction préparée. Pour de plus amples informations, veuillez consulter IO:XactSync.

IPC : AuroraLogicalSchemaUpdate

Deux processus dorsaux tentent d'insérer la même entrée dans le cache du schéma. Un seul processus se poursuit pendant que l'autre attend qu'il se termine.

IPC : AuroraOptimizedReadsCacheWriteStop

Un processus attend que l'enregistreur en arrière-plan arrête d'écrire dans le cache à plusieurs niveaux Optimized Reads.

IPC : BackupWaitWalArchive

Un processus attend les fichiers du journal d'écriture anticipée (WAL) nécessaires à l'archivage réussi d'une sauvegarde.

IPC : BgWorkerShutdown

Un processus attend la fermeture d'une application de travail en arrière-plan.

IPC : BgWorkerStartup

Un processus attend le démarrage d'une application de travail en arrière-plan.

IPC : BtreePage

Un processus est en attente de la disponibilité du numéro de page nécessaire pour poursuivre une analyse parallèle de l'arborescence B.

IPC : CheckpointDone

Un processus attend la fin d'un point de contrôle.

IPC : CheckpointStart

Un processus attend le début d'un point de contrôle.

IPC : ClogGroupUpdate

Un processus attend que le chef de groupe mette à jour le statut de la transaction à la fin d'une transaction.

IPC : DamRecordTxAck

Un processus dorsal a généré un événement de flux d'activité de base de données et attend que l'événement devienne durable. Pour de plus amples informations, veuillez consulter IPC:DamRecordTxAck.

IPC : ExecuteGather

Un processus attend l'activité d'un processus enfant lors de l'exécution d'un nœud de plan Gather.

IPC : Hash/Batch/Allocating

Un processus attend qu'un participant au hachage parallèle élu alloue une table de hachage.

IPC : Hash/Batch/Electing

Un processus élit un participant au hachage parallèle pour allouer une table de hachage.

IPC : Hash/Batch/Loading

Un processus attend que d'autres participants au hachage parallèle terminent le chargement d'une table de hachage.

IPC : Hash/Build/Allocating

Un processus attend qu'un participant au hachage parallèle élu alloue la table de hachage initiale.

IPC : Hash/Build/Electing

Un processus élit un participant au hachage parallèle pour allouer la table de hachage initiale.

IPC : Hash/Build/HashingInner

Un processus attend que d'autres participants au hachage parallèle terminent le hachage de la relation interne.

IPC : Hash/Build/HashingOuter

Un processus attend que d'autres participants au hachage parallèle terminent le partitionnement de la relation externe.

IPC : Hash/GrowBatches/Allocating

Un processus attend qu'un participant au hachage parallèle élu alloue d'autres lots.

IPC : Hash/GrowBatches/Deciding

Un processus élit un participant au hachage parallèle pour décider de la croissance future des lots.

IPC : Hash/GrowBatches/Electing

Un processus élit un participant au hachage parallèle pour allouer plus de lots.

IPC : Hash/GrowBatches/Finishing

Un processus attend qu'un participant au hachage parallèle élu décide de la croissance future des lots.

IPC : Hash/GrowBatches/Repartitioning

Un processus attend que d'autres participants au hachage parallèle terminent le partitionnement.

IPC : Hash/GrowBuckets/Allocating

Un processus attend qu'un participant au hachage parallèle élu finisse d'allouer plus de compartiments.

IPC : Hash/GrowBuckets/Electing

Un processus élit un participant au hachage parallèle pour allouer plus de compartiments.

IPC : Hash/GrowBuckets/Reinserting

Un processus attend que d'autres participants au hachage parallèle finissent d'insérer des tuples dans de nouveaux compartiments.

IPC : HashBatchAllocate

Un processus attend qu'un participant au hachage parallèle élu alloue une table de hachage.

IPC : HashBatchElect

Un processus attend d'élire un participant au hachage parallèle pour allouer une table de hachage.

IPC : HashBatchLoad

Un processus attend que d'autres participants au hachage parallèle terminent le chargement d'une table de hachage.

IPC : HashBuildAllocate

Un processus attend qu'un participant au hachage parallèle élu alloue la table de hachage initiale.

IPC : HashBuildElect

Un processus attend d'élire un participant au hachage parallèle pour allouer la table de hachage initiale.

IPC : HashBuildHashInner

Un processus attend que d'autres participants au hachage parallèle terminent le hachage de la relation interne.

CIP : 'HashBuildHashOuter

Un processus attend que d'autres participants au hachage parallèle terminent le partitionnement de la relation externe.

IPC : HashGrowBatchesAllocate

Un processus attend qu'un participant au hachage parallèle élu alloue d'autres lots.

CIP : 'HashGrowBatchesDecide

Un processus attend d'élire un participant au hachage parallèle pour décider de la croissance future des lots.

IPC : HashGrowBatchesElect

Un processus attend d'élire un participant au hachage parallèle pour allouer d'autres lots.

IPC : HashGrowBatchesFinish

Un processus attend qu'un participant au hachage parallèle élu décide de la croissance future des lots.

IPC : HashGrowBatchesRepartition

Un processus attend que d'autres participants au hachage parallèle terminent le partitionnement.

IPC : HashGrowBucketsAllocate

Un processus attend qu'un participant au hachage parallèle élu finisse d'allouer plus de compartiments.

IPC : HashGrowBucketsElect

Un processus attend d'élire un participant au hachage parallèle pour allouer plus de compartiments.

IPC : HashGrowBucketsReinsert

Un processus attend que d'autres participants au hachage parallèle finissent d'insérer des tuples dans de nouveaux compartiments.

IPC : LogicalSyncData

Un processus attend qu'un serveur distant de réplication logique envoie des données pour la synchronisation initiale de la table.

IPC : LogicalSyncStateChange

Un processus attend qu'un serveur distant de réplication logique change d'état.

IPC : MessageQueueInternal

Un processus attend qu'un autre processus soit attaché à une file d'attente de messages partagée.

IPC : MessageQueuePutMessage

Un processus attend d'écrire un message de protocole dans une file d'attente de messages partagée.

IPC : MessageQueueReceive

Un processus attend de recevoir des octets d'une file d'attente de messages partagée.

IPC : MessageQueueSend

Un processus attend d'envoyer des octets à une file d'attente de messages partagée.

IPC : ParallelBitmapScan

Un processus attend l'initialisation d'une analyse bitmap parallèle.

IPC : ParallelCreateIndexScan

Un processus attend que les applications de travail CREATE INDEX parallèles terminent une analyse de pile.

IPC : ParallelFinish

Un processus attend que les applications de travail parallèles terminent le calcul.

IPC : ProcArrayGroupUpdate

Un processus attend que le chef de groupe efface le statut de la transaction à la fin de la transaction.

IPC : ProcSignalBarrier

Un processus attend qu'un événement de barrière soit traité par tous les systèmes dorsaux.

IPC:Promote

Un processus attend une promotion en attente.

IPC : RecoveryConflictSnapshot

Un processus attend la résolution des conflits de récupération pour un nettoyage par le vide.

IPC : RecoveryConflictTablespace

Un processus attend la résolution des conflits de récupération pour supprimer un espace de table.

IPC : RecoveryPause

Un processus attend la reprise de la récupération.

IPC : ReplicationOriginDrop

Un processus attend qu'une origine de réplication devienne inactive pour pouvoir la supprimer.

IPC : ReplicationSlotDrop

Un processus attend qu'un emplacement de réplication devienne inactif pour pouvoir le supprimer.

IPC : SafeSnapshot

Un processus attend d'obtenir un instantané valide pour une transaction READ ONLY DEFERRABLE.

IPC : SyncRep

Un processus est en attente de confirmation de la part d'un serveur distant pendant la réplication synchrone.

IPC : XactGroupUpdate

Un processus attend que le chef de groupe mette à jour le statut de la transaction à la fin de la transaction.

Lock:advisory

Un processus dorsal a demandé un verrou consultatif et l'attend. Pour plus d'informations, consultez Lock:advisory.

Lock:extend

Un processus dorsal attend qu'un verrou soit libéré afin de pouvoir étendre une relation. Ce verrou est nécessaire car un seul processus de backend peut étendre une relation à la fois. Pour plus d'informations, consultez Lock:extend.

Lock:frozenid

Un processus est en attente de mise à jour de `pg_database.datfrozenxid` et `pg_database.datminmxid`.

Lock:object

Un processus attend d'obtenir un verrou sur un objet de base de données sans relation.

Lock:page

Un processus attend d'obtenir un verrou sur une page d'une relation.

Lock:Relation

Un processus dorsal attend d'acquérir un verrou sur une relation verrouillée par une autre transaction. Pour plus d'informations, consultez Lock:Relation.

Lock:spectoken

Un processus attend d'obtenir un verrou d'insertion spéculatif.

Lock:speculative token

Un processus attend d'acquérir un verrou d'insertion spéculatif.

Lock:transactionid

Une transaction est en attente d'un verrou au niveau de la ligne. Pour plus d'informations, consultez Lock:transactionid.

Lock:tuple

Un processus dorsal attend d'acquérir un verrou sur un tuple alors qu'un autre processus dorsal contient un verrou conflictuel sur le même tuple. Pour plus d'informations, consultez Lock:tuple.

Lock:userlock

Un processus attend d'obtenir un verrou utilisateur.

Lock:virtualxid

Un processus attend d'obtenir un verrou d'ID de transaction virtuel.

LWLock:AddinShmemInit

Un processus est en attente de gestion de l'allocation d'espace d'une extension dans la mémoire partagée.

LWLock:AddinShmemInitLock

Un processus est en attente de gestion de l'allocation d'espace dans la mémoire partagée.

LWLock:asynchrone

Un processus est en attente I/O sur une mémoire tampon asynchrone (notification).

LWLock:AsyncCtlLock

Un processus est en attente de lecture ou de mise à jour d'un état de notification partagé.

LWLock:AsyncQueueLock

Un processus est en attente de lecture ou de mise à jour des messages de notification.

LWLock:AuroraOptimizedReadsCacheMapping

Un processus attend d'associer un bloc de données à une page dans le cache à plusieurs niveaux Optimized Reads.

LWLock:AutoFile

Un processus est en attente de mise à jour du fichier `postgresql.auto.conf`.

LWLock:AutoFileLock

Un processus est en attente de mise à jour du fichier `postgresql.auto.conf`.

LWLock: Aspirateur automatique

Un processus est en attente de lecture ou de mise à jour de l'état actuel des applications de travail d'autovacuum.

LWLock:AutovacuumLock

Une application de travail ou un lanceur d'applications d'autovacuum attend de mettre à jour ou de lire l'état actuel des applications de travail d'autovacuum.

LWLock:AutovacuumSchedule

Un processus est en attente pour s'assurer qu'une table sélectionnée pour l'autovacuum doit encore être vidée.

LWLock:AutovacuumScheduleLock

Un processus est en attente pour s'assurer que la table choisie pour un vidage doit encore être vidée.

LWLock:BackendRandomLock

Un processus est en attente pour générer un nombre aléatoire.

LWLock:BackgroundWorker

Un processus est en attente de lecture ou de mise à jour de l'état de l'application de travail en arrière-plan.

LWLock:BackgroundWorkerLock

Un processus est en attente de lecture ou de mise à jour de l'état de l'application de travail en arrière-plan.

LWLock:BtreeVacuum

Un processus est en attente de lecture ou de mise à jour des informations relatives au vide pour un index d'arborescence B.

LWLock:BtreeVacuumLock

Un processus est en attente de lecture ou de mise à jour des informations relatives au vide pour un index d'arborescence B.

LWLock:contenu_du tampon

Un processus dorsal attend d'acquérir un verrou léger sur le contenu d'un tampon de mémoire partagée. Pour de plus amples informations, veuillez consulter LWLock:buffer_content (BufferContent).

LWLock:mappage_tampon

Un processus dorsal attend d'associer un bloc de données à une mémoire tampon dans le groupe de mémoires tampons partagées. Pour de plus amples informations, veuillez consulter LWLock:buffer_mapping.

LWLock: BufferIO

Un processus de backend souhaite lire une page en mémoire partagée. Le processus attend que les autres processus terminent leur tâche I/O pour la page. Pour de plus amples informations, veuillez consulter LWLock:BufferIO (IPC:BufferIO).

LWLock:Point de contrôle

Un processus attend de commencer un point de contrôle.

LWLock:CheckpointLock

Un processus attend d'exécuter un point de contrôle.

LWLock:CheckpointerComm

Un processus attend de gérer les demandes `fsync`.

LWLock:CheckpointerCommLock

Un processus attend de gérer les demandes `fsync`.

LWLock:sabot

Un processus est en attente I/O dans une mémoire tampon obstruée (statut de transaction).

LWLock:CLogControlLock

Un processus attend de lire ou de mettre à jour l'état de la transaction.

LWLock:CLogTruncationLock

Un processus attend d’exécuter `txid_status` ou de mettre à jour l’ID de transaction le plus ancien disponible.

LWLock:commit_timestamp

Un processus est en attente I/O sur une mémoire tampon d'horodatage de validation.

LWLock:CommitTs

Un processus attend de lire ou de mettre à jour la dernière valeur définie pour un horodatage de validation de transaction.

LWLock:CommitTsBuffer

Un processus attend un horodatage de validation I/O sur un tampon simple utilisé le moins récemment (SLRU).

LWLock:CommitTsControlLock

Un processus attend de lire ou de mettre à jour les horodatages de validation de transactions.

LWLock:CommitTsLock

Un processus attend de lire ou de mettre à jour la dernière valeur définie pour l’horodatage de la transaction.

LWLock: CommitTs SLRU

Un processus attend d’accéder au cache simple le moins récemment utilisé (SLRU) pour un horodatage de livraison.

LWLock:ControlFile

Un processus attend de lire ou de mettre à jour le fichier `pg_control` ou de créer un nouveau fichier WAL.

LWLock:ControlFileLock

Un processus attend de lire ou de mettre à jour le fichier de contrôle ou de créer un nouveau fichier WAL.

LWLock:DynamicSharedMemoryControl

Un processus attend de lire ou de mettre à jour les informations d’allocation de mémoire partagée dynamique.

LWLock:DynamicSharedMemoryControlLock

Un processus attend de lire ou de mettre à jour l'état de la mémoire partagée dynamique.

LWLock:lock_manager

Un processus dorsal attend d'ajouter ou d'examiner des verrous pour les processus dorsal. Ou il attend de rejoindre ou de quitter un groupe de verrouillage utilisé par une requête parallèle. Pour de plus amples informations, veuillez consulter LWLock:lock_manager.

LWLock:LockFastPath

Un processus attend de lire ou de mettre à jour les informations de verrouillage du chemin d'accès rapide d'un processus.

LWLock:LogicalRepWorker

Un processus attend de lire ou de mettre à jour l'état des applications de travail de réplication logique.

LWLock:LogicalRepWorkerLock

Un processus attend la fin d'une action sur une application de travail de réplication logique.

LWLock:LogicalSchemaCache

Un processus a modifié le cache du schéma.

LWLock:multixact_member

Un processus est en attente I/O sur un tampon multixact_member.

LWLock:multixact_offset

Un processus est en attente I/O sur un tampon de décalage multixact.

LWLock:MultiXactGen

Un processus est en attente de lecture ou de mise à jour de l'état multixact partagé.

LWLock:MultiXactGenLock

Un processus est en attente de lecture ou de mise à jour d'un état multixact partagé.

LWLock:MultiXactMemberBuffer

Un processus est en attente I/O sur une mémoire tampon simple la moins récemment utilisée (SLRU) pour un membre multixact. Pour de plus amples informations, veuillez consulter LWLock:MultiXact.

LWLock:MultiXactMemberControlLock

Un processus est en attente de lecture ou de mise à jour des mappages de membres multixact.

LWLock: MultiXactMember SLRU

Un processus attend d'accéder au cache simple le moins récemment utilisé (SLRU) pour un membre multixact. Pour de plus amples informations, veuillez consulter LWLock:MultiXact.

LWLock:MultiXactOffsetBuffer

Un processus attend un décalage I/O multixact sur un tampon simple le moins récemment utilisé (SLRU). Pour de plus amples informations, veuillez consulter LWLock:MultiXact.

LWLock:MultiXactOffsetControlLock

Un processus est en attente de lecture ou de mise à jour des mappages de décalages multixact.

LWLock: MultiXactOffset SLRU

Un processus attend d'accéder au cache simple le moins récemment utilisé (SLRU) pour un décalage multixact. Pour de plus amples informations, veuillez consulter LWLock:MultiXact.

LWLock:MultiXactTruncation

Un processus attend de lire ou de tronquer des informations multixact.

LWLock:MultiXactTruncationLock

Un processus attend de lire ou de tronquer des informations multixact.

LWLock:NotifyBuffer

Un processus attend un message NOTIFY I/O sur la mémoire tampon la moins récemment utilisée (SLRU).

LWLock:NotifyQueue

Un processus est en attente de lecture ou de mise à jour des messages NOTIFY.

LWLock:NotifyQueueTail

Un processus attend de mettre à jour une limite sur le stockage des messages NOTIFY.

LWLock:NotifyQueueTailLock

Un processus attend de mettre à jour la limite de stockage des messages de notification.

LWLock: Notifier SLRU

Un processus attend d’accéder au cache simple le moins récemment utilisé (SLRU) pour un message NOTIFY.

LWLock:OidGen

Un processus attend d’allouer un nouvel ID d’objet (OID).

LWLock:OidGenLock

Un processus attend d’allouer ou d’affecter un nouvel ID d’objet (OID).

LWLock:oldserxid

Un processus est en attente I/O sur un tampon oldserxid.

LWLock:OldSerXidLock

Un processus attend de lire ou d’enregistrer des transactions sérialisables en conflit.

LWLock:OldSnapshotTimeMap

Un processus attend de lire ou de mettre à jour les anciennes informations de contrôle des instantanés.

LWLock:OldSnapshotTimeMapLock

Un processus attend de lire ou de mettre à jour les anciennes informations de contrôle des instantanés.

LWLock:parallel_ajouter

Un processus attend de choisir le sous-plan suivant lors de l’exécution du plan d’ajout parallèle.

LWLock:parallel_hash_join

Un processus attend d’allouer ou d’échanger un morceau de mémoire ou de mettre à jour des compteurs pendant l’exécution d’un plan de hachage parallèle.

LWLock:parallel_query_dsa

Un processus attend le verrouillage de l’allocation dynamique de mémoire partagée pour une requête parallèle.

LWLock:ParallelAppend

Un processus attend de choisir le sous-plan suivant lors de l’exécution du plan d’ajout parallèle.

LWLock:ParallelHashJoin

Un processus est en attente de synchronisation des applications de travail pendant l'exécution du plan pour une jointure de hachage parallèle.

Lloque : DSA ParallelQuery

Un processus attend une allocation dynamique de mémoire partagée pour une requête parallèle.

Lloque : DSA PerSession

Un processus attend une allocation dynamique de mémoire partagée pour une requête parallèle.

Lloque : PerSessionRecordType

Un processus attend d'accéder aux informations d'une requête parallèle sur les types composites.

Lloque : PerSessionRecordTypmod

Un processus attend d'accéder aux informations d'une requête parallèle sur les modificateurs de type qui identifient les types d'enregistrements anonymes.

Lloque : PerXactPredicateList

Un processus attend d'accéder à la liste des verrous de prédicats détenus par la transaction sérialisable en cours lors d'une requête parallèle.

Lwlock:predicate_lock_manager

Un processus attend d'ajouter ou d'examiner des informations de verrouillage du prédicat.

Lloque : PredicateLockManager

Un processus attend d'accéder aux informations de verrouillage des prédicats utilisées par les transactions sérialisables.

Lwlock:proc

Un processus attend de lire ou de mettre à jour les informations de verrouillage du chemin d'accès rapide.

LWLock:ProcArray

Un processus attend d'accéder aux structures de données partagées par processus (généralement pour obtenir un instantané ou signaler l'ID de transaction d'une session).

LWLock:ProcArrayLock

Un processus attend d'obtenir un instantané ou d'effacer un ID de transaction à la fin d'une transaction.

LWLock:RelationMapping

Un processus attend de lire ou de mettre à jour un fichier `pg_filenode.map` (utilisé pour suivre les affectations de nœuds de fichiers de certains catalogues système).

LWLock:RelationMappingLock

Un processus attend de mettre à jour le fichier de mappage des relations utilisé pour stocker le catalog-to-file-node mappage.

LWLock:RelCacheInit

Un processus attend de lire ou de mettre mise à jour un fichier `pg_internal.init` (fichier d'initialisation du cache de relation).

LWLock:RelCacheInitLock

Un processus attend de lire ou d'écrire un fichier d'initialisation du cache de relation.

LWLock:origine de la réplication

Un processus attend de lire ou de mettre à jour la progression de la réplication.

LWLock:replication_slot_io

Un processus est en attente I/O sur un emplacement de réplication.

LWLock:ReplicationOrigin

Un processus est en attente de création, de suppression ou d'utilisation d'une origine de réplication.

LWLock:ReplicationOriginLock

Un processus est en attente de configuration, de suppression ou d'utilisation d'une origine de réplication.

LWLock:ReplicationOriginState

Un processus attend de lire ou de mettre à jour la progression d'une origine de réplication.

LWLock:ReplicationSlotAllocation

Un processus attend d'allouer ou de libérer un emplacement de réplication.

LWLock:ReplicationSlotAllocationLock

Un processus attend d'allouer ou de libérer un emplacement de réplication.

LWLock:ReplicationSlotControl

Un processus attend de lire ou de mettre à jour un état d'emplacement de réplication.

LWLock:ReplicationSlotControlLock

Un processus attend de lire ou de mettre à jour l'état d'emplacement de réplication.

LWLock: ReplicationSlot IO

Un processus est en attente I/O sur un emplacement de réplication.

LWLock:SerialBuffer

Un processus attend un simple tampon utilisé I/O le moins récemment (SLRU) pour un conflit de transaction sérialisable.

LWLock:SerializableFinishedList

Un processus attend d'accéder à la liste des transactions sérialisables terminées.

LWLock:SerializableFinishedListLock

Un processus attend d'accéder à la liste des transactions sérialisables terminées.

LWLock:SerializablePredicateList

Un processus attend d'accéder à la liste des verrous de prédicats détenus par des transactions sérialisables.

LWLock:SerializablePredicateLockListLock

Un processus est en attente d'exécution d'une opération sur une liste de verrous détenus par des transactions sérialisables.

LWLock:SerializableXactHash

Un processus attend de lire ou de mettre à jour des informations sur les transactions sérialisables.

LWLock:SerializableXactHashLock

Un processus attend de récupérer ou de stocker des informations sur les transactions sérialisables.

LWLock: Série SLRU

Un processus attend d'accéder au cache simple le moins récemment utilisé (SLRU) pour un conflit de transaction sérialisable.

LWLock:SharedTidBitmap

Un processus attend d’accéder à un bitmap d’identifiant de tuple partagé (TID) lors d’une analyse d’index bitmap parallèle.

LWLock:SharedTupleStore

Un processus attend d’accéder à un magasin de tuples partagé lors d’une requête parallèle.

LWLock:ShmemIndex

Un processus attend de trouver ou d’allouer de l’espace dans la mémoire partagée.

LWLock:ShmemIndexLock

Un processus attend de trouver ou d’allouer de l’espace dans la mémoire partagée.

LWLock: SInval Lisez

Un processus attend de récupérer des messages de la file d’attente d’invalidation du catalogue partagé.

LWLock:SInvalReadLock

Un processus attend de récupérer ou de supprimer des messages d’une file d’attente d’invalidation partagée.

LWLock: SInval Écrire

Un processus attend d’ajouter un message à la file d’attente d’invalidation du catalogue partagé.

LWLock:SInvalWriteLock

Un processus attend d’ajouter un message dans une file d’attente d’invalidation partagée.

LWLock:SyncRep

Un processus attend de lire ou de mettre à jour des informations sur l’état de la réplication synchrone.

LWLock:SyncRepLock

Un processus attend de lire ou de mettre à jour des informations sur les réplicas synchrones.

LWLock:SyncScan

Un processus attend de sélectionner l’emplacement de début d’une analyse de table synchronisée.

LWLock:SyncScanLock

Un processus attend d'obtenir l'emplacement de début d'une analyse sur une table pour les analyses synchronisées.

LWLock:TablespaceCreate

Un processus est en attente de création ou de suppression d'un espace de table.

LWLock:TablespaceCreateLock

Un processus est en attente de création ou de suppression de l'espace de table.

LWLock:tbm

Un processus attend un verrou d'itérateur partagé sur un bitmap d'arborescence (TBM).

LWLock:TwoPhaseState

Un processus attend de lire ou de mettre à jour l'état des transactions préparées.

LWLock:TwoPhaseStateLock

Un processus attend de lire ou de mettre à jour l'état des transactions préparées.

LWLock:wal_insert

Un processus attend d'insérer le journal WAL dans un tampon de mémoire.

LWLock: WALBuf Cartographie

Un processus attend de remplacer une page dans les tampons WAL.

LWLock:WALBufMappingLock

Un processus attend de remplacer une page dans les tampons WAL.

LWLock:WALInsert

Un processus attend d'insérer les données WAL dans un tampon de mémoire.

LWLock:WALWrite

Un processus attend l'écriture de tampons WAL sur le disque.

LWLock: WALWrite Verrouiller

Un processus attend l'écriture de tampons WAL sur le disque.

LWLock:WrapLimitsVacuum

Un processus attend de mettre à jour les limites relatives à l'ID de transaction et à la consommation multixact.

LWLock:WrapLimitsVacuumLock

Un processus attend de mettre à jour les limites relatives à l'ID de transaction et à la consommation multixact.

LWLock:XactBuffer

Un processus attend l'état d'une transaction I/O sur une simple mémoire tampon utilisée le moins récemment (SLRU).

LWLock:XactSLRU

Un processus attend d'accéder au cache simple le moins récemment utilisé (SLRU) pour un état de transaction.

LWLock:XactTruncation

Un processus attend d'exécuter pg_xact_status ou de mettre à jour l'ID de transaction le plus ancien disponible.

LWLock:XidGen

Un processus attend d'allouer un nouvel ID de transaction.

LWLock:XidGenLock

Un processus attend d'allouer ou d'attribuer un ID de transaction.

Délai d'expiration : BaseBackupThrottle

Un processus est en attente pendant la sauvegarde de base lors de la limitation de l'activité.

Délai d'expiration : PgSleep

Un processus dorsal a appelé la fonction pg_sleep et attend l'expiration du délai de veille. Pour de plus amples informations, veuillez consulter Timeout:PgSleep.

Délai d'expiration : RecoveryApplyDelay

Un processus attend d'appliquer le journal WAL pendant la restauration en raison d'un paramètre de retard.

Délai d'expiration : RecoveryRetrieveRetryInterval

Un processus est en attente pendant la récupération lorsque les données WAL ne sont disponibles à partir d’aucune source (pg_wal, archive ou flux).

Délai d'expiration : VacuumDelay

Un processus est en attente dans un délai de vide basé sur les coûts.

Pour obtenir la liste complète des événements d’attente PostgreSQL, consultez The Statistics Collector > Wait Event tables dans la documentation de PostgreSQL.

Mises à jour du moteur de base de données pour Amazon Aurora PostgreSQL

Ci-dessous, vous trouverez des informations sur les versions et mises à jour du moteur Amazon Aurora PostgreSQL. Vous pouvez également découvrir comment mettre à niveau votre moteur Aurora PostgreSQL. Pour plus d'informations sur les versions d'Aurora en général, consultez Versions d'Amazon Aurora.

> **Tip**
>
> Vous pouvez minimiser la durée d'indisponibilité nécessaire à la mise à niveau d'un cluster de bases de données en utilisant un déploiement bleu/vert. Pour plus d'informations, consultez Utilisation des Blue/Green déploiements pour les mises à jour de bases de.

Rubriques

Identification des versions Amazon Aurora PostgreSQL

Amazon Aurora inclut certaines fonctions qui sont générales d'Aurora et disponibles pour tous les clusters de bases de données Aurora. Aurora inclut d'autres fonctions spécifiques d'un moteur de base de données particulier qu'Aurora prend en charge. Ces fonctions sont uniquement disponibles pour les clusters de bases de données Aurora qui utilisent ce moteur de base de données, comme Aurora PostgreSQL.

Une version de base de données Aurora a généralement deux numéros de version : le numéro de version du moteur de base de données et le numéro de version Aurora. Si une version Aurora PostgreSQL possède un numéro de version Aurora, il est inclus après le numéro de version du moteur dans la liste Versions Amazon Aurora PostgreSQL et versions du moteur.

Rubriques

Numéro de version Aurora

Les numéros de version Aurora utilisent le schéma de dénomination *major*.*minor*.*patch* (majeure.mineure.correctif). Une version de correctif Aurora inclut des corrections de bogues importantes apportées à une version mineure après sa publication. Pour en savoir plus sur les versions majeure, mineure et correctif d'Amazon Aurora, consultez Versions majeures d'Amazon Aurora, Versions mineures d'Amazon Aurora et Versions correctives d'Amazon Aurora.

Vous pouvez connaître le numéro de version Aurora de votre instance de base de données Aurora PostgreSQL avec la requête SQL suivante :

```
postgres=> SELECT aurora_version();
```

À partir de PostgreSQL versions 13.3, 12.8, 11.13, 10.18, et pour toutes les autres versions ultérieures, les numéros de version Aurora correspondent de manière plus précise à la version du moteur PostgreSQL. Par exemple, l'interrogation d'un cluster de bases de données Aurora PostgreSQL 13.3 renvoie ce qui suit :

```
aurora_version
----------------
 13.3.1
(1 row)
```

Les versions antérieures, telles que le cluster de bases de données Aurora PostgreSQL 10.14, renvoient des numéros de version similaires aux suivants :

```
aurora_version
----------------
 2.7.3
(1 row)
```

Numéros de version du moteur PostgreSQL

À partir de PostgreSQL 10, les versions du moteur de base de données PostgreSQL utilisent un schéma de numérotation *major*.*minor* pour toutes les versions. Voici quelques exemples : PostgreSQL 10.18, PostgreSQL 12.7 et PostgreSQL 13.3.

Les versions antérieures à PostgreSQL 10 utilisaient un schéma de numérotation *major*.*major*.*minor* dans lequel les deux premiers chiffres constituent le numéro de version majeure et un troisième chiffre indique une version mineure. Par exemple, PostgreSQL 9.6 était une version majeure, les versions mineures 9.6.21 ou 9.6.22 étant indiquées par le troisième chiffre.

> **Note**
>
> Le moteur de version PostgreSQL 9.6 n'est plus pris en charge. Pour effectuer une mise à niveau, consultez Mise à niveau des clusters de base de données Amazon Aurora PostgreSQL. Pour les politiques de version et les calendriers de publication, consultez Durée de disponibilité des versions majeures d'Amazon Aurora.

Vous pouvez trouver le numéro de version du moteur de base de données PostgreSQL à l'aide de la requête SQL suivante :

```
postgres=> SELECT version();
```

Pour un cluster de bases de données Aurora PostgreSQL 13.3, les résultats sont les suivants :

```
version
------------------------------------------------------------------------------------------------------------
 PostgreSQL 13.3 on x86_64-pc-linux-gnu, compiled by x86_64-pc-linux-gnu-gcc (GCC)
 7.4.0, 64-bit
(1 row)
```

Versions Amazon Aurora PostgreSQL et versions du moteur

Les versions d'Amazon Aurora Édition compatible avec PostgreSQL sont régulièrement mises à jour. Les mises à jour sont appliquées aux clusters de base de données Aurora PostgreSQL pendant les fenêtres de maintenance du système. Le moment où les mises à jour sont appliquées dépend du type de mise à jour, de la Région AWS et du paramètre de fenêtre de maintenance du cluster de base de données. La plupart des versions répertoriées incluent à la fois un numéro de version PostgreSQL et un numéro de version Amazon Aurora. Cependant, à partir de PostgreSQL versions 13.3, 12.8, 11.13, 10.18, et pour toutes les autres versions ultérieures, les numéros de version Aurora ne sont pas utilisés. Pour identifier les numéros de version de votre base de données Aurora PostgreSQL, consultez Identification des versions Amazon Aurora PostgreSQL.

Pour en savoir plus sur les extensions et les modules, consultez Versions d'extension pour Amazon Aurora PostgreSQL.

> **Note**
>
> Pour obtenir plus d'informations sur les politiques de version et les calendriers de publication d'Amazon Aurora, consultez Durée de disponibilité des versions majeures d'Amazon Aurora.

Pour plus d'informations sur la prise en charge d'Amazon Aurora, consultez la section Amazon RDS FAQs (FAQ Amazon RDS).

Pour déterminer les versions du moteur de base de données Aurora PostgreSQL disponibles dans une Région AWS, utilisez la commande describe-db-engine-versions de l'AWS CLI. Par exemple :

```
aws rds describe-db-engine-versions --engine aurora-postgresql --query '*[].
[EngineVersion]' --output text --region aws-region
```

Pour obtenir la liste des Régions AWS, consultez Aurora PostgreSQL : disponibilité dans les régions.

Pour obtenir plus de détails sur les versions de PostgreSQL disponibles pour Aurora PostgreSQL, consultez Release Notes for Aurora PostgreSQL (Notes de mise à jour de Aurora PostgreSQL).

Versions d'extension pour Amazon Aurora PostgreSQL

Vous pouvez installer et configurer diverses extensions PostgreSQL à utiliser avec les clusters de base de données Aurora PostgreSQL. Par exemple, vous pouvez utiliser l'extension `pg_partman` PostgreSQL pour automatiser la création et la maintenance des partitions de table. Pour en savoir plus sur cette extension et les autres extensions disponibles pour Aurora PostgreSQL, consultez Utilisation d'extensions avec encapsuleurs de données externes.

Pour obtenir plus de détails sur les extensions PostgreSQL prises en charge par Aurora PostgreSQL, consultez Extension versions for Amazon Aurora PostgreSQL (Versions des extensions pour Amazon Aurora PostgreSQL) dans Release Notes for Aurora PostgreSQL (Notes de mise à jour pour Aurora PostgreSQL).

Mise à niveau des clusters de base de données Amazon Aurora PostgreSQL

Amazon Aurora met à disposition de nouvelles versions du moteur de base de données PostgreSQL dans les Régions AWS seulement après des tests approfondis. Vous pouvez mettre à niveau vos clusters de bases de données Aurora PostgreSQL vers la nouvelle version lorsqu'elle est disponible dans votre région.

Selon la version d'Aurora PostgreSQL actuellement exécutée par votre cluster de bases de données, une mise à niveau vers la nouvelle version est soit une mise à niveau mineure, soit une mise à niveau majeure. Par exemple, la mise à niveau d'un cluster de bases de données Aurora PostgreSQL 11.15 vers Aurora PostgreSQL 13.6 est une mise à niveau de version majeure. La mise à niveau d'un cluster de bases de données Aurora PostgreSQL 13.3 vers Aurora PostgreSQL 13.7 est une mise à niveau de version mineure. Dans les rubriques suivantes, vous apprendrez comment effectuer les deux types de mises à niveau.

Table des matières

Présentation des processus de mise à niveau Aurora PostgreSQL

Les différences entre les mises à niveau des versions majeures et mineures sont les suivantes :

Mises à niveau des versions mineures et correctifs

Les mises à niveau de versions mineures et les correctifs contiennent uniquement des modifications rétrocompatibles avec les applications existantes. Les mises à niveau des versions mineures et les correctifs ne sont disponibles qu'une fois qu'Aurora PostgreSQL les a testés et approuvés.

Aurora peut automatiquement appliquer à votre place les mises à niveau mineures. Lorsque vous créez un cluster de bases de données Aurora PostgreSQL, l'option Activer la mise à niveau des versions mineures est activée par défaut. À moins de désactiver manuellement cette option, Aurora applique régulièrement des mises à niveau automatiques des versions mineures durant votre fenêtre de maintenance planifiée. Pour plus d'informations sur l'option de mise à niveau automatique des versions mineures (AmVU) et sur la façon de modifier votre cluster de bases de données Aurora pour l'utiliser, consultez Mises à niveau automatiques des versions mineures pour les clusters de bases de données Aurora.

Si l'option de mise à niveau automatique des versions mineures n'est pas définie pour votre cluster de bases de données Aurora PostgreSQL, ce dernier n'est pas mis à niveau automatiquement vers la nouvelle version mineure. Au lieu de cela, lorsqu'une nouvelle version mineure est publiée dans votre Région AWS et que votre cluster de bases de données Aurora PostgreSQL exécute une version mineure plus ancienne, Aurora vous invite à le mettre à niveau. Pour ce faire, il ajoute une recommandation aux tâches de maintenance de votre cluster.

Les correctifs ne sont pas considérés comme une mise à niveau et ils ne sont pas appliqués automatiquement. Aurora PostgreSQL vous invite à appliquer les éventuels correctifs en ajoutant une recommandation aux tâches de maintenance de votre cluster de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez Comment effectuer des mises à niveau de versions mineures et appliquer des correctifs.

> **Note**
>
> Les correctifs qui résolvent les problèmes de sécurité ou d'autres problèmes critiques sont également ajoutés en tant que tâches de maintenance. Ces correctifs sont toutefois obligatoires. Assurez-vous d'appliquer les correctifs de sécurité à votre cluster de bases de données Aurora PostgreSQL lorsqu'ils sont mis à disposition dans vos tâches de maintenance en attente.
> Des mises à niveau automatiques de version mineure sont effectuées vers la version mineure par défaut.

Il est possible que de courtes pannes se produisent pendant le processus de mise à niveau car chaque instance du cluster est mise à niveau vers la nouvelle version. Cependant, après Aurora PostgreSQL 14.3.3, 13.7.3, 12.11.3, 11.16.3, 10.21.3, et d'autres versions ultérieures de ces versions mineures et les nouvelles versions majeures, le processus de mise à niveau utilise la fonction ZDP (application de correctifs sans durée d'indisponibilité). Cette fonctionnalité réduit les pannes et les élimine complètement dans la plupart des cas. Pour plus d'informations, consultez Mises à niveau de versions mineures et application de correctifs sans durée d'indisponibilité. Pour plus d'informations sur les fonctionnalités prises en charge et les limites de ZDP, consultez Limites des correctifs sans durée d'indisponibilité.

Mises à niveau de version majeure.

Contrairement aux mises à niveau et aux correctifs des versions mineures, Aurora PostgreSQL ne dispose pas d'une option de mise à niveau automatique des versions majeures. Les nouvelles versions majeures de PostgreSQL peuvent contenir des modifications de base de données qui ne sont pas rétrocompatibles avec les applications existantes. Les nouvelles fonctionnalités peuvent empêcher vos applications existantes de fonctionner correctement.

Pour éviter tout problème, nous vous recommandons vivement de suivre le processus décrit dans Test d'une mise à niveau de votre cluster de bases de données de production vers une nouvelle version majeure avant de mettre à niveau les instances de base de données de vos clusters de base de données Aurora PostgreSQL. Assurez-vous tout d'abord que vos applications peuvent s'exécuter sur la nouvelle version en procédant comme suit. Vous pouvez ensuite mettre à niveau manuellement votre cluster de bases de données Aurora PostgreSQL vers la nouvelle version.

Il est possible que de courtes pannes se produisent pendant la mise à niveau vers la nouvelle version de toutes les instances du cluster. Le processus de planification préliminaire prend également un certain temps. Nous vous recommandons de toujours effectuer les tâches de mise à niveau pendant la fenêtre de maintenance de votre cluster ou lorsque la charge d'opérations est minimale. Pour plus d'informations, consultez Réalisation d'une mise à niveau de version majeure.

> **Note**
>
> Les mises à niveau de versions mineures et de versions majeures peuvent impliquer de courtes pannes. Nous vous recommandons ainsi vivement d'effectuer ou de planifier vos mises à niveau pendant votre fenêtre de maintenance ou pendant les périodes de faible utilisation.

Les clusters de bases de données Aurora PostgreSQL nécessitent parfois des mises à jour du système d'exploitation. Ces mises à jour peuvent inclure une version plus récente de la bibliothèque glibc. Lors de ces mises à jour, nous vous recommandons de suivre les directives décrites dans Les classements pris en charge dans Aurora PostgreSQL.

Obtenir une liste des versions disponibles dans votre Région AWS

Vous pouvez obtenir une liste de toutes les versions de moteur disponibles en tant que cibles de mise à niveau pour votre cluster de base de données Aurora PostgreSQL en Région AWS vous interrogeant à l'aide de describe-db-engine-versionsAWS CLIla commande suivante.

Pour Linux, macOS ou Unix :

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version version-number \
  --query 'DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}' \
  --output text
```

Pour Windows :

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version version-number ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
  --output text
```

Par exemple, pour identifier les cibles de mise à niveau valides pour un cluster de base de données Aurora PostgreSQL version 12.10, exécutez la commande suivante : AWS CLI

Pour Linux, macOS ou Unix :

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version 12.10 \
  --query 'DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}' \
  --output text
```

Pour Windows :

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version 12.10 ^
  --query "DBEngineVersions[*].ValidUpgradeTarget[*].{EngineVersion:EngineVersion}" ^
  --output text
```

Ce tableau indique les cibles de mise à niveau des versions majeures et mineures vers différentes versions de base de données Aurora PostgreSQL. Pour garantir la compatibilité, toutes les versions ne sont pas proposées comme cibles de mise à niveau. Aurora PostgreSQL introduit de nouvelles fonctionnalités et des corrections de bogues à chaque publication trimestrielle de version mineure. Pour plus d'informations sur les versions mineures d'Aurora PostgreSQL, consultez Notes de mise à jour d'Aurora PostgreSQL.

Version source actuelle	Cibles de mise à niveau
17,7	Aucune
17,6	17,7
17,5	17,7, 17,6
17,4	17,7, 17,6, 17,5
16,11	Aucune
16,10	17,7 16,11
16,9	17,7 16,11, 17,6, 16,10, 17,5
16,8	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9
16,6	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9, 16,8

Version source actuelle	Cibles de mise à niveau
16,4	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9, 16,8, 16,6
16,3	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9, 16,8, 16,6, 16,4
16,2	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9, 16,8, 16,6, 16,4, 16,3
16,1	17,7 16,11 , 17,6, 16,10, 17,5, 17,4, 16,9, 16,8, 16,6, 16,4, 16,3, 16,2
15,15	Aucune
15,14	17,7 16,11 15,15
15,13	17,7 16,11 , 15,15, 17,6, 16,10, 15,14, 17,5, 16,9
15,12	17,7 16,11 , 15,15, 17,6, 16,10, 15,14, 15,13
15,10	17,7 16,11 , 15,15, 17,6, 16,10, 15,14, 17,5, 17,4, 16,9, 16,8, 15,13
15,11	17,7 16,11 , 15,15, 17,6, 16,10, 15,14, 17,5, 17,4, 16,9, 16,8, 16,6, 15,13, 15,12

Version source actuelle	Cibles de mise à niveau
14,20	Aucune

Pour toute version que vous envisagez d'utiliser, vérifiez toujours la disponibilité de la classe d'instance de base de données de votre cluster. Par exemple, `db.r4` n'est pas pris en charge pour Aurora PostgreSQL 13. Si votre cluster de base de données Aurora PostgreSQL utilise actuellement une classe d'instance db.r4, vous devez la modifier pour utiliser une classe d'instance de base de données prise en charge avant de pouvoir passer à Aurora PostgreSQL 13. Pour plus d'informations sur les classes d'instance de base de données Aurora PostgreSQL disponibles, notamment celles basées sur Graviton2 et celles basées sur Intel, consultez. Classes d'instance de base de données Amazon Aurora

Réalisation d'une mise à niveau de version majeure

Les mises à niveau de version majeure risquent de contenir des modifications de base de données qui ne sont pas rétrocompatibles avec les versions antérieures de la base de données. Les nouvelles fonctionnalités d'une nouvelle version peuvent empêcher vos applications existantes de fonctionner correctement. Pour éviter les problèmes, Amazon Aurora n'applique pas les mises à niveau de versions majeures automatiquement. Nous vous recommandons plutôt de planifier soigneusement une mise à niveau de version majeure en procédant comme suit :

1. Choisissez la version majeure souhaitée dans la liste des cibles disponibles parmi celles répertoriées pour votre version dans le tableau. Vous pouvez obtenir une liste précise des versions disponibles dans votre Région AWS pour votre version actuelle à l'aide d'AWS CLI. Pour en savoir plus, consultez Obtenir une liste des versions disponibles dans votre Région AWS.
2. Vérifiez que vos applications fonctionnent comme prévu lors d'un déploiement d'essai de la nouvelle version. Pour plus d'informations sur le processus complet, consultez Test d'une mise à niveau de votre cluster de bases de données de production vers une nouvelle version majeure.
3. Après avoir vérifié que vos applications fonctionnent comme prévu lors du déploiement d'essai, vous pouvez mettre à niveau votre cluster. Pour en savoir plus, consultez Mise à niveau du moteur Aurora PostgreSQL vers une nouvelle version majeure.

> **Note**
>
> Vous pouvez effectuer une mise à niveau de la version majeure de Babelfish pour Aurora PostgreSQL des versions 13 (à partir de 13.6) aux versions basées sur Aurora PostgreSQL 14 (à partir de 14.6). Babelfish pour Aurora PostgreSQL versions 13.4 et 13.5 ne prend pas en charge les mises à niveau de versions majeures.

Vous pouvez obtenir la liste des versions de moteur disponibles en tant que cibles de mise à niveau de version majeure pour votre cluster de bases de données Aurora PostgreSQL en interrogeant votre Région AWS à l'aide de la commande AWS CLI describe-db-engine-versions, comme suit.

Pour Linux, macOS ou Unix :

```
aws rds describe-db-engine-versions \
  --engine aurora-postgresql \
  --engine-version version-number \
  --query 'DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`].
{EngineVersion:EngineVersion}' \
  --output text
```

Pour Windows :

```
aws rds describe-db-engine-versions ^
  --engine aurora-postgresql ^
  --engine-version version-number ^
  --query "DBEngineVersions[].ValidUpgradeTarget[?IsMajorVersionUpgrade == `true`].
{EngineVersion:EngineVersion}" ^
  --output text
```

Dans certains cas, la version vers laquelle vous souhaitez effectuer une mise à niveau n'est pas une cible pour votre version actuelle. Dans ce cas, utilisez les informations du versions table pour effectuer des mises à niveau de versions mineures jusqu'à ce que votre cluster atteigne une version dont la cible choisie figure sur sa ligne de cibles.

Test d'une mise à niveau de votre cluster de bases de données de production vers une nouvelle version majeure

Chaque nouvelle version majeure comprend des améliorations de l'optimiseur de requêtes destinées à améliorer les performances. Cependant, votre charge de travail peut inclure des requêtes qui

donnent lieu à un plan moins performant dans la nouvelle version. C'est pourquoi nous vous recommandons de tester et d'examiner les performances avant de procéder à la mise à niveau en production. Vous pouvez gérer la stabilité du plan de requête entre les versions en utilisant l'extension Query Plan Management (QPM), comme indiqué dans Assurer la stabilité du plan après une mise à niveau majeure de la version.

Avant de mettre à niveau vos clusters de base de données Aurora PostgreSQL de production vers une nouvelle version majeure, nous vous recommandons vivement de tester la mise à niveau pour vérifier que toutes vos applications fonctionnent correctement :

1. Préparez un groupe de paramètres compatible avec la version.

   Si vous utilisez une instance de base de données personnalisée ou un groupe de paramètres de cluster de bases de données, vous pouvez choisir parmi deux options :

   a. Spécifiez l'instance de base de données par défaut, le groupe de paramètres de cluster de bases de données ou ces deux éléments pour la nouvelle version du moteur de base de données.

   b. Créez votre propre groupe de paramètres personnalisé pour la nouvelle version du moteur de base de données.

2. Vérifiez les bases de données non valides et supprimez celles qui existent.

   La colonne `datconnlimit` du catalogue `pg_database` inclut une valeur de `-2` pour marquer comme non valides les bases de données qui ont été interrompues au cours d'une opération `DROP DATABASE`. Utilisez la requête suivante pour vérifier la présence de bases de données non valides :

   ```
   SELECT
       datname
   FROM
       pg_database
   WHERE
       datconnlimit = - 2;
   ```

   Si la requête renvoie des noms de base de données, ces bases de données ne sont pas valides. Utilisez l'instruction `DROP DATABASE invalid_db_name` pour supprimer les bases de données non valides. Vous pouvez utiliser l'instruction dynamique suivante pour supprimer toutes les bases de données non valides.

```
SELECT
    'DROP DATABASE ' || quote_ident(datname) || ';'
FROM
    pg_database
WHERE
    datconnlimit = -2 \gexec
```

3. Recherchez une utilisation non prise en charge :

   - Validez ou restaurez toutes les transactions préparées ouvertes avant d'essayer d'effectuer une mise à niveau. Vous pouvez utiliser la requête suivante pour vérifier qu'aucune transaction préparée ouverte ne figure sur votre instance.

     ```
     SELECT count(*) FROM pg_catalog.pg_prepared_xacts;
     ```

   - Supprimez toutes les utilisations des types de données reg* avant d'essayer d'effectuer une mise à niveau. À l'exception de `regtype` et `regclass`, vous ne pouvez pas mettre à niveau les types de données reg*. L'utilitaire pg_upgrade (utilisé par Amazon Aurora pour effectuer la mise à niveau) ne peut pas conserver ce type de données. Pour en savoir plus sur cet utilitaire, consultez pg_upgrade dans la documentation PostgreSQL.

     Pour vérifier l'absence de toute utilisation des types de données reg* non pris en charge, utilisez la requête suivante pour chaque base de données.

     ```
     SELECT count(*) FROM pg_catalog.pg_class c, pg_catalog.pg_namespace n,
      pg_catalog.pg_attribute a
       WHERE c.oid = a.attrelid
           AND NOT a.attisdropped
           AND a.atttypid IN ('pg_catalog.regproc'::pg_catalog.regtype,
                              'pg_catalog.regprocedure'::pg_catalog.regtype,
                              'pg_catalog.regoper'::pg_catalog.regtype,
                              'pg_catalog.regoperator'::pg_catalog.regtype,
                              'pg_catalog.regconfig'::pg_catalog.regtype,
                              'pg_catalog.regdictionary'::pg_catalog.regtype)
           AND c.relnamespace = n.oid
           AND n.nspname NOT IN ('pg_catalog', 'information_schema');
     ```

   - Si vous mettez à niveau un cluster de bases de données Aurora PostgreSQL version 10.18 ou supérieure sur lequel l'extension `pgRouting` est installée, retirez l'extension avant de mettre à niveau vers la version 12.4 ou supérieure.

Si vous mettez à niveau une version d'Aurora PostgreSQL 10.x sur laquelle l'extension `pg_repack` version 1.4.3 est installée, supprimez l'extension avant d'effectuer la mise à niveau vers une version ultérieure.

4. Vérifiez les bases de données template1 et template0.

   Pour une mise à niveau réussie, les bases de données des modèles 1 et 0 doivent exister et doivent être répertoriées en tant que modèles. Pour vérifier cela, utilisez la commande suivante :

   ```
   SELECT datname, datistemplate FROM pg_database;

   datname    | datistemplate
   -----------+---------------
   template0  | t
   rdsadmin   | f
   template1  | t
   postgres   | f
   ```

   Dans la sortie de la commande, la valeur `datistemplate` des bases de données template1 et template0 doit être `t`.

5. Supprimez des emplacements logiques de réplication.

   Le processus de mise à niveau ne peut pas avoir lieu si le cluster de bases de données Aurora PostgreSQL utilise des emplacements de réplication logiques. Les emplacements de réplication logiques sont généralement utilisés pour des tâches de migration de données à court terme, telles que la migration de données à l'aide de AWS DMS ou pour la réplication de tables de la base de données vers des lacs de données, des outils de BI ou d'autres cibles. Avant de procéder à la mise à niveau, assurez-vous de connaître l'utilité de tout emplacement de réplication logique existant et confirmez que vous pouvez les supprimer. Vous pouvez vérifier les emplacements de réplication logiques à l'aide de la requête suivante :

   ```
   SELECT * FROM pg_replication_slots;
   ```

   Si les emplacements de réplication logique sont toujours utilisés, vous ne devez pas les supprimer et vous ne pouvez pas procéder à la mise à niveau. Toutefois, si les emplacements de réplication logique ne sont pas nécessaires, vous pouvez les supprimer à l'aide du code SQL suivant :

   ```
   SELECT pg_drop_replication_slot(slot_name);
   ```

Les scénarios de réplication logique qui utilisent l'extension `pglogical` doivent également avoir des emplacements supprimés du nœud de serveur de publication pour que la mise à niveau de version majeure réussisse sur ce nœud. Toutefois, vous pouvez redémarrer le processus de réplication à partir du nœud d'abonné après la mise à niveau. Pour plus d'informations, consultez Rétablissement de la réplication logique après une mise à niveau majeure.

6. Effectuez une sauvegarde.

   Le processus de mise à niveau crée un instantané de cluster de bases de données de votre cluster de bases de données lors de la mise à niveau. Si vous souhaitez également effectuer une sauvegarde manuelle avant le processus de mise à niveau, consultez Création d'un instantané de cluster de bases de données pour de plus amples informations.

7. Mettez à niveau certaines extensions vers la dernière version disponible avant d'effectuer la mise à niveau de la version majeure. Les extensions à mettre à jour sont les suivantes :

   - `pgRouting`
   - `postgis_raster`
   - `postgis_tiger_geocoder`
   - `postgis_topology`
   - `address_standardizer`
   - `address_standardizer_data_us`

   Exécutez la commande suivante pour chaque extension actuellement installée.

   ```
   ALTER EXTENSION PostgreSQL-extension UPDATE TO 'new-version';
   ```

   Pour plus d'informations, consultez Mise à niveau des extensions PostgreSQL. Pour en savoir plus sur la mise à niveau de PostGIS, consultez Étape 6 : Mettre à niveau l'extension PostGIS.

8. Si vous effectuez une mise à niveau vers la version 11.x, supprimez les extensions que celle-ci ne prend pas en charge avant d'effectuer la mise à niveau de la version majeure. Les extensions à supprimer sont :

   - `chkpass`
   - `tsearch2`

9. Supprimez les types de données `unknown` en fonction de votre version cible.

PostgreSQL version 10 ne prend pas en charge le type de données `unknown`. Si une base de données version 9.6 utilise le type de données `unknown`, une mise à niveau vers la version 10 affiche un message d'erreur semblable au suivant :

```
Database instance is in a state that cannot be upgraded: PreUpgrade checks failed:
The instance could not be upgraded because the 'unknown' data type is used in user
 tables.
Please remove all usages of the 'unknown' data type and try again."
```

Pour rechercher le type de données `unknown` dans votre base de données afin de pouvoir supprimer de telles colonnes ou les remplacer par un type de données pris en charge, utilisez le code SQL suivant pour chaque base de données.

```
SELECT n.nspname, c.relname, a.attname
    FROM pg_catalog.pg_class c,
    pg_catalog.pg_namespace n,
    pg_catalog.pg_attribute a
    WHERE c.oid = a.attrelid AND NOT a.attisdropped AND
    a.atttypid = 'pg_catalog.unknown'::pg_catalog.regtype AND
    c.relkind IN ('r','m','c') AND
    c.relnamespace = n.oid AND
    n.nspname !~ '^pg_temp_' AND
    n.nspname !~ '^pg_toast_temp_' AND n.nspname NOT IN ('pg_catalog',
 'information_schema');
```

10. Procédez à un essai de mise à niveau.

    Nous vous recommandons vivement de tester la mise à niveau de version majeure sur une copie de votre base de données de production avant d'essayer d'effectuer la mise à niveau sur votre base de données de production. Vous pouvez surveiller les plans d'exécution sur l'instance de test dupliquée pour détecter d'éventuelles régressions du plan d'exécution et évaluer ses performances. Pour créer une copie d'une instance pour les tests, vous pouvez restaurer votre base de données à partir d'un instantané récent ou cloner votre base de données. Pour plus d'informations, consultez Restaurer à partir d'un instantané ou Clonage d'un volume pour un cluster de bases de données Amazon Aurora.

    Pour plus d'informations, consultez Mise à niveau du moteur Aurora PostgreSQL vers une nouvelle version majeure.

11. Mettez à niveau votre instance de production.

Lorsque l'essai de mise à niveau de version majeure est un succès, vous pouvez mettre à niveau en toute confiance votre base de données de production. Pour plus d'informations, consultez Mise à niveau du moteur Aurora PostgreSQL vers une nouvelle version majeure.

> **Note**
>
> Lors d'un processus de mise à niveau, Aurora PostgreSQL prend un instantané du cluster de bases de données. Vous ne pouvez pas effectuer une restauration ponctuelle de votre cluster pendant ce processus. Plus tard, vous pouvez effectuer une restauration ponctuelle à des moments antérieurs au début de la mise à niveau et après la fin de l'instantané automatique de votre instance. Cependant, vous ne pouvez pas effectuer une restauration ponctuelle vers une version mineure antérieure.

Pour obtenir des informations sur une mise à niveau en cours, vous pouvez utiliser Amazon RDS for afficher deux journaux produits par l'utilitaire pg_upgrade. Il s'agit des journaux `pg_upgrade_internal.log` et `pg_upgrade_server.log`. Amazon Aurora ajoute un horodatage au nom de fichier de ces journaux. Vous pouvez consultez ces journaux comme tout autre journal. Pour plus d'informations, consultez Surveillance des fichiers journaux Amazon Aurora.

12. Mettez à niveau les extensions PostgreSQL. Le processus de mise à niveau de PostgreSQL ne met à niveau aucune extension PostgreSQL. Pour plus d'informations, consultez Mise à niveau des extensions PostgreSQL.

Recommandations après la mise à niveau

Après avoir effectué une mise à niveau majeure de la version, nous vous recommandons de procéder comme suit :

- Exécutez l'opération `ANALYZE` pour actualiser la table `pg_statistic`. Vous devez le faire pour chaque base de données de toutes vos instances de base de données PostgreSQL. Les statistiques de l'optimiseur ne sont pas transférées lors d'une mise à niveau majeure de la version. Vous devez donc régénérer toutes les statistiques pour éviter les problèmes de performances. Exécutez la commande sans paramètres pour générer des statistiques pour toutes les tables normales de la base de données actuelle, comme suit :

```
ANALYZE VERBOSE;
```

L'indicateur VERBOSE est facultatif, mais son utilisation vous montre la progression. Pour en savoir plus, consultez ANALYZE dans la documentation PostgreSQL.

Lorsque vous analysez des tables spécifiques au lieu d'utiliser ANALYZE VERBOSE, exécutez la commande ANALYZE pour chaque table comme suit :

```
ANALYZE table_name;
```

Pour les tables partitionnées, analysez toujours la table parent. Ce processus :

- Échantillonne automatiquement les lignes de toutes les partitions
- Met à jour les statistiques de chaque partition de manière récursive
- Maintient les statistiques de planification essentielles au niveau des parents

Bien que les tables parents ne stockent aucune donnée réelle, leur analyse est essentielle pour l'optimisation des requêtes. L'exécution d'ANALYZE uniquement sur des partitions individuelles peut nuire aux performances des requêtes, car l'optimiseur ne dispose pas des statistiques complètes nécessaires à une planification efficace entre partitions.

> **Note**
>
> Exécutez ANALYZE sur votre système après la mise à niveau pour éviter les problèmes de performances.

- Si vous avez effectué une mise à niveau vers PostgreSQL version 10, exécutez REINDEX sur tous les index de hachage dont vous disposez. Les index de hachage ont été modifiés dans la version 10 et doivent être reconstruits. Pour localiser des index de hachage non valides, exécutez le code SQL suivant pour chaque base de données contenant des index de hachage.

```
SELECT idx.indrelid::regclass AS table_name,
   idx.indexrelid::regclass AS index_name
FROM pg_catalog.pg_index idx
   JOIN pg_catalog.pg_class cls ON cls.oid = idx.indexrelid
   JOIN pg_catalog.pg_am am ON am.oid = cls.relam
WHERE am.amname = 'hash'
AND NOT idx.indisvalid;
```

- Nous vous recommandons de tester votre application sur la base de données mise à niveau avec une charge de travail similaire afin de vérifier que tout fonctionne comme prévu. Une fois la mise à niveau vérifiée, vous pouvez supprimer l'instance de test.

Mise à niveau du moteur Aurora PostgreSQL vers une nouvelle version majeure

Lorsque vous lancez le processus de mise à niveau vers une nouvelle version majeure, Aurora PostgreSQL prend un instantané du cluster de bases de données Aurora avant d'apporter des modifications à votre cluster. Cet instantané est créé uniquement pour les mises à niveau de versions majeures, pas pour les mises à niveau de versions mineures. Lorsque le processus de mise à niveau est terminé, cet instantané figure parmi les instantanés manuels répertoriés sous Snapshots (Instantanés) dans la console RDS. Le nom de l'instantané inclut le préfixe `preupgrade`, le nom de votre cluster de bases de données Aurora PostgreSQL, la version source, la version cible, ainsi que la date et l'horodatage, comme illustré dans l'exemple suivant.

```
preupgrade-docs-lab-apg-global-db-12-8-to-13-6-2022-05-19-00-19
```

Une fois la mise à niveau terminée, vous pouvez utiliser l'instantané créé et stocké par Aurora dans votre liste d'instantanés manuels pour restaurer le cluster de bases de données vers sa version précédente, si nécessaire.

> **Tip**
>
> En général, les instantanés offrent de nombreuses façons de restaurer votre cluster de bases de données Aurora à différents moments. Pour en savoir plus, consultez Restauration à partir d'un instantané de cluster de bases de données et Restauration d'un cluster de bases de données à une date définie. Toutefois, Aurora PostgreSQL ne prend pas en charge l'utilisation d'un instantané pour restaurer une version mineure antérieure.

Au cours du processus de mise à niveau de version majeure, Aurora alloue un volume et clone le cluster de bases de données Aurora PostgreSQL source. Si la mise à niveau échoue pour une raison quelconque, Aurora PostgreSQL utilise le clone pour annuler la mise à niveau. Lorsque plus de 15 clones d'un volume source sont alloués, les clones suivants deviennent des copies complètes et prennent plus de temps. Cela peut également allonger le temps nécessaire au processus de mise à niveau. Si Aurora PostgreSQL annule la mise à niveau, sachez que :

- Il se pourrait que des entrées de facturation et des métriques apparaissent pour le volume d'origine et le volume cloné alloué lors de la mise à niveau. Aurora PostgreSQL nettoie le volume supplémentaire une fois que la fenêtre de rétention de la sauvegarde du cluster dépasse l'échéance de la mise à niveau.
- La copie de l'instantané entre régions suivante à partir de ce cluster sera une copie complète au lieu d'une copie incrémentielle.

Pour mettre à niveau en toute sécurité les instances de base de données qui composent votre cluster, Aurora PostgreSQL utilise l'utilitaire pg_upgrade. Une fois la mise à niveau de l'enregistreur terminée, chaque instance de lecteur subit une brève panne pendant sa mise à niveau vers la nouvelle version majeure. Pour en savoir plus sur cet utilitaire PostgreSQL, consultez pg_upgrade dans la documentation PostgreSQL.

Vous pouvez mettre à niveau votre cluster de bases de données Aurora PostgreSQL vers une nouvelle version à l'aide d'AWS Management Console, d'AWS CLI ou de l'API RDS.

Console

Pour mettre à niveau la version de moteur d'un cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis le cluster de bases de données que vous souhaitez mettre à niveau.
3. Sélectionnez Modify. La page Modify DB cluster (Modifier le cluster DB) s'affiche.
4. Dans le champ Version du moteur, sélectionnez la nouvelle version.
5. Choisissez Continuer et vérifiez le récapitulatif des modifications.
6. Pour appliquer les modifications immédiatement, choisissez Appliquer immédiatement. La sélection de cette option peut entraîner une interruption de service dans certains cas. Pour plus d'informations, consultez Modification d'un cluster de bases de données Amazon Aurora.
7. Sur la page de confirmation, examinez vos modifications. Si elles sont correctes, choisissez Modifier le cluster pour enregistrer vos modifications.

   Ou choisissez Retour pour revoir vos modifications, ou choisissez Annuler pour les annuler.

AWS CLI

Pour mettre à niveau la version de moteur d'un cluster de bases de données, utilisez la commande AWS CLI modify-db-cluster. Spécifiez les paramètres suivants :

- `--db-cluster-identifier` : nom du cluster de bases de données.
- `--engine-version` : numéro de version du moteur de base de données vers lequel effectuer la mise à niveau. Pour obtenir des informations sur les versions de moteur valides, utilisez la commande describe-db-engine-versions de l'AWS CLI.
- `--allow-major-version-upgrade` : indicateur obligatoire lorsque le paramètre `--engine-version` est une version majeure différente de la version majeure actuelle du cluster de bases de données.
- `--no-apply-immediately` : appliquer les modifications pendant le créneau de maintenance suivant. Pour appliquer les modifications immédiatement, utilisez `--apply-immediately`.

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine-version new_version \
    --allow-major-version-upgrade \
    --no-apply-immediately
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine-version new_version ^
    --allow-major-version-upgrade ^
    --no-apply-immediately
```

API RDS

Pour mettre à niveau la version du moteur d'un cluster de bases de données, utilisez l'opération ModifyDBCluster. Spécifiez les paramètres suivants :

- `DBClusterIdentifier` : nom du cluster de bases de données, par exemple *mydbcluster*.

- `EngineVersion` : numéro de version du moteur de base de données vers lequel effectuer la mise à niveau. Pour obtenir des informations sur les versions de moteur valides, utilisez l'opération DescribeDBEngineVersions.
- `AllowMajorVersionUpgrade` : indicateur obligatoire lorsque le paramètre `EngineVersion` est une version majeure différente de la version majeure actuelle du cluster de bases de données.
- `ApplyImmediately` : si des modifications doivent être appliquées immédiatement ou au cours du prochain créneau de maintenance. Pour appliquer les modifications immédiatement, définissez la valeur sur `true`. Pour appliquer les modifications pendant le créneau de maintenance suivant, définissez la valeur sur `false`.

Mises à niveau majeures des bases de données globales

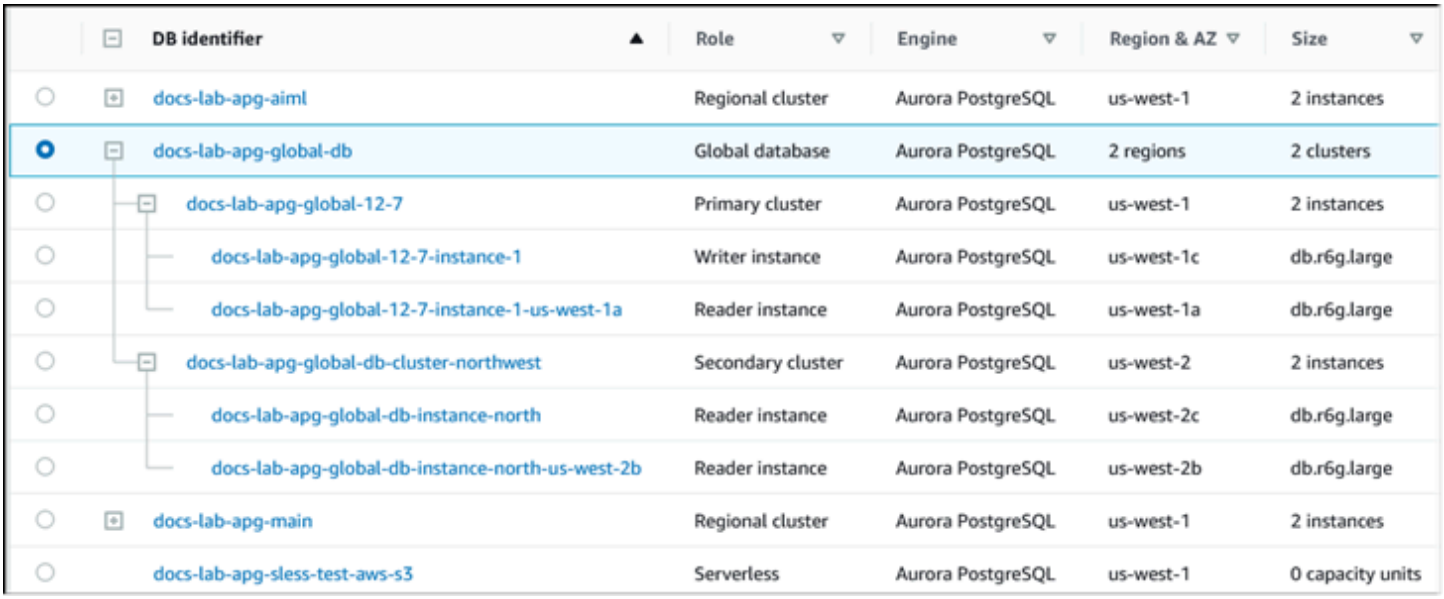
Pour un cluster de bases de données global Aurora, le processus de mise à niveau met à niveau tous les clusters de bases de données qui composent votre base de données globale Aurora en même temps. Cela garantit que chaque cluster exécute la même version d'Aurora PostgreSQL. Cela garantit également que toutes les modifications apportées aux tables système, aux formats de fichiers de données et autres éléments sont automatiquement répliquées sur tous les clusters secondaires.

Pour mettre à niveau un cluster de bases de données global vers une nouvelle version majeure d'Aurora PostgreSQL, nous vous recommandons de tester vos applications sur la version mise à niveau, comme décrit dans Test d'une mise à niveau de votre cluster de bases de données de production vers une nouvelle version majeure. Assurez-vous de préparer les paramètres du groupe de paramètres de cluster de bases de données et du groupe de paramètres de base de données pour chaque Région AWS dans votre base de données globale Aurora avant la mise à niveau, comme décrit à l'step 1. de la section Test d'une mise à niveau de votre cluster de bases de données de production vers une nouvelle version majeure.

Si votre cluster de bases de données global Aurora PostgreSQL a un objectif de point de reprise (RPO) défini pour son paramètre `rds.global_db_rpo`, assurez-vous de réinitialiser le paramètre avant de procéder à la mise à niveau. Le processus de mise à niveau de version majeure ne fonctionne pas si le RPO est activé. Ce paramètre est désactivé par défaut. Pour plus d'informations sur les bases de données globales Aurora PostgreSQL et le RPO, consultez Gestion des RPO pour les bases de données globales basées sur Aurora PostgreSQL–.

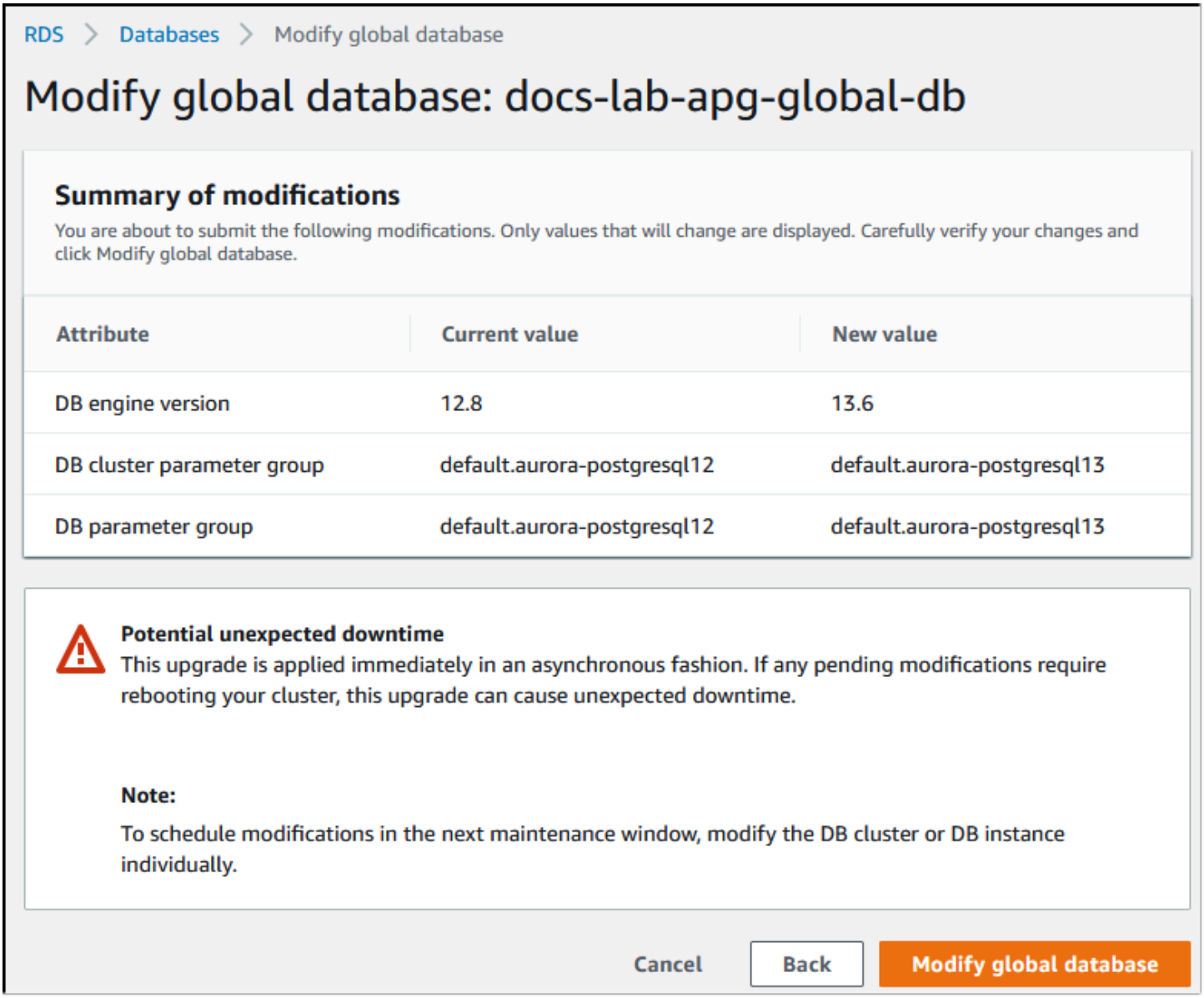
Si vous vérifiez que vos applications peuvent s'exécuter comme prévu lors du déploiement d'essai de la nouvelle version, vous pouvez démarrer le processus de mise à niveau. Pour ce faire, consultez

Mise à niveau du moteur Aurora PostgreSQL vers une nouvelle version majeure. Assurez-vous de choisir l'élément de niveau supérieur dans la liste Bases de données de la console RDS, à savoir Base de données globale, comme illustré dans l'image suivante.



DB identifier	Role	Engine	Region & AZ	Size
docs-lab-apg-aiml	Regional cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-global-db	Global database	Aurora PostgreSQL	2 regions	2 clusters
docs-lab-apg-global-12-7	Primary cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-global-12-7-instance-1	Writer instance	Aurora PostgreSQL	us-west-1c	db.r6g.large
docs-lab-apg-global-12-7-instance-1-us-west-1a	Reader instance	Aurora PostgreSQL	us-west-1a	db.r6g.large
docs-lab-apg-global-db-cluster-northwest	Secondary cluster	Aurora PostgreSQL	us-west-2	2 instances
docs-lab-apg-global-db-instance-north	Reader instance	Aurora PostgreSQL	us-west-2c	db.r6g.large
docs-lab-apg-global-db-instance-north-us-west-2b	Reader instance	Aurora PostgreSQL	us-west-2b	db.r6g.large
docs-lab-apg-main	Regional cluster	Aurora PostgreSQL	us-west-1	2 instances
docs-lab-apg-sless-test-aws-s3	Serverless	Aurora PostgreSQL	us-west-1	0 capacity units

Comme pour toute modification, vous pouvez confirmer que vous souhaitez que le processus se poursuive lorsque vous y êtes invité.



Plutôt que d'utiliser la console, vous pouvez démarrer le processus de mise à niveau en utilisant AWS CLI ou l'API RDS. Comme pour la console, vous utilisez le cluster de bases de données global Aurora plutôt que l'un de ses composants, comme suit :

- Utilisez la commande AWS CLI modify-global-cluster pour lancer la mise à niveau de votre base de données globale Aurora à l'aide d'AWS CLI.
- Utilisez l'API ModifyGlobalCluster pour démarrer la mise à niveau.

Réalisation d’une mise à niveau de version mineure

Vous pouvez utiliser les méthodes suivantes pour mettre à niveau la version mineure d’un cluster de bases de données ou appliquer un correctif à un cluster de bases de données :

Rubriques

Avant d’effectuer une mise à niveau de version mineure

Nous vous recommandons d’effectuer les actions suivantes pour réduire la durée d’indisponibilité lors d’une mise à niveau de version mineure :

- La maintenance du cluster de bases de données Aurora doit être effectuée pendant une période de faible trafic. Utilisez Performance Insights pour identifier ces périodes afin de configurer correctement les fenêtres de maintenance. Pour plus d’informations sur Performance Insights, consultez Surveillance de la charge de la base de données avec Performance Insights sur Amazon RDS. Pour plus d’informations sur la fenêtre de maintenance du cluster de bases de données, consultez Ajustement du créneau de maintenance préféré pour un cluster de bases de données.
- Utilisez des kits AWS SDK qui prennent en charge le backoff exponentiels et l’instabilité en tant que bonne pratique. Pour plus d’informations, consultez Backoff exponentiel et instabilité.

Comment effectuer des mises à niveau de versions mineures et appliquer des correctifs

Les mises à niveau de versions mineures et les correctifs sont disponibles dans les Régions AWS uniquement après avoir réalisé des tests rigoureux. Avant de publier des mises à niveau et des correctifs, Aurora PostgreSQL effectue des tests pour s’assurer que les problèmes de sécurité connus, les bogues et les autres problèmes survenus après la publication de la version communautaire mineure ne perturbent pas la stabilité de la flotte Aurora PostgreSQL.

Si vous activez Activer la mise à niveau automatique des versions mineures, Aurora PostgreSQL met régulièrement à jour votre cluster de bases de données durant la fenêtre de maintenance spécifiée.

Assurez-vous que l'option Activer la mise à niveau automatique des versions mineures est activée pour toutes les instances de cluster de bases de données Aurora PostgreSQL. Pour obtenir des informations sur la façon de définir Mise à niveau automatique des versions mineures et sur la façon dont ce paramètre fonctionne lorsqu'il est appliqué aux niveaux du cluster et de l'instance, consultez Mises à niveau automatiques des versions mineures pour les clusters de bases de données Aurora.

Vérifiez la valeur de l'option Activer la mise à niveau automatique des versions mineures pour tous vos clusters de bases de données Aurora PostgreSQL à l'aide de la commande AWS CLI describe-db-instances comme suit.

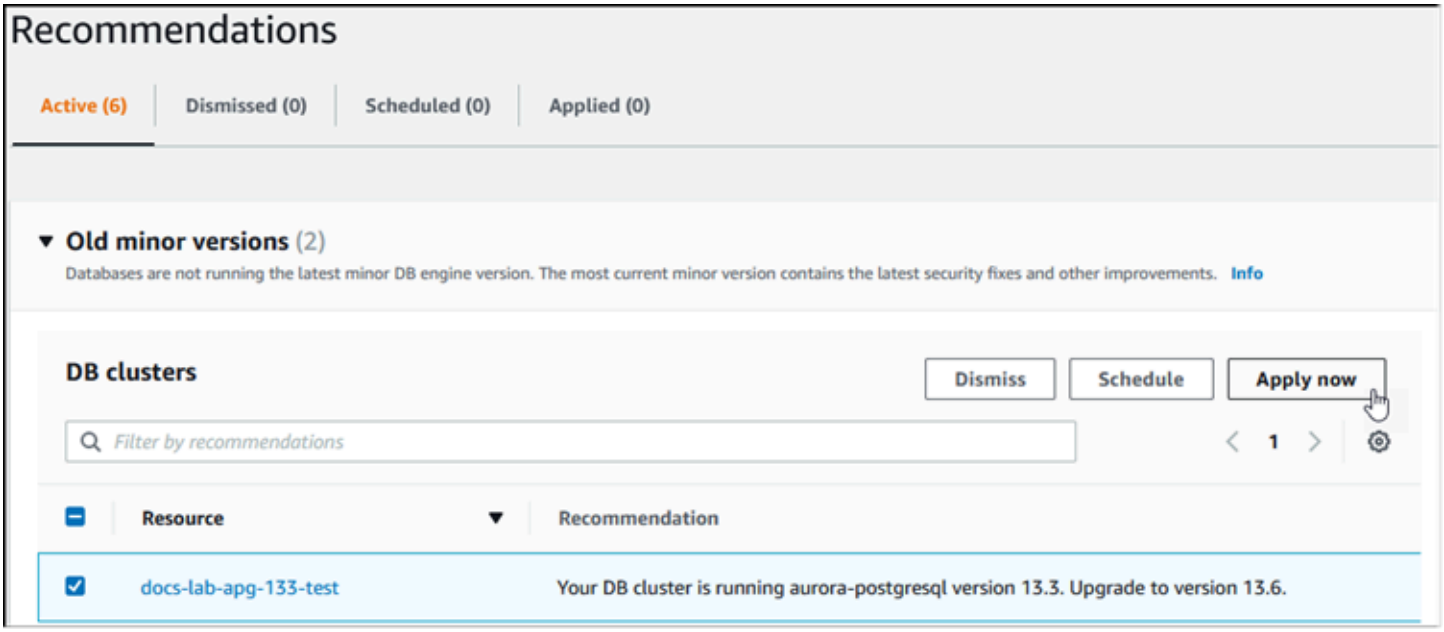
```
aws rds describe-db-instances \
  --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,DBInstanceIdentifier:DBInstanceIdentifier,AutoMinorVer
```

Cette requête renvoie une liste de tous les clusters de base de données Aurora et de leurs instances dont le paramètre `AutoMinorVersionUpgrade` est défini sur `true` ou `false`. La commande suppose qu'AWS CLI est configuré pour cibler votre Région AWS par défaut.

Pour plus d'informations sur l'option AmVU et sur la façon de modifier votre cluster de bases de données Aurora pour l'utiliser, consultez Mises à niveau automatiques des versions mineures pour les clusters de bases de données Aurora.

Vous pouvez mettre à niveau vos clusters de base de données Aurora PostgreSQL vers de nouvelles versions mineures, soit en répondant aux tâches de maintenance, soit en modifiant le cluster pour utiliser la nouvelle version.

Vous pouvez identifier toutes les mises à niveau ou tous les correctifs disponibles pour vos clusters de bases de données Aurora PostgreSQL à l'aide de la console RDS et en ouvrant le menu Recommendations (Recommandations). Vous y trouverez une liste des problèmes de maintenance tels que Old minor versions (Anciennes versions mineures). En fonction de votre environnement de production, vous pouvez choisir de planifier (Schedule) la mise à niveau ou de prendre des mesures immédiates, en choisissant Apply now (Appliquer maintenant), comme indiqué ci-après.



Pour en savoir plus sur la gestion d'un cluster de bases de données Aurora, y compris sur l'application manuelle des correctifs et des mises à niveau de versions mineures, consultez Entretien d'un cluster de bases de données Amazon Aurora.

Mises à niveau de versions mineures et application de correctifs sans durée d'indisponibilité

Il est possible qu'une panne se produise pendant la mise à niveau d'un cluster de bases de données Aurora PostgreSQL. Au cours du processus de mise à niveau, la base de données est arrêtée. Si vous démarrez la mise à niveau alors que la base de données est occupée, vous perdez toutes les connexions et transactions traitées par le cluster de bases de données. Si vous attendez que la base de données soit inactive pour effectuer la mise à niveau, vous devrez peut-être attendre longtemps.

La fonctionnalité ZDP (application de correctifs sans durée d'indisponibilité) améliore le processus de mise à niveau. Avec ZDP, les mises à niveau de versions mineures et les correctifs peuvent être appliqués avec un impact minimal sur votre cluster de bases de données Aurora PostgreSQL. ZDP est utilisé lors de l'application de correctifs ou de mises à jour de versions mineures plus récentes vers des versions d'Aurora PostgreSQL et d'autres versions ultérieures de ces versions mineures, et de nouvelles versions majeures. En d'autres termes, la mise à niveau vers de nouvelles versions mineures à partir de l'une de ces versions utilise ZDP.

Le tableau suivant montre les versions d'Aurora PostgreSQL et les classes d'instance de base de données où ZDP est disponible :

Version	Classes d'instance db.r*	Classes d'instance db.t*	Classes d'instance db.x*	Classe d'instance db.serverless
10.21 et versions ultérieures	Oui	Oui	Oui	N/A
11.16 et versions ultérieures	Oui	Oui	Oui	N/A
11.17 et versions ultérieures	Oui	Oui	Oui	N/A
12.11 et versions ultérieures	Oui	Oui	Oui	N/A
12.12 et versions ultérieures	Oui	Oui	Oui	N/A
13.7 et versions ultérieures	Oui	Oui	Oui	N/A
13.8 et versions ultérieures	Oui	Oui	Oui	Oui
14.3 et versions ultérieures	Oui	Oui	Oui	N/A
14.4 et versions ultérieures	Oui	Oui	Oui	N/A
14.5 et versions ultérieures	Oui	Oui	Oui	Oui
15.3 et versions ultérieures	Oui	Oui	Oui	Oui
16.1 et versions ultérieures	Oui	Oui	Oui	Oui

Au cours du processus de mise à niveau utilisant l'application de correctifs sans durée d'indisponibilité, le moteur de base de données recherche un point silencieux pour suspendre toutes les nouvelles transactions. Cette action protège la base de données lors des applications de correctifs et des mises à niveau. Pour garantir le bon fonctionnement de vos applications quand les transactions sont suspendues, nous vous recommandons d'intégrer une logique de nouvelle tentative dans votre code. Cette approche garantit que le système pourra gérer toute durée d'indisponibilité de courte durée sans défaillance et pourra réessayer les nouvelles transactions après la mise à niveau.

Lorsque l'application de correctifs sans durée d'indisponibilité réussit, les sessions d'application sont conservées, à l'exception de celles avec des connexions interrompues, et le moteur de base de données redémarre avant la fin de la mise à niveau. Le redémarrage du moteur de base de données peut entraîner une chute temporaire du débit, mais celle-ci ne dure généralement que quelques secondes ou une minute environ tout au plus.

Dans certains cas, l'application de correctifs sans durée d'indisponibilité (ZDP) peut échouer. Par exemple, les modifications de paramètres à l'état `pending` sur votre cluster de bases de données Aurora PostgreSQL ou ses instances interfèrent avec l'application de correctifs sans durée d'indisponibilité.

Vous trouverez des métriques et des événements pour les opérations ZDP sur la page Events (Événements) de la console. Les événements se déroulent du début de la mise à niveau ZDP à la fin de la mise à niveau. Dans ce cas, vous pouvez obtenir la durée du processus et le nombre de connexions conservées et supprimées survenues pendant le redémarrage. Pour plus de détails, consultez le journal des erreurs de votre base de données.

Limites des correctifs sans durée d'indisponibilité

Les limitations suivantes s'appliquent à l'application de correctifs sans durée d'indisponibilité :

- ZDP essaie de préserver les connexions actuelles des clients à votre instance d'enregistreur Aurora PostgreSQL tout au long du processus de mise à niveau d'Aurora PostgreSQL. Toutefois, dans les cas suivants, les connexions seront interrompues pour que l'application de correctifs sans durée d'indisponibilité réussisse :
  - Des requêtes ou des transactions de longue durée sont en cours.
  - Des instructions en langage de définition de données (DDL) sont en cours.
  - Des tables temporaires ou des verrous de table sont utilisés
  - Toutes les sessions sont écoutées sur des canaux de notification.
  - Un curseur dont le statut est « WITH HOLD » est en cours d'utilisation.

  - Des connexions TLSv1.1 sont en cours d'utilisation. À partir des versions d'Aurora PostgreSQL ultérieures à 16.1, 15.3, 14.8, 13.11, 12.15 et 11.20, ZDP est pris en charge avec les connexions TLSv1.3.
- ZDP n'est pas pris en charge dans les cas suivants :
  - Lorsque les clusters de bases de données Aurora PostgreSQL sont configurés en tant qu'Aurora Serverless v1.
  - Lors de la mise à niveau des instances de lecture Aurora.
  - Lors de la mise à niveau des instances de lecture Aurora faisant partie d'un cluster de bases de données globale Aurora dans une région secondaire.
  - Lors des correctifs et mises à niveau du système d'exploitation.

Mise à niveau du moteur Aurora PostgreSQL vers une nouvelle version mineure

Vous pouvez mettre à niveau votre cluster de bases de données Aurora PostgreSQL vers une nouvelle version mineure à l'aide de la console, d'AWS CLI ou de l'API RDS. Avant d'effectuer la mise à niveau, nous vous recommandons de suivre les mêmes bonnes pratiques que celles que nous recommandons pour les mises à niveau de versions majeures. Comme pour les nouvelles versions majeures, les nouvelles versions mineures peuvent également comporter des améliorations de l'optimiseur, telles que des corrections, qui peuvent entraîner des régressions du plan de requête. Pour assurer la stabilité du plan, nous vous recommandons d'utiliser l'extension Query Plan Management (QPM) comme indiqué dans Assurer la stabilité du plan après une mise à niveau majeure de la version.

Console

Pour mettre à niveau la version de moteur de votre cluster de bases de données Aurora PostgreSQL

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Bases de données, puis le cluster de bases de données que vous souhaitez mettre à niveau.
3. Sélectionnez Modify. La page Modify DB cluster (Modifier le cluster DB) s'affiche.
4. Dans le champ Version du moteur, sélectionnez la nouvelle version.
5. Choisissez Continuer et vérifiez le récapitulatif des modifications.

6. Pour appliquer les modifications immédiatement, choisissez Appliquer immédiatement. La sélection de cette option peut entraîner une interruption de service dans certains cas. Pour plus d'informations, consultez Modification d'un cluster de bases de données Amazon Aurora.
7. Sur la page de confirmation, examinez vos modifications. Si elles sont correctes, choisissez Modifier le cluster pour enregistrer vos modifications.

   Ou choisissez Retour pour revoir vos modifications, ou choisissez Annuler pour les annuler.

AWS CLI

Pour mettre à niveau la version de moteur d'un cluster de bases de données, utilisez la commande AWS CLI modify-db-cluster avec les paramètres requis suivants :

- `--db-cluster-identifier` : nom de votre cluster de bases de données Aurora PostgreSQL.
- `--engine-version` : numéro de version du moteur de base de données vers lequel effectuer la mise à niveau. Pour obtenir des informations sur les versions de moteur valides, utilisez la commande describe-db-engine-versions de l'AWS CLI.
- `--no-apply-immediately` : appliquer les modifications pendant le créneau de maintenance suivant. Pour appliquer les modifications immédiatement, utilisez plutôt `--apply-immediately`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --engine-version new_version \
    --no-apply-immediately
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --engine-version new_version ^
    --no-apply-immediately
```

API RDS

Pour mettre à niveau la version du moteur d'un cluster de bases de données, utilisez l'opération ModifyDBCluster. Spécifiez les paramètres suivants :

- `DBClusterIdentifier` : nom du cluster de bases de données, par exemple *`mydbcluster`*.
- `EngineVersion` : numéro de version du moteur de base de données vers lequel effectuer la mise à niveau. Pour obtenir des informations sur les versions de moteur valides, utilisez l'opération DescribeDBEngineVersions.
- `ApplyImmediately` : si des modifications doivent être appliquées immédiatement ou au cours du prochain créneau de maintenance. Pour appliquer les modifications immédiatement, définissez la valeur sur `true`. Pour appliquer les modifications pendant le créneau de maintenance suivant, définissez la valeur sur `false`.

Mise à niveau des extensions PostgreSQL

La mise à niveau de votre cluster de base de données Aurora PostgreSQL vers une nouvelle version majeure ou mineure ne met pas à niveau les extensions PostgreSQL en même temps. Pour la plupart des extensions, vous mettez à niveau l'extension une fois la mise à niveau de la version majeure ou mineure terminée. Toutefois, dans certains cas, vous mettez à niveau l'extension avant de mettre à niveau le moteur de base de données Aurora PostgreSQL. Pour obtenir plus d'informations, consultez list of extensions to update dans Test d'une mise à niveau de votre cluster de bases de données de production vers une nouvelle version majeure.

L'installation des extensions PostgreSQL exige des privilèges `rds_superuser`. En règle générale, un utilisateur `rds_superuser` délègue les autorisations sur des extensions spécifiques aux utilisateurs concernés (rôles) afin de faciliter la gestion d'une extension donnée. Cela signifie que la mise à niveau de toutes les extensions de votre cluster de base de données Aurora PostgreSQL peut impliquer plusieurs utilisateurs différents (rôles). Gardez cela à l'esprit en particulier si vous souhaitez automatiser le processus de mise à niveau à l'aide de scripts. Pour plus d'informations sur les privilèges et les rôles PostgreSQL, veuillez consulter Sécurité avec Amazon Aurora PostgreSQL.

> **Note**
>
> Pour plus d'informations sur la mise à jour de l'extension PostGIS, consultez Gestion des données spatiales avec l'extension PostGIS (Étape 6 : Mettre à niveau l'extension PostGIS). Pour mettre à jour l'extension `pg_repack`, supprimez l'extension, puis créez la nouvelle version dans l'instance de base de données mise à niveau. Pour plus d'informations, veuillez consulter pg_repack installation dans la documentation `pg_repack`.

Pour mettre à jour une extension après une mise à niveau du moteur, utilisez la commande `ALTER EXTENSION UPDATE`.

```
ALTER EXTENSION extension_name UPDATE TO 'new_version';
```

Pour afficher une liste des extensions actuellement installées, utilisez le catalogue pg_extension PostgreSQL dans la commande suivante.

```
SELECT * FROM pg_extension;
```

Pour afficher une liste des versions d'extensions spécifiques disponibles pour votre installation, utilisez la vue pg_available_extension_versions PostgreSQL dans la commande suivante.

```
SELECT * FROM pg_available_extension_versions;
```

Technique alternative de mise à niveau bleu/vert

Dans certains cas, votre priorité absolue est d'effectuer une bascule immédiate de l'ancien cluster vers un cluster mis à niveau. Dans de telles situations, vous pouvez utiliser un processus en plusieurs étapes qui exécute les anciens et les nouveaux clusters side-by-side. Dans ce cas, répliquez les données de l'ancien cluster au nouveau jusqu'à ce que ce dernier soit prêt à prendre le relais. Pour en savoir plus, consultez Utilisation des Blue/Green déploiements pour les mises à jour de bases de.

Utilisation d'une version LTS (Long-Term Support) d'Aurora PostgreSQL

Chaque nouvelle version d'Aurora PostgreSQL reste disponible pendant un certain temps afin que vous puissiez l'utiliser pour créer ou mettre à niveau un cluster de bases de données. Une fois cette période écoulée, vous devez mettre à niveau tout cluster utilisant cette version. Vous pouvez mettre à niveau votre cluster manuellement avant la fin de la période de prise en charge. Aurora peut également effectuer sa mise à niveau automatiquement une fois que sa version d'Aurora PostgreSQL n'est plus prise en charge.

Aurora désigne certaines versions d'Aurora PostgreSQL comme étant des versions « long-term support (LTS) ». Les clusters de bases de données qui utilisent des versions LTS peuvent conserver la même version plus longtemps et faire l'objet de cycles de mise à niveau moins nombreux que les clusters qui utilisent des versions non-LTS. Les versions mineures de LTS incluent uniquement des corrections de bogues (via des versions de correctif) pour des problèmes critiques de stabilité et de sécurité ; une version LTS n'inclut pas les nouvelles fonctionnalités publiées après son introduction.

Une fois par an, les clusters de bases de données s'exécutant sur une version mineure LTS sont mis à jour avec la dernière version de correctif de la version LTS. Nous effectuons cette mise à jour pour nous assurer que vous bénéficiez des correctifs cumulatifs de sécurité et de stabilité. Il peut arriver que nous corrigions une version mineure LTS plus fréquemment si des correctifs critiques, par exemple de sécurité, doivent être appliqués.

> **Note**
>
> Si vous souhaitez rester sur une version mineure LTS pendant toute la durée de son cycle de vie, veillez à désactiver la mise à niveau automatique des versions mineures pour vos instances de base de données. Pour éviter la mise à niveau automatique de votre cluster de bases de données à partir de la version mineure LTS, décochez la case Activer la mise à niveau automatique des versions mineures pour toutes les instances de base de données de votre cluster Aurora.

Nous vous recommandons d'effectuer la mise à niveau vers la dernière version, au lieu d'utiliser la version LTS, pour la plupart de vos clusters Aurora PostgreSQL. Cela vous permet de bénéficier des avantages d'Aurora en tant que service géré et vous donne accès aux dernières fonctionnalités et aux derniers correctifs de bogues. Les versions LTS sont destinées aux clusters présentant les caractéristiques suivantes :

- Vous ne pouvez pas vous permettre d'avoir une durée d'indisponibilité sur votre application Aurora PostgreSQL pour les mises à niveau, en dehors des rares occurrences de correctifs critiques.
- Le cycle de test du cluster et des applications associées dure très longtemps pour chaque mise à niveau du moteur de base de données Aurora PostgreSQL.
- La version de la base de données de votre cluster Aurora PostgreSQL dispose de toutes les fonctionnalités de moteur de base de données et de tous les correctifs de bogues dont votre application a besoin.

Les versions LTS actuelles pour Aurora PostgreSQL sont les suivantes :

- PostgreSQL 16.8. Il a été publié le 7 avril 2025. Pour plus d'informations, consultez PostgreSQL 16.8 dans les notes de mise à jour d'Aurora PostgreSQL.
- PostgreSQL 15.10. Cette version est sortie le 27 décembre 2024. Pour plus d'informations, consultez PostgreSQL 15.10 dans Notes de mise à jour d'Aurora PostgreSQL.

- PostgreSQL 14.6. Cette version a été publiée le 20 janvier 2023. Pour plus d'informations, consultez PostgreSQL 14.6 dans Notes de mise à jour d'Aurora PostgreSQL.
- PostgreSQL 13.9. Cette version a été publiée le 20 janvier 2023. Pour plus d'informations, consultez PostgreSQL 13.9 dans Notes de mise à jour d'Aurora PostgreSQL.
- PostgreSQL 12.9. Date de sortie : 25 février 2022. Pour plus d'informations, consultez PostgreSQL 12.9 dans Notes de mise à jour d'Aurora PostgreSQL.
- PostgreSQL 11.9 (Aurora PostgreSQL version 3.4). Elle est sortie le 11 décembre 2020. Pour plus d'informations sur cette version, consultez PostgreSQL 11.9, Aurora PostgreSQL version 3.4 dans Notes de mise à jour d'Aurora PostgreSQL.

Pour plus d'informations sur les délais de support et les cycles de distribution des versions LTS, consultez Calendriers de publication pour Aurora PostgreSQL.

Pour plus d'informations sur l'identification des versions d'Aurora et du moteur de base de données, consultez Identification des versions Amazon Aurora PostgreSQL.

Utilisation d’Amazon Aurora PostgreSQL Limitless Database

La base de données Amazon Aurora PostgreSQL Limitless assure une mise à l’échelle horizontale automatisée pour traiter des millions de transactions d’écriture par seconde et gère des pétaoctets de données tout en préservant la simplicité offerte par l’utilisation d’une seule et même base de données. Avec la base de données Aurora PostgreSQL Limitless, vous pouvez vous concentrer sur la création d’applications à grande échelle sans avoir à créer ni à gérer de solutions complexes pour mettre à l’échelle vos données entre plusieurs instances de base de données afin de répondre aux besoins de vos charges de travail.

Rubriques

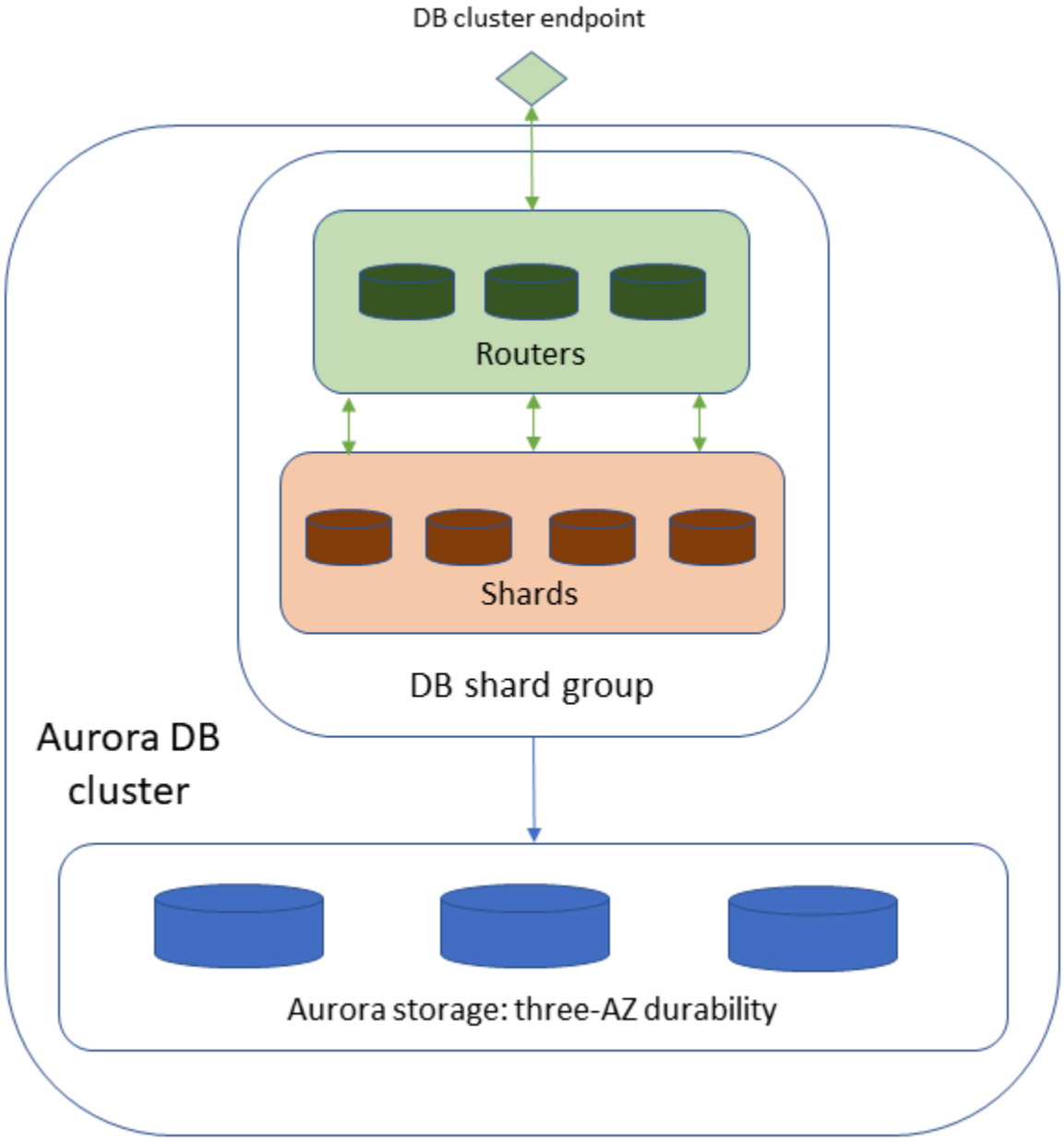
Architecture d’Aurora PostgreSQL Limitless Database

Limitless Database assure sa montée en charge grâce à une architecture à deux couches constituée de multiples nœuds de base de données. Les nœuds sont soit des routeurs, soit des partitions.

- Les partitions sont des instances de base de données Aurora PostgreSQL qui stockent chacune un sous-ensemble des données de votre base de données, permettant ainsi un traitement parallèle pour obtenir un débit d’écriture plus élevé.
- Les routeurs gèrent la nature distribuée de la base de données et présentent une image de base de données unique aux clients de base de données. Les routeurs gèrent les métadonnées relatives à l’emplacement de stockage des données, analysent les commandes SQL entrantes et envoient ces commandes aux partitions. Ils agrègent ensuite les données provenant des différentes partitions pour renvoyer un résultat unique au client, tout en gérant les transactions distribuées afin d’assurer la cohérence de l’ensemble de la base de données distribuée.

Aurora PostgreSQL Limitless Database se distingue de la Clusters DB Aurora standard en utilisant un groupe de partitions de base de données plutôt qu’une instance de base de données d’écriture et plusieurs instances de base de données de lecture. Tous les nœuds qui constituent votre architecture Limitless Database sont contenus dans le groupe de partitions de base de données. Les partitions et routeurs individuels du groupe de partitions de base de données ne sont pas visibles dans votre Compte AWS. Vous utilisez le point de terminaison du cluster de bases de données pour accéder à Limitless Database.

La figure suivante illustre l’architecture générale d’Aurora PostgreSQL Limitless Database.



Pour plus d'informations sur l'architecture d'Aurora PostgreSQL Limitless Database et sur la façon dont vous pouvez l'utiliser, consultez Réaliser la mise à l'échelle avec Amazon Aurora PostgreSQL Limitless Database sur la chaîne Événements AWS sur YouTube.

Pour plus d'informations sur l'architecture d'un cluster de bases de données Aurora Standard, consultez Clusters de bases de données Amazon Aurora.

Termes clés relatifs à Aurora PostgreSQL Limitless Database

Groupe de partitions de base de données

Un conteneur pour les nœuds de Limitless Database (partitions et routeurs).

Routeur

Nœud qui accepte les connexions SQL des clients, envoie des commandes SQL aux partitions, assure la cohérence à l'échelle du système et renvoie les résultats aux clients.

Partition

Nœud qui stocke un sous-ensemble de tables partitionnées, des copies complètes des tables de référence et des tables standard. Accepte les requêtes des routeurs, sans autoriser de connexions directes depuis les clients.

Table partitionnée

Une table dont les données sont réparties entre plusieurs partitions.

Clé de partition

Colonne ou ensemble de colonnes d'une table partitionnée utilisée pour déterminer le partitionnement entre les partitions.

Tables colocalisées

Deux tables partitionnées qui partagent la même clé de partition et sont explicitement déclarées comme colocalisées. Toutes les données correspondant à la même valeur de clé de partition sont envoyées à la même partition.

Table de référence

Une table dont les données sont copiées intégralement sur chaque partition.

Table standard

Type de table par défaut dans une base de données Limitless. Vous pouvez convertir des tables standard en tables partitionnées et en tables de référence.

Toutes les tables standard sont stockées sur la même partition sélectionnée par le système, ce qui permet d'effectuer des jointures entre tables standard au sein d'une seule partition. Cependant, les tables standard sont limitées par la capacité maximale de la partition (128 Tio). Cette partition stocke également les données des tables partitionnées et de référence, de sorte que la limite effective des tables standard est inférieure à 128 Tio.

Types de tables pour Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database prend en charge trois types de tables partitionnées référence et standard.

Les données des tables partitionnées sont réparties sur toutes les partitions du groupe de partitions de base de données. Limitless Database effectue cette opération automatiquement à l'aide d'une clé de partition, c'est-à-dire une colonne ou un ensemble de colonnes que vous définissez lors du partitionnement de la table. Toutes les données correspondant à la même valeur de clé de partition sont envoyées à la même partition. Le partitionnement est basé sur le hachage, et non sur des plages de valeurs ou des listes.

Voici quelques exemples de cas d'utilisation adaptés aux tables partitionnées :

- L'application fonctionne avec un sous-ensemble de données distinct.
- La table est très volumineuse.
- La table est susceptible de croître plus rapidement que les autres tables.

Les tables partitionnées peuvent être colocalisées, c'est-à-dire qu'elles partagent une clé de partition commune ; ainsi, les données présentant la même valeur de clé de partition dans les deux tables sont envoyées sur la même partition. Si vous colocalisez des tables et les joignez à l'aide de la clé de partition, la jointure peut s'effectuer sur une seule partition, puisque toutes les données nécessaires s'y trouvent.

Les tables de référence contiennent une copie complète de toutes leurs données sur chaque partition du groupe de partitions de base de données. Les tables de référence sont généralement utilisées pour les tables plus petites, dont le volume d'écriture est limité, mais qui nécessitent des jointures fréquentes et ne conviennent pas au partitionnement. Parmi les exemples de tables de référence figurent les tables de dates ainsi que les tables de données géographiques, telles que celles des États, des villes et des codes postaux.

Les tables standard constituent le type de table par défaut dans Aurora PostgreSQL Limitless Database. Ce ne sont pas des tables distribuées. Aurora PostgreSQL Limitless Database prend en charge les jointures entre les tables standard, ainsi qu'entre les tables standard, partitionnées et de référence.

Facturation relative à Aurora PostgreSQL Limitless Database

Pour en savoir plus sur la facturation d'Aurora PostgreSQL Limitless Database, consultez Facturation d'une instance de base de données pour Aurora.

Pour obtenir des informations sur la tarification Aurora, consultez la page de tarification Aurora.

Mise en route avec Aurora PostgreSQL Limitless Database

Vous devez effectuer les actions suivantes pour démarrer avec Aurora PostgreSQL Limitless Database :

1. Créez un cluster de bases de données Aurora PostgreSQL et un groupe de partitions de base de données pour Limitless Database. Pour plus d'informations, consultez Création d'un cluster de bases de données utilisant Aurora PostgreSQL Limitless Database.
2. Créez des tables partitionnées et des tables de référence dans le groupe de partitions de base de données. Pour plus d'informations, consultez Création de tables Aurora PostgreSQL Limitless Database.
3. Modifiez la capacité de votre groupe de partitions de base de données, divisez les partitions et ajoutez des routeurs. Pour plus d'informations, consultez Utilisation de groupes de partitions de base de données.
4. Chargez les données dans le groupe de partitions de base de données. Pour plus d'informations, consultez Chargement de données dans Aurora PostgreSQL Limitless Database.
5. Exécutez des requêtes et d'autres instructions SQL sur le groupe de partitions de base de données. Pour plus d'informations, consultez Interrogation d'Aurora PostgreSQL Limitless Database.
6. Surveillez les performances de Limitless Database. Pour plus d'informations, consultez Surveillance d'Aurora PostgreSQL Limitless Database.

Exigences et considérations relatives à Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database présente les exigences et considérations suivantes.

Rubriques

Exigences relatives à Aurora PostgreSQL Limitless Database

Assurez-vous de respecter les exigences suivantes pour Aurora PostgreSQL Limitless Database.

- La base de données Aurora PostgreSQL Limitless est disponible Régions AWS partout sauf en Asie-Pacifique (Taipei).

  > **Note**
  >
  > Si vous créez votre cluster de bases de données Aurora PostgreSQL Limitless Database dans la région USA Est (Virginie du Nord), n'incluez pas la zone de disponibilité (AZ) `us-east-1e` dans votre groupe de sous-réseaux de base de données. En raison des limites de ressources, Aurora Serverless v2, et par conséquent, Aurora PostgreSQL Limitless Database, n'est pas prise en charge dans l'AZ `us-east-1e`.

- Aurora PostgreSQL Limitless Database prend uniquement en charge la configuration de stockage en cluster de bases de données Aurora I/O-Optimized. Pour plus d'informations, consultez Configurations de stockage pour les clusters de bases de données Amazon Aurora.
- Aurora PostgreSQL Limitless Database utilise des versions spécifiques du moteur de base de données Aurora PostgreSQL dédiées à Aurora PostgreSQL Limitless Database :
  - `16.6.8-limitless`
  - `16.6-limitless`
  - `16.4-limitless`
- Votre cluster de bases de données ne peut pas contenir d'instance de base de données en écriture ni en lecture.

- Vous devez utiliser Enhanced Monitoring et Performance Insights. La période de conservation de Performance Insights doit être fixée à au moins 1 mois (31 jours).
- Vous devez exporter le journal PostgreSQL vers Amazon Logs. CloudWatch

> **Note**
>
> Certaines fonctionnalités requises, telles que Enhanced Monitoring, Performance Insights et CloudWatch Logs, entraînent des frais supplémentaires. Pour obtenir des informations sur la tarification Aurora, consultez la page de tarification Aurora.

Considérations relatives à Aurora PostgreSQL Limitless Database

Les considérations suivantes s'appliquent aux groupes de partitions de base de données d'Aurora PostgreSQL Limitless Database :

- Un cluster de bases de données ne peut contenir qu'un seul groupe de partitions de base de données.
- Vous pouvez avoir jusqu'à cinq groupes de partitions de base de données par groupe. Région AWS

  Par conséquent, vous pouvez disposer d'un maximum de cinq clusters de base de données Aurora PostgreSQL Limitless par cluster. Région AWS Pour de plus amples informations, veuillez consulter Quotas dans Amazon Aurora.
- Vous pouvez définir la capacité maximale d'un groupe de partitions de base de données entre 16 ACUs et 6 144. Pour les limites de capacité supérieures à 6144 ACUs, contactez AWS.

  Le nombre initial de routeurs et de partitions est déterminé par la capacité maximale que vous définissez lorsque vous créez un groupe de partitions de base de données. Pour plus d'informations, consultez Corrélation de la capacité maximale du groupe de partitions de base de données avec le nombre de routeurs et de partitions créés.
- Le nombre de routeurs et de partitions ne change pas lorsque vous modifiez la capacité maximale d'un groupe de partitions de base de données.
- Assurez-vous que le sous-réseau de base de données dans lequel vous créez le groupe de partitions de base de données dispose d'un nombre suffisant d'adresses IP libres pour permettre la connexion au groupe de partitions de base de données. Chaque routeur nécessite une adresse IP,

et chaque partition du groupe de partitions de base de données peut utiliser jusqu'à trois adresses IP.

Pour plus d'informations sur le nombre de routeurs créés lorsque vous créez un groupe de partitions de base de données, consultez Corrélation de la capacité maximale du groupe de partitions de base de données avec le nombre de routeurs et de partitions créés.

- Si vous rendez votre groupe de partitions de base de données accessible au public, assurez-vous de configurer une passerelle Internet dans votre VPC.
- Les fonctions SQL sont utilisées pour diviser les partitions et ajouter des routeurs.
- La fusion de partitions n'est pas prise en charge.
- Vous ne pouvez pas supprimer des partitions ni des routeurs individuels.
- Vous ne pouvez en aucun cas modifier les clés de partition (ou effectuer des opérations `UPDATE`), y compris changer leurs valeurs dans les lignes de table.

  Pour modifier une clé de partition, vous devez la supprimer, puis la recréer.
- Les niveaux d'isolement lecture répétable, lecture validée et lecture non validée sont pris en charge. Vous ne pouvez pas définir le niveau d'isolement sur sérialisable.
- Certaines commandes SQL ne sont pas prises en charge. Pour plus d'informations, consultez Référence Aurora PostgreSQL Limitless Database.
- Toutes les extensions PostgreSQL ne sont pas prises en charge. Pour de plus amples informations, veuillez consulter Extensions.
- Lors de la création d'un groupe de partitions ou lors de l'ajout de nouveaux nœuds de groupe de partitions (partitions ou routeurs), ces nœuds sont créés dans l'une des zones de disponibilité (AZs) disponibles pour le cluster de bases de données. Vous ne pouvez pas affecter une zone de disponibilité précise à des nœuds individuels.
- Si vous utilisez une redondance de calcul de 2 (deux serveurs de secours pour le groupe de partitions de base de données), assurez-vous que votre groupe de sous-réseaux de base de données en compte au moins trois. AZs
- Aurora PostgreSQL Limitless Database prend en charge jusqu'à 54 caractères pour les noms de table partitionnées.

Les considérations suivantes s'appliquent au cluster de bases de données d'Aurora PostgreSQL Limitless Database :

- Nous vous recommandons d'utiliser des politiques AWS gérées pour limiter les autorisations relatives à votre base de données et à vos applications à celles dont les clients ont besoin pour leurs cas d'utilisation. Pour de plus amples informations, veuillez consulter Bonnes pratiques en matière de politiques.
- Lorsque vous créez votre cluster de bases de données Aurora PostgreSQL Limitless Database, vous définissez uniquement les paramètres de mise à l’échelle pour le groupe de partitions de base de données.
- Si vous devez supprimer votre cluster de bases de données, vous devez d’abord supprimer le groupe de partitions de base de données.
- Aurora PostgreSQL Limitless Database ne peut pas être une source de réplication.

Fonctionnalités non prises en charge dans Aurora PostgreSQL Limitless Database

Les fonctionnalités Aurora PostgreSQL suivantes ne sont pas prises en charge dans Aurora PostgreSQL Limitless Database :

- Authentification Active Directory (Kerberos)
- Amazon DevOps Guru
- Amazon ElastiCache
- Déploiements Amazon RDS Blue/Green
- Proxy Amazon RDS
- Aurora Auto Scaling (ajout automatique d’instances de lecteur au cluster de bases de données)
- Aurora Global Database
- Machine Learning Aurora
- Recommandations concernant Aurora
- Aurora Serverless v1
- Intégrations zéro ETL Aurora
- AWS Backup
- AWS Lambda intégration
- AWS Secrets Manager
- Babelfish for Aurora PostgreSQL
- Clonage de clusters de bases de données
- Points de terminaison personnalisés
- Flux d’activité de base de données.
- Réplicas en lecture
- API de données RDS

Prérequis pour l’utilisation d’Aurora PostgreSQL Limitless Database

Pour utiliser Aurora PostgreSQL Limitless Database, vous devez d’abord effectuer les tâches suivantes.

Rubriques

Activation des opérations de groupes de partitions de base de données

Avant de créer un groupe de partitions de base de données, vous devez activer les opérations de groupe de partitions de base de données.

- Ajoutez la section suivante à la politique IAM du rôle IAM de l’utilisateur qui accède à Aurora PostgreSQL Limitless Database :

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDBShardGroup",
            "Effect": "Allow",
            "Action": [
                "rds:CreateDBShardGroup",
                "rds:DescribeDBShardGroups",
                "rds:DeleteDBShardGroup",
                "rds:ModifyDBShardGroup",
                "rds:RebootDBShardGroup"
            ],
            "Resource": [
                "arn:aws:rds:*:*:shard-group:*",
                "arn:aws:rds:*:*:cluster:*"
            ]
        }
    ]
}
```

Création d’un cluster de bases de données utilisant Aurora PostgreSQL Limitless Database

Vous créez un cluster de bases de données Aurora à l’aide de la version Limitless Database d’Aurora PostgreSQL, puis vous ajoutez un groupe de partitions de base de données au cluster. Lorsque vous ajoutez un groupe de partitions de base de données, vous indiquez la capacité de calcul maximale pour l’ensemble du groupe de partitions de base de données (somme des capacités de l’ensemble des routeurs et des partitions) en unités de capacité Aurora (ACU). Chaque ACU combine environ 2 gibioctets (Gio) de mémoire, avec une UC et une mise en réseau correspondantes. La mise à l’échelle augmente ou diminue la capacité de votre groupe de partitions de base de données, en fonction de la charge de travail de votre application, comme c’est le cas avec Aurora Serverless v2.

Rubriques

Corrélation de la capacité maximale du groupe de partitions de base de données avec le nombre de routeurs et de partitions créés

Le nombre initial de routeurs et de partitions dans un groupe de partitions de base de données est déterminé par la capacité maximale que vous définissez lorsque vous créez le groupe de partitions de base de données. Plus la capacité maximale est élevée, plus le nombre de routeurs et de partitions créés dans le groupe de partitions de base de données est élevé.

Chaque nœud (partition ou routeur) possède sa propre valeur de capacité actuelle, également mesurée en ACU.

- Limitless Database met à l’échelle la capacité d’un nœud lorsque sa capacité actuelle est insuffisante pour supporter la charge. Cependant, les nœuds n’augmentent pas verticalement lorsque la capacité totale atteint son maximum.
- Limitless Database réduit la capacité d’un nœud lorsque sa capacité actuelle dépasse celle requise pour la charge de travail. Cependant, les nœuds ne sont pas réduits verticalement lorsque la capacité totale atteint son minimum.

Le tableau suivant montre la corrélation entre la capacité maximale du groupe de partitions de base de données en unités de capacité Aurora (ACU) et le nombre de nœuds (routeurs et partitions) créés.

> **Note**
>
> Ces valeurs sont susceptibles d'être modifiées.
> Si vous définissez la redondance de calcul sur une valeur autre que zéro, le nombre total de partitions sera doublé ou triplé. Des frais supplémentaires vous seront alors facturés.
> La capacité des nœuds en veille de calcul est augmentée ou réduite verticalement pour être identique à celle du nœud d'écriture. Il n'est pas nécessaire de définir une plage de capacité distincte pour les instances de secours.

Nombre total de nœuds	Routeur	Partitions	Capacité minimale par défaut (ACU)	Plage de capacité maximale (ACU)
4	2	2	16	16 – 400
5	2	3	20	401 – 500
6	2	4	24	501 – 600
7	3	4	28	601 – 700
8	3	5	32	701 – 800
9	3	6	36	801 – 900
10	4	6	40	901 – 1 000
11	4	7	44	1 001 – 1 100
12	4	8	48	1 101 – 1 200
13	5	8	52	1 201 – 1 300
14	5	9	56	1 301 – 1 400
15	5	10	60	1 401 – 1 500

Nombre total de nœuds	Routeur	Partitions	Capacité minimale par défaut (ACU)	Plage de capacité maximale (ACU)
16	6	10	64	1 501 – 1 600
17	6	11	68	1 601 – 1 700
18	6	12	72	1 701 – 1 800
19	7	12	76	1 801 – 1 900
20	7	13	80	1 901 – 2 000
21	7	14	84	2 001 – 2 100
22	8	14	88	2 101 – 2 200
23	8	15	92	2 201 – 2 300
24	8	16	96	2 301 – 6 144

La configuration dynamique basée sur la capacité maximale du groupe de partitions de base de données n'est disponible qu'au moment de la création. Le nombre de routeurs et de partitions reste le même lorsque la capacité maximale est modifiée. Pour plus d'informations, consultez Modification de la capacité d'un groupe de partitions de base de données.

Vous pouvez utiliser les commandes SQL pour ajouter des partitions et des routeurs à un groupe de partitions de base de données. Pour plus d'informations, consultez les ressources suivantes :

- Fractionnement d'une partition dans un groupe de partitions de base de données
- Ajout d'un routeur à un groupe de partitions de base de données

> **Note**
>
> Les partitions ou les routeurs ne peuvent pas être supprimés.

Création d’un cluster de bases de données

Utilisez les procédures suivantes pour créer un cluster de bases de données Aurora PostgreSQL utilisant Aurora PostgreSQL Limitless Database.

Vous pouvez utiliser la AWS Management Console ou l’AWS CLI pour créer votre cluster de bases de données utilisant Aurora PostgreSQL Limitless Database. Vous créez le cluster de bases de données principal et le groupe de partitions de base de données.

Console

Lorsque vous utilisez la AWS Management Console pour créer le cluster de bases de données principal, le groupe de partitions de base de données est également créé dans le cadre de la même procédure.

Pour créer un cluster de bases de données à partir de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Choisissez Create database (Créer une base de données).

   La page Create database (Créer une base de données) s’affiche.
3. Pour Type de moteur, choisissez Aurora (compatible avec PostgreSQL).
4. Pour Version, choisissez l’une des options suivantes :

   - Aurora PostgreSQL avec Limitless Database (compatible avec PostgreSQL 16.4)
   - Aurora PostgreSQL avec Limitless Database (compatible avec PostgreSQL 16.6)
5. Pour Aurora PostgreSQL Limitless Database :

a. Entrez un identifiant de groupe de partitions de base de données.

> **Important**
>
> Après avoir créé le groupe de partitions de base de données, vous ne pouvez pas modifier l'identifiant du cluster de bases de données ou l'identifiant du groupe de partitions de base de données.

b. Pour la plage de capacité du groupe de partitions de base de données :

   i. Entrez la Capacité minimale (ACU). Utilisez une valeur d'au moins 16 ACU.

      La valeur par défaut est fixée à 16 ACU dans un environnement de développement. La valeur par défaut est fixée à 24 ACU dans un environnement de production.

   ii. Entrez la Capacité maximale (ACU). Indiquez une valeur comprise entre 16 ACU et 6 144 ACU.

      La valeur par défaut est fixée à 64 ACU dans un environnement de développement. La valeur par défaut est fixée à 384 ACU dans un environnement de production.

Pour plus d'informations, consultez Corrélation de la capacité maximale du groupe de partitions de base de données avec le nombre de routeurs et de partitions créés.

c. Pour le déploiement d'un groupe de partitions de base de données, choisissez si vous souhaitez créer des instances de secours pour le groupe de partitions de base de données :

- Aucune redondance de calcul : crée un groupe de partitions de base de données sans instance de secours pour chaque partition. C'est la valeur par défaut.
- Redondance des calculs avec une seule cible de basculement : crée un groupe de partitions de base de données avec une instance de calcul de secours dans une zone de disponibilité (AZ) différente.
- Redondance de calcul avec deux cibles de basculement : crée un groupe de partitions de base de données avec deux instances de calcul de secours situées dans deux zones de disponibilité différentes.

> **Note**
>
> Si vous définissez la redondance de calcul sur une valeur autre que zéro, le nombre total d'instances de partitions de base de données sera doublé ou triplé. Ces instances de base de données supplémentaires sont des instances de calcul de secours, dont la capacité est augmentée ou réduite verticalement pour être identique à celle du nœud d'écriture. Il n'est pas nécessaire de définir une plage de capacité distincte pour les instances de secours. Par conséquent, l'utilisation des ACU ainsi que la facturation doublent ou triplent en conséquence. Pour connaître l'utilisation exacte des ACU résultant de la redondance de calcul, reportez-vous à la métrique `DBShardGroupComputeRedundancyCapacity` figurant dans Métriques DBShardGroup.

d. Choisissez si le groupe de partitions de base de données doit être accessible publiquement.

> **Note**
>
> Ce paramètre ne peut plus être modifié une fois le groupe de partitions de base de données créé.

6. Pour la Connectivité :

a. (Facultatif) Sélectionnez Se connecter à une ressource de calcul EC2, puis choisissez une instance EC2 existante ou créez-en une nouvelle.

> **Note**
>
> Si vous vous connectez à une instance EC2, le groupe de partitions de base de données ne peut pas être rendu public.

b. Pour le Type de réseau, choisissez IPv4 ou le mode Double pile.

c. Choisissez le Cloud privé virtuel (VPC) et le Groupe de sous-réseaux de base de données, ou utilisez les paramètres par défaut.

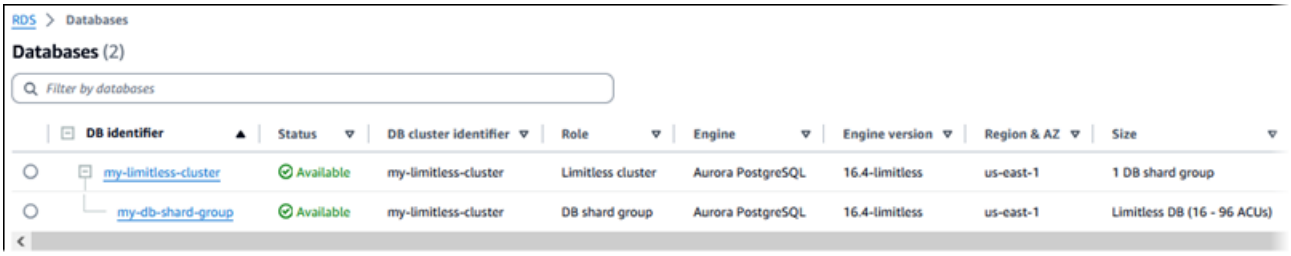
> **Note**
>
> Si vous créez votre cluster de bases de données Limitless Database dans la région USA Est (Virginie du Nord), n'incluez pas la zone de disponibilité (AZ) `us-east-1e` dans votre groupe de sous-réseaux de base de données. En raison des limites de ressources, Aurora Serverless v2, et par conséquent, Limitless Database, n'est pas prise en charge dans l'AZ `us-east-1e`.

d. Choisissez le Groupe de sécurité VPC (pare-feu) ou utilisez le paramètre par défaut.

7. Pour Authentification de base de données, choisissez Authentification par mot de passe ou Authentification par mot de passe et IAM.
8. Pour Surveillance, assurez-vous que les cases Activer Performance Insights et Activer Enhanced Monitoring sont cochées.

   La durée de conservation doit être fixée à au moins 1 mois pour Performance Insights.
9. Développez la dernière Configuration supplémentaire de la page.
10. Pour les Exportations de journaux, assurez-vous que la case Journal PostgreSQL est cochée.
11. Renseignez les autres paramètres si nécessaire. Pour plus d'informations, consultez Paramètres pour les clusters de bases de données Aurora.
12. Choisissez Create database (Créer une base de données).

Une fois le cluster de bases de données principal et le groupe de partitions de base de données créés, ils sont affichés sur la page Bases de données.


RDS > Databases
Databases (2)
Filter by databases
DB identifier
Status
DB cluster identifier
Role
Engine
Engine version
Region & AZ
Size
my-limitless-cluster
Available
my-limitless-cluster
Limitless cluster
Aurora PostgreSQL
16.4-limitless
us-east-1
1 DB shard group
my-db-shard-group
Available
my-limitless-cluster
DB shard group
Aurora PostgreSQL
16.4-limitless
us-east-1
Limitless DB (16 - 96 ACUs)

Interface de ligne de commande (CLI)

Lorsque vous utilisez AWS CLI pour créer un cluster de bases de données utilisant Aurora PostgreSQL Limitless Database, vous devez effectuer les tâches suivantes :

1. Créez un cluster de bases de données principal.
2. Création du groupe de partitions de base de données.

Créez un cluster de bases de données principal

Les paramètres suivants sont nécessaires pour créer le cluster de bases de données :

- `--db-cluster-identifier` : nom du cluster de bases de données.
- `--engine` : le cluster de bases de données doit utiliser le moteur de base de données `aurora-postgresql`.
- `--engine-version` : le cluster de bases de données doit utiliser l'une des versions du moteur de base de données.
  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type` : le cluster de bases de données doit utiliser la configuration de stockage du cluster de bases de données `aurora-iopt1`.
- `--cluster-scalability-type` : spécifie le mode de capacité de mise à l'échelle du cluster de bases de données Aurora. Lorsqu'il est défini sur `limitless`, le cluster fonctionne comme une base de données Aurora PostgreSQL Limitless Database. Lorsqu'il est défini sur `standard` (valeur par défaut), le cluster utilise la création d'instance de base de données normale.

  > **Note**
  >
  > Ce paramètre ne peut plus être modifié une fois le cluster de bases de données créé.

- `--master-username` : le nom de l'utilisateur principal du cluster de bases de données.
- `--master-user-password` : le mot de passe de l'utilisateur principal.
- `--enable-performance-insights` : vous devez activer Performance Insights.
- `--performance-insights-retention-period` : la période de conservation de Performance Insights doit être fixée à au moins 31 jours.

- `--monitoring-interval` : l'intervalle, en secondes, entre les points lorsque des métriques Enhanced Monitoring sont collectées pour le cluster de bases de données. Cette valeur ne peut pas être `0`.
- `--monitoring-role-arn` : l'Amazon Resource Name (ARN) du rôle IAM qui autorise RDS à envoyer des métriques Enhanced Monitoring à Amazon CloudWatch Logs.
- `--enable-cloudwatch-logs-exports` : vous devez exporter les journaux `postgresql` vers CloudWatch Logs.

Les paramètres suivants sont facultatifs :

- `--db-subnet-group-name` : le nom du groupe de sous-réseaux de base de données à associer à ce cluster de bases de données. Ce paramètre détermine également le VPC associé au cluster de bases de données.

  > **Note**
  >
  > Si vous créez votre cluster de bases de données Limitless Database dans la région USA Est (Virginie du Nord), n'incluez pas la zone de disponibilité (AZ) `us-east-1e` dans votre groupe de sous-réseaux de base de données. En raison des limites de ressources, Aurora Serverless v2, et par conséquent, Limitless Database, n'est pas prise en charge dans l'AZ `us-east-1e`.

- `--vpc-security-group-ids` : une liste des groupes de sécurité VPC à associer à ce cluster de bases de données.
- `--performance-insights-kms-key-id` : l'identifiant AWS KMS key pour le chiffrement des données de Performance Insights. Si vous ne spécifiez aucune clé KMS, la clé par défaut de votre Compte AWS est utilisée.
- `--region` : la Région AWS où vous créez le cluster de bases de données. Celle-ci doit prendre en charge Aurora PostgreSQL Limitless Database.

Pour utiliser le VPC et le groupe de sécurité VPC par défaut, omettez les options `--db-subnet-group-name` et `--vpc-security-group-ids`.

Pour créer un cluster de bases de données principal

- ```
  aws rds create-db-cluster \
  ```
  ```

```
    --db-cluster-identifier my-limitless-cluster \
    --engine aurora-postgresql \
    --engine-version 16.6-limitless \
    --storage-type aurora-iopt1 \
    --cluster-scalability-type limitless \
    --master-username myuser \
    --master-user-password mypassword \
    --db-subnet-group-name mysubnetgroup \
    --vpc-security-group-ids sg-c7e5b0d2 \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
    --enable-cloudwatch-logs-exports postgresql
```

Pour plus d'informations, consultez create-db-cluster.

Création du groupe de partitions de base de données

Vous créez ensuite le groupe de partitions de base de données dans le cluster de bases de données que vous venez de créer. Les paramètres suivants sont obligatoires :

- `--db-shard-group-identifier` : le nom du groupe de partitions de base de données.

  L'identifiant du groupe de partitions de base de données respecte les contraintes suivantes :
  - Il doit être unique dans le Compte AWS et la Région AWS où vous le créez.
  - Il doit comporter entre 1 et 63 lettres, chiffres ou traits d'union.
  - Le premier caractère doit être une lettre.
  - Il ne peut pas se terminer par un trait d'union ou contenir deux traits d'union consécutifs.
  -
    > **Important**
    >
    > Après avoir créé le groupe de partitions de base de données, vous ne pouvez pas modifier l'identifiant du cluster de bases de données ou l'identifiant du groupe de partitions de base de données.
- `--db-cluster-identifier` : le nom du cluster de bases de données dans lequel vous créez le groupe de partitions de base de données.
- `--max-acu` : la capacité maximale du groupe de partitions de base de données. Elle doit être comprise entre 16 et 6 144 ACU. Pour les limites de capacité supérieures à 6 144 ACU, contactez AWS.

  Le nombre initial de routeurs et de partitions est déterminé par la capacité maximale que vous définissez lorsque vous créez le groupe de partitions de base de données. Plus la capacité maximale est élevée, plus le nombre de routeurs et de partitions créés dans le groupe de partitions de base de données est élevé. Pour plus d'informations, consultez Corrélation de la capacité maximale du groupe de partitions de base de données avec le nombre de routeurs et de partitions créés.

Les paramètres suivants sont facultatifs :

- `--compute-redundancy` : déterminer s'il faut créer des instances de secours pour le groupe de partitions de base de données. Ce paramètre peut présenter les valeurs suivantes :
  - `0` : crée un groupe de partitions de base de données sans instance de secours pour chaque partition. C'est la valeur par défaut.

- 1 : crée un groupe de partitions de base de données avec une instance de calcul de secours dans une zone de disponibilité (AZ) différente.
- 2 : crée un groupe de partitions de base de données avec deux instances de calcul de secours situées dans deux AZ différentes.

> **Note**
>
> Si vous définissez la redondance de calcul sur une valeur autre que zéro, le nombre total de partitions sera doublé ou triplé. Des frais supplémentaires vous seront alors facturés. La capacité des nœuds en veille de calcul est augmentée ou réduite verticalement pour être identique à celle du nœud d'écriture. Il n'est pas nécessaire de définir une plage de capacité distincte pour les instances de secours.

- `--min-acu` : la capacité minimale de votre groupe de partitions de base de données. Il doit comporter au moins 16 ACU, qui est la valeur par défaut.
- `--publicly-accessible|--no-publicly-accessible` : déterminer s'il faut attribuer des adresses IP accessibles au public au groupe de partitions de base de données. L'accès au groupe de partitions de base de données est contrôlé par les groupes de sécurité utilisés par le cluster.

  La valeur par défaut est `--no-publicly-accessible`.

> **Note**
>
> Ce paramètre ne peut plus être modifié une fois le groupe de partitions de base de données créé.

Pour créer le groupe de partitions de base de données

- ```
  aws rds create-db-shard-group \
      --db-shard-group-identifier my-db-shard-group \
      --db-cluster-identifier my-limitless-cluster \
      --max-acu 1000
  ```
  ```

Utilisation de groupes de partitions de base de données

Les tâches suivantes permettent d'ajouter et de gérer un groupe de partitions de base de données dans Aurora PostgreSQL Limitless Database.

Rubriques

Connexion de votre cluster de bases de données dans Aurora PostgreSQL Limitless Database

Pour utiliser Aurora PostgreSQL Limitless Database, connectez-vous au point de terminaison du cluster d'écriture ou de lecture. Vous pouvez utiliser `psql` ou de tout autre utilitaire de connexion compatible avec PostgreSQL :

```
$ psql -h DB_cluster_endpoint -p port_number -U database_username -d postgres_limitless
```

L'exemple suivant illustre l'utilisation du point de terminaison du cluster de bases de données que vous avez créé dans Interface de ligne de commande (CLI).

```
$ psql -h my-limitless-cluster.cluster-ckifpdyyyxxx.us-east-1.rds.amazonaws.com -p 5432
 -U postgres -d postgres_limitless
```

> **Note**
>
> La base de données par défaut pour le groupe de partitions de base de données dans Aurora PostgreSQL Limitless Database est `postgres_limitless`.

Utilisation de Limitless Connection Plugin

Lors de la connexion à Aurora PostgreSQL Limitless Database, les clients se connectent via le point de terminaison du cluster et sont acheminés vers un routeur de transactions par Amazon Route 53. Cependant, la capacité de Route 53 à équilibrer la charge est limitée et peut entraîner des charges de travail inégales sur les routeurs de transactions. Le Limitless Connection Plugin du pilote JDBC AWS résout ce problème en effectuant un équilibrage de charge côté client en tenant compte de la charge. Pour plus d'informations sur le pilote JDBC AWS, consultez Connexion à Aurora PostgreSQL avec le pilote Amazon Web Services (AWS) JDBC.

Recherche du nombre de routeurs et de partitions dans un groupe de partitions de base de données

Vous pouvez utiliser la requête suivante pour rechercher le nombre de routeurs et de partitions :

```
SELECT * FROM rds_aurora.limitless_subclusters;

 subcluster_id | subcluster_type
---------------+-----------------
 1             | router
 2             | router
 3             | shard
 4             | shard
 5             | shard
 6             | shard
```

Description des groupes de partitions de base de données

Utilisez la commande `describe-db-shard-groups` de l'AWS CLI pour décrire vos groupes de partitions de base de données. Le paramètre suivant est facultatif :

- `--db-shard-group-identifier` : le nom du groupe de partitions de base de données.

L'exemple suivant décrit un groupe de partitions de base de données spécifique.

```
aws rds describe-db-shard-groups --db-shard-group-identifier my-db-shard-group
```

La sortie ressemble à l'exemple suivant.

```
{
    "DBShardGroups": [
        {
            "DBShardGroupResourceId": "shardgroup-8986d309a93c4da1b1455add17abcdef",
            "DBShardGroupIdentifier": "my-shard-group",
            "DBClusterIdentifier": "my-limitless-cluster",
            "MaxACU": 1000.0,
            "ComputeRedundancy": 0,
            "Status": "available",
            "PubliclyAccessible": false,
```

```
            "Endpoint": "my-limitless-cluster.limitless-ccetp2abcdef.us-
east-1.rds.amazonaws.com"
        }
    ]
}
```

Redémarrage d'un groupe de partitions de base de données

Il peut être nécessaire de redémarrer le groupe de partitions de base de données, notamment lorsque le paramètre `max_connections` est modifié à la suite d'un changement de capacité maximale.

Vous pouvez utiliser la AWS Management Console ou l'AWS CLI pour modifier la capacité d'un groupe de partitions de base de données.

Console

Utilisez la procédure suivante.

Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

1. Accédez à la page Bases de données.
2. Sélectionnez le groupe de partitions de base de données que vous souhaitez redémarrer.
3. Pour Actions, choisissez Redémarrer.
4. Choisissez Confirmer.

Interface de ligne de commande (CLI)

Pour redémarrer votre groupe de partitions de base de données, utilisez la commande `reboot-db-shard-group` de l'AWS CLI avec le paramètre suivant :

- `--db-shard-group-identifier` : le nom du groupe de partitions de base de données.

L'exemple suivant illustre le redémarrage d'un groupe de partitions de base de données.

```
aws rds reboot-db-shard-group --db-shard-group-identifier my-db-shard-group
```

Modification de la capacité d'un groupe de partitions de base de données

Vous pouvez utiliser la AWS Management Console ou l'AWS CLI pour modifier la capacité d'un groupe de partitions de base de données.

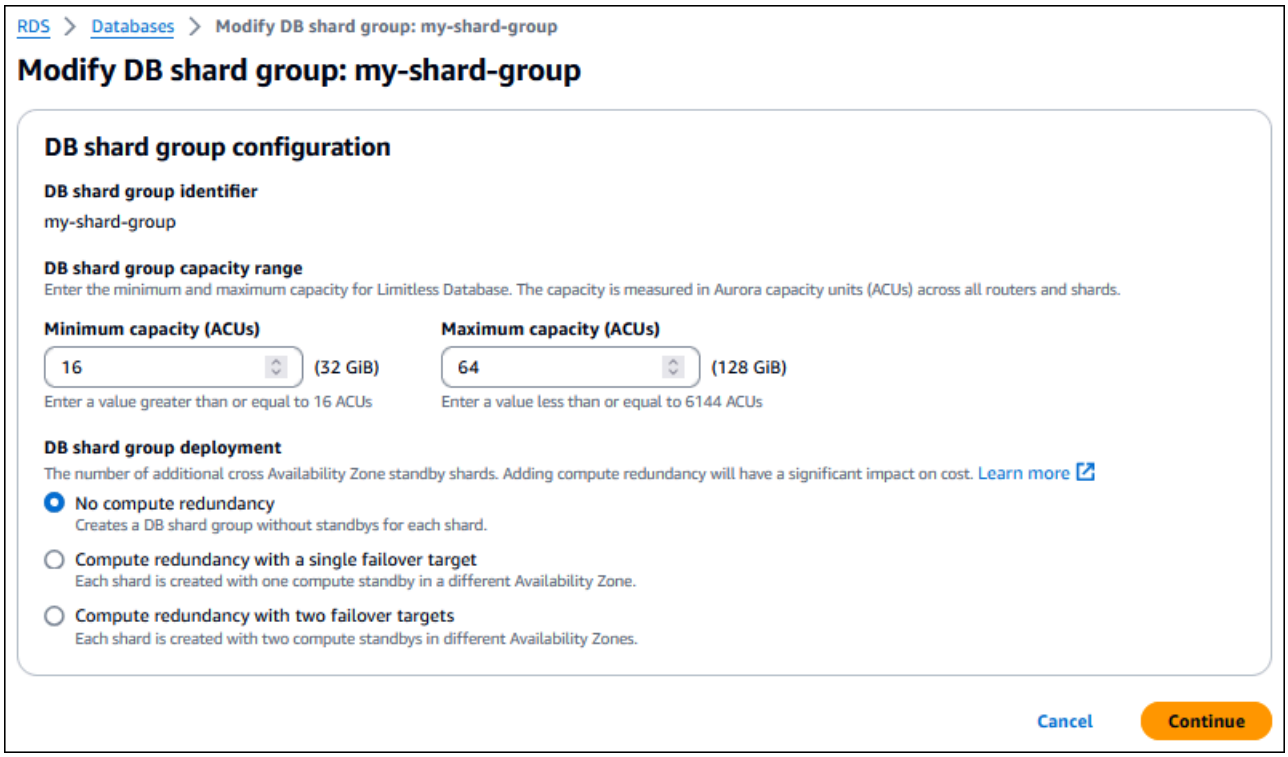
Console

Utilisez la procédure suivante.

Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

1. Accédez à la page Bases de données.
2. Sélectionnez le groupe de partitions de base de données que vous souhaitez modifier.
3. Pour Actions, choisissez Modifier.

   La page Modifier le groupe de partitions de base de données s'affiche.



4. Entrez une nouvelle valeur Capacité minimale (ACU), par exemple **`100`**.
5. Entrez une nouvelle valeur Capacité maximale (ACU), par exemple **`1000`**.
6. Choisissez Continuer.

   La page de confirmation s'affiche, avec un résumé de vos modifications.

7. Passez en revue vos modifications, puis choisissez Modifier le groupe de partitions de base de données.

Interface de ligne de commande (CLI)

Utilisez la commande `modify-db-shard-group` de l'AWS CLI avec les paramètres suivants :

- `--db-shard-group-identifier` : nom du groupe de partitions de base de données.
- `--max-acu` : la nouvelle capacité maximale du groupe de partitions de base de données. Vous pouvez définir la capacité maximale du groupe de partitions de base de données entre 16 et 6 144 ACU. Pour les limites de capacité supérieures à 6 144 ACU, contactez AWS.

  Le nombre de routeurs et de partitions ne change pas.
- `--min-acu` : la nouvelle capacité minimale de votre groupe de partitions de base de données. Il doit comporter au moins 16 ACU, qui est la valeur par défaut.

L'exemple ci-dessous montre comment modifier, via la CLI, la plage de capacité d'un groupe de partitions de base de données à 100 – 1 000 ACU.

```
aws rds modify-db-shard-group \
    --db-shard-group-identifier my-db-shard-group \
    --min-acu 100 \
    --max-acu 1000
```

Fractionnement d’une partition dans un groupe de partitions de base de données

Vous pouvez fractionner manuellement une partition d’un groupe de partitions de base de données en deux partitions plus petites. C’est ce qu’on appelle un fractionnement de partition initiée par l’utilisateur.

Aurora PostgreSQL Limitless Database peut également fractionner des partitions lorsqu’elles contiennent un volume de données très important ou lorsqu’elles sont fortement sollicitées. C’est ce qu’on appelle un fractionnement de partition initiée par le système.

Rubriques

Prérequis

Les fractionnements de partitions initiés par l’utilisateur sont soumis aux conditions préalables suivantes :

- Vous devez disposer d’un groupe de partitions de base de données.
- Le groupe de partitions de base de données ne peut pas être vide : il doit contenir au moins une table partitionnée.
- Un utilisateur doit disposer du privilège `rds_aurora_limitless_cluster_admin`. Ce privilège est détenu par le rôle `rds_superuser` ; il est donc également accordé à l’utilisateur principal. Le `rds_superuser` peut accorder le privilège à d’autres utilisateurs :

```
/* Logged in as the master user or a user with rds_superuser privileges */
CREATE USER username;
GRANT rds_aurora_limitless_cluster_admin to username;
```

- Vous devez connaître l’ID du sous-cluster (nœud) de la partition que vous souhaitez fractionner. Vous pouvez identifier les ID à l’aide de la requête suivante :

```
SELECT * FROM rds_aurora.limitless_subclusters;

 subcluster_id | subcluster_type
---------------+-----------------
 1             | router
 2             | router
 3             | shard
 4             | shard
 5             | shard
 6             | shard
```

Pour activer les fractionnements de partitions initiés par le système, configurez les paramètres suivants du cluster de bases de données dans un groupe de paramètres personnalisé associé à votre cluster de bases de données :

Paramètre	Valeur
`rds_aurora.limitless_enable_auto_scale`	on
`rds_aurora.limitless_auto_scale_options`	`split_shard` ou `add_router,split_shard`
`rds_aurora.limitless_finalize_split_shard_mode`	Ce paramètre détermine la manière dont les fractionnements de partitions initiées par le système sont finalisés. La valeur peut être l'une des suivantes : • `user_initiated` : vous décidez quand finaliser le fractionnement de partitions. C'est la valeur par défaut. • `immediate` : les fractionnements de partitions sont finalisés immédiatement. Pour plus d'informations, consultez Finalisation des fractionnements de partitions.

Paramètre	Valeur
	Note Ce paramètre s'applique uniquement aux fractionnements de partitions initiés par le système.

Pour plus d'informations, consultez Groupes de paramètres de cluster de bases de données pour les clusters de bases de données Amazon Aurora.

Fractionnement d'une partition

Pour fractionner une partition dans un groupe de partitions de base de données, utilisez la fonction `rds_aurora.limitless_split_shard`. Cette fonction lance une tâche de fractionnement de partition qui s'exécute de manière asynchrone.

```
SELECT rds_aurora.limitless_split_shard('subcluster_id');
```

Une fois la tâche soumise avec succès, attendez que l'ID de tâche soit renvoyé, par exemple :

```
SELECT rds_aurora.limitless_split_shard('3');

    job_id
---------------
 1691300000000
(1 row)
```

Note

Les opérations simultanées de fractionnement de partition ne sont pas prises en charge. Effectuez chaque opération l'une après l'autre, en veillant à finaliser une opération avant d'en initier une nouvelle.

Suivi des fractionnements de partitions

Vous pouvez utiliser l'ID de tâche pour suivre une tâche de fractionnement de partition. Pour décrire une tâche particulière et obtenir plus de détails à son sujet, exécutez la requête suivante :

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs(job_id);
```

Exemples :

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs(1691300000000);

    job_id     |    action   |      job_details      | status  |    submission_time
 |                 message
---------------+-------------+-----------------------+---------
+------------------------+-------------------------------------------
 1691300000000 | SPLIT_SHARD | Split Shard 3 by User | SUCCESS | 2023-08-06 05:33:20+00
 | Scaling job succeeded.                 +
               |             |                       |         |
 | New shard instance with ID 7 was created.
(1 row)
```

La requête génère une erreur si la tâche spécifiée en entrée n'existe pas.

```
SELECT * from rds_aurora.limitless_list_shard_scale_jobs(1691300000001);

ERROR:  no job found with the job ID provided
```

Vous pouvez suivre l'état de toutes les tâches de fractionnement de partition en utilisant la même requête sans ID de tâche, par exemple :

```
SELECT * FROM rds_aurora.limitless_list_shard_scale_jobs();

    job_id     |   action    |  job_details          |   status    |    submission_time
     |                 message
---------------+-------------+-----------------------+-------------
+------------------------
+-------------------------------------------------------------
 1691200000000 | SPLIT_SHARD | Split Shard 3 by User | IN_PROGRESS | 2023-08-05
 01:46:40+00 |
 1691300000000 | SPLIT_SHARD | Split Shard 4 by User | SUCCESS     | 2023-08-06
 05:33:20+00 | Scaling job succeeded. +
```

```
               |            |                       |            |
    | New shard instance with ID 7 was created.
 1691400000000 | SPLIT_SHARD | Split Shard 5 by User | FAILED      | 2023-08-07
 09:20:00+00 | Error occurred for the add shard job 1691400000000.
               |            |                       |            |
    | Retry the command. If the issue persists, contact AWS Support.
 1691500000000 | SPLIT_SHARD | Split Shard 5 by User | CANCELED    | 2023-08-07
 09:20:00+00 | Scaling job was cancelled.
(4 rows)
```

La tâche peut présenter l'un des états suivants :

- `IN_PROGRESS` : la tâche de fractionnement de partition a été soumise et est en cours. Une seule tâche peut être en cours d'exécution à la fois.
- `PENDING` : la tâche de fractionnement de partition attend que vous la finalisiez. Pour plus d'informations, consultez Finalisation des fractionnements de partitions.
- `CANCELLATION_IN_PROGRESS` : la tâche de fractionnement de partition est annulée par l'utilisateur.
- `CANCELED` : la tâche de fractionnement de partition a été annulée par l'utilisateur ou le système.
- `SUCCESS` : la tâche de fractionnement de partition s'est terminée avec succès. Le champ `message` contient l'ID d'instance de la nouvelle partition.
- `FAILED` : la tâche de fractionnement de partition a échoué. Le champ `message` contient les détails de l'échec et toutes les actions qui peuvent être prises en conséquence.

Finalisation des fractionnements de partitions

La finalisation est la dernière étape du processus de fractionnement de partition. Cela peut engendrer une période d'indisponibilité. Si vous démarrez une tâche de fractionnement de partition, la finalisation est effectuée immédiatement une fois la tâche terminée.

Parfois, le système fractionne les partitions en fonction de la charge de travail, lorsque vous avez activé les fractionnements de partitions initiés par le système à l'aide du paramètre `rds_aurora.limitless_enable_auto_scale`.

Dans ce cas, vous pouvez décider si la finalisation doit avoir lieu immédiatement ou au moment de votre choix. Le paramètre `rds_aurora.limitless_finalize_split_shard_mode` de cluster de bases de données vous permet de définir quand cette opération a lieu :

- Si vous définissez la valeur sur `immediate`, la finalisation se produit immédiatement.
- Si vous définissez la valeur sur `user_initiated`, vous devez finaliser la tâche de fractionnement de partition manuellement.

  Un événement RDS vous est envoyé et l'état de la tâche de fractionnement de partition est défini sur `PENDING`.

Lorsque ce paramètre est défini sur `user_initiated`, vous utilisez la fonction `rds_aurora.limitless_finalize_split_shard` pour finaliser la tâche de fractionnement de partition :

```
SELECT * FROM rds_aurora.limitless_finalize_split_shard(job_id);
```

> **Note**
>
> Cette fonction s'applique uniquement aux fractionnements de partitions initiés par le système, et non par vous.

Annulation d'un fractionnement de partition

Les fractionnements de partitions initiés par l'utilisateur ou par le système `IN_PROGRESS` ou `PENDING` peuvent être annulés. Vous avez besoin de l'ID de la tâche pour pouvoir l'annuler.

```
SELECT * from rds_aurora.limitless_cancel_shard_scale_jobs(job_id);
```

Aucune sortie n'est renvoyée sauf en cas d'erreur. Vous pouvez suivre l'annulation à l'aide d'une requête de suivi des tâches.

Ajout d’un routeur à un groupe de partitions de base de données

Vous pouvez ajouter un routeur à un groupe de partitions de base de données.

Rubriques

Prérequis

Les conditions préalables suivantes doivent être remplies avant d’ajouter un routeur :

- Vous devez disposer d’un groupe de partitions de base de données.
- Un utilisateur doit disposer du privilège `rds_aurora_limitless_cluster_admin`. Ce privilège est détenu par le rôle `rds_superuser` ; il est donc également accordé à l’utilisateur principal. Le `rds_superuser` peut accorder le privilège à d’autres utilisateurs :

```
/* Logged in as the master user or a user with rds_superuser privileges */
CREATE USER username;
GRANT rds_aurora_limitless_cluster_admin to username;
```

> **Note**
>
> Si vous modifiez le certificat CA par défaut de votre Compte AWS après la création du groupe de partitions de base de données, le nouveau routeur utilisera le nouveau certificat CA, qui est différent du certificat CA du routeur existant. Il se peut que certaines connexions échouent, selon la configuration de votre magasin d’approbations.

- Pour activer l’ajout de routeur initié par le système, configurez les paramètres suivants du cluster de bases de données dans un groupe de paramètres personnalisé associé à votre cluster de bases de données :

Paramètre	Valeur
`rds_aurora.limitless_enable_auto_scale`	`on`
`rds_aurora.limitless_auto_scale_options`	`add_router` ou `add_router,split_shard`

Pour plus d'informations, consultez Groupes de paramètres de cluster de bases de données pour les clusters de bases de données Amazon Aurora.

Ajout d'un routeur

Pour ajouter un routeur, utilisez la fonction `rds_aurora.limitless_add_router`. Cette fonction lance une tâche d'ajout de routeur qui s'exécute de manière asynchrone.

```
SELECT rds_aurora.limitless_add_router();
```

Une fois la tâche soumise avec succès, attendez que l'ID de tâche soit renvoyé, par exemple :

```
    job_id
---------------
 1691300000000
(1 row)
```

> **Note**
>
> Les opérations simultanées d'ajout de routeur ne sont pas prises en charge. Procédez aux opérations d'ajout une par une et attendez la fin de chacune avant de lancer la suivante.

Suivi des ajouts de routeurs

Vous pouvez utiliser l'ID de tâche pour suivre une tâche d'ajout de routeur. Pour décrire une tâche particulière et obtenir plus de détails à son sujet, exécutez la requête suivante :

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs(job_id);
```

Exemples :

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs(1691300000000);

    job_id     |   action   |        job_details       | status  |    submission_time
   |                   message
---------------+------------+--------------------------+---------
+------------------------+-------------------------------------------
 1691300000000 | ADD_ROUTER | Add 1 new Router by User | SUCCESS | 2023-08-06
 05:33:20+00 | Scaling job succeeded.                   +
               |            |                          |         |
   | New router instance with ID 7 was created.
(1 row)
```

La requête génère une erreur si la tâche spécifiée en entrée n'existe pas.

```
SELECT * from rds_aurora.limitless_list_router_scale_jobs(1691300000001);

ERROR:  no job found with the job ID provided
```

Vous pouvez suivre l'état de toutes les tâches d'ajout de routeurs en utilisant la même requête sans ID de tâche, par exemple :

```
SELECT * FROM rds_aurora.limitless_list_router_scale_jobs();

    job_id     |   action   |        job_details       |   status    |
 submission_time     |                 message
---------------+------------+--------------------------+-------------
+------------------------+-------------------------------------------
 1691200000000 | ADD_ROUTER | Add 1 new Router by User | IN_PROGRESS | 2023-08-05
 01:46:40+00 |
 1691300000000 | ADD_ROUTER | Add 1 new Router by User | SUCCESS     | 2023-08-06
 05:33:20+00 | Scaling job succeeded.                 +
               |            |                          |             |
       | New router instance with ID 7 was created.
 1691400000000 | ADD_ROUTER | Add 1 new Router by User | FAILED      | 2023-08-07
 09:20:00+00 | Error occurred for the add router job 1691400000000.
               |            |                          |             |
       | Retry the command. If the issue persists, contact AWS Support.
```

```
 1691500000000 | ADD_ROUTER | Add 1 new Router by User | CANCELED    | 2023-08-07
 09:20:00+00 | Scaling job was cancelled.
(4 rows)
```

La tâche peut présenter l'un des états suivants :

- `IN_PROGRESS` : la tâche d'ajout de routeur a été soumise et est en cours. Une seule tâche peut être en cours d'exécution à la fois.
- `CANCELLATION_IN_PROGRESS` : la tâche d'ajout de routeur est annulée par l'utilisateur.
- `CANCELED` : la tâche d'ajout de routeur a été annulée par l'utilisateur ou le système.
- `SUCCESS` : la tâche d'ajout de routeur a bien été effectuée. Le champ `message` contient l'ID d'instance du nouveau routeur.
- `FAILED` : la tâche d'ajout de routeur a échoué. Le champ `message` contient les détails de l'échec et toutes les actions qui peuvent être prises en conséquence.

> **Note**
>
> Il n'y a aucun état `PENDING`, car les ajouts de routeurs n'ont pas besoin d'être finalisés. Ils n'entraînent aucune durée d'indisponibilité.

Annulation de l'ajout d'un routeur

Vous pouvez annuler l'ajout d'un routeur présentant l'état `IN_PROGRESS`. Vous avez besoin de l'ID de la tâche pour pouvoir l'annuler.

```
SELECT * from rds_aurora.limitless_cancel_router_scale_jobs(job_id);
```

Aucune sortie n'est renvoyée sauf en cas d'erreur. Vous pouvez suivre l'annulation à l'aide d'une requête de suivi des tâches.

Suppression d’un groupe de partitions de base de données

Vous pouvez supprimer un groupe de partitions de base de données si nécessaire. La suppression du groupe de partitions de base de données supprime les nœuds de calcul (partitions et routeurs), tout en conservant le stockage.

Console

Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.

1. Accédez à la page Bases de données.
2. Sélectionnez le groupe de partitions de base de données que vous souhaitez supprimer.
3. Pour Actions, choisissez Supprimer.
4. Saisissez **`delete me`** dans la case, puis choisissez Supprimer.

Le groupe de partitions de base de données est supprimé.

AWS CLI

Pour supprimer votre groupe de partitions de base de données, utilisez la commande `delete-db-shard-group` de l’AWS CLI avec le paramètre suivant :

- `--db-shard-group-identifier` : nom du groupe de partitions de base de données.

L’exemple suivant illustre la suppression d’un groupe de partitions de base de données dans le cluster de bases de données Aurora PostgreSQL créé précédemment.

```
aws rds delete-db-shard-group --db-shard-group-identifier my-db-shard-group
```

Ajout d'un groupe de partitions de base de données à un cluster de bases de données Aurora PostgreSQL Limitless Database existant

Vous pouvez créer un groupe de partitions de base de données dans un cluster de bases de données existant, par exemple si vous restaurez un cluster de bases de données ou si vous avez supprimé le groupe de partitions de base de données.

Pour plus d'informations sur les exigences relatives au cluster de bases de données principal et au groupe de partitions de base de données, consultez Exigences et considérations relatives à Aurora PostgreSQL Limitless Database.

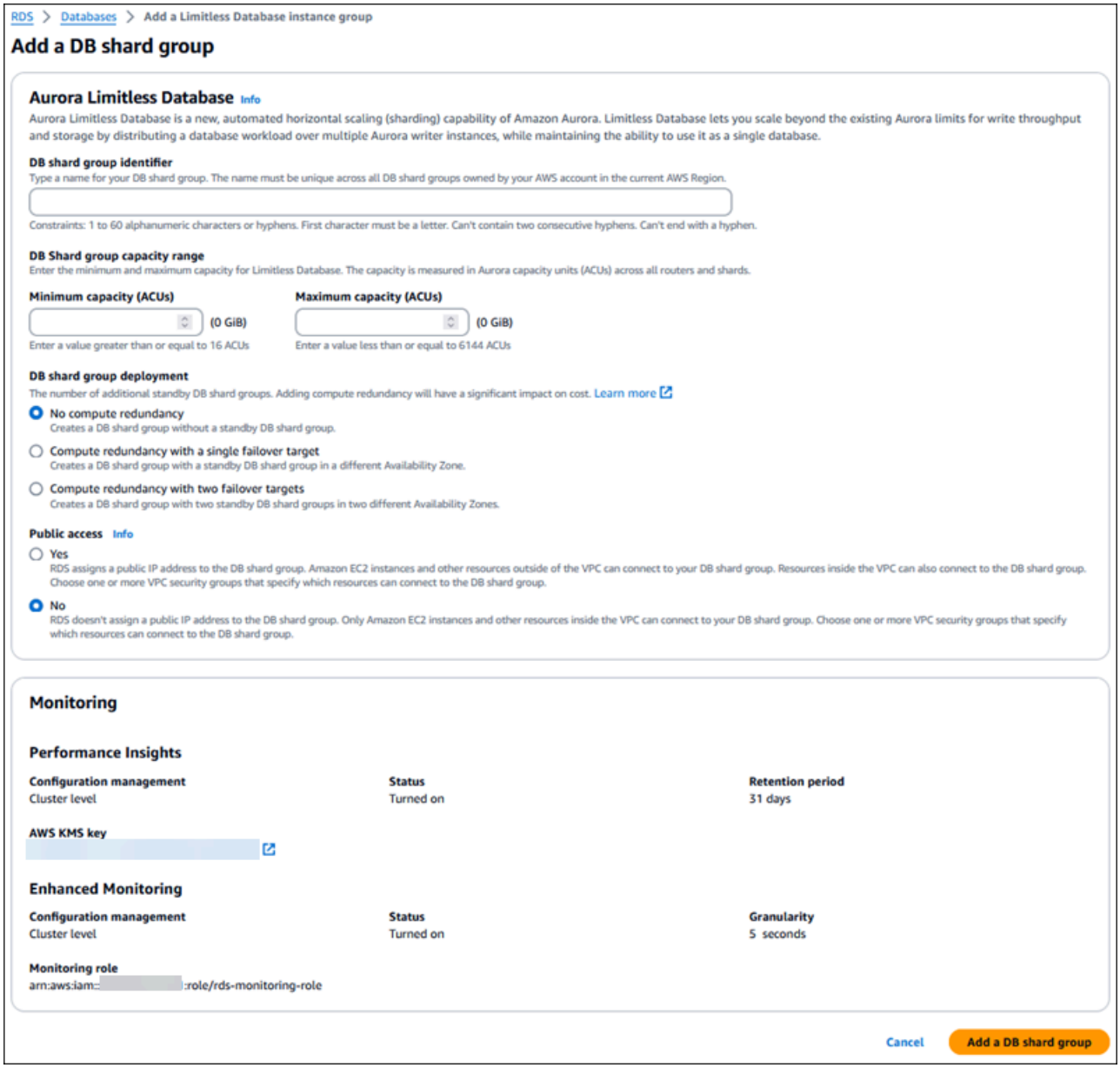
> **Note**
>
> Un cluster ne peut contenir qu'un seul groupe de partitions de base de données.
> Le cluster de bases de données Limitless Database doit présenter l'état `available` pour que vous puissiez créer un groupe de partitions de base de données.

Console

Vous pouvez utiliser la AWS Management Console pour ajouter un groupe de partitions de base de données à un cluster de bases de données existant.

Pour ajouter un groupe de partitions de base de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Accédez à la page Bases de données.
3. Sélectionnez le cluster de bases de données Limitless Database dans lequel vous souhaitez ajouter un groupe de partitions de base de données.
4. Dans Actions, choisissez Ajouter un groupe de partitions de base de données.



5. Entrez un identifiant de groupe de partitions de base de données.

> **Important**
>
> Après avoir créé le groupe de partitions de base de données, vous ne pouvez pas modifier l'identifiant du cluster de bases de données ou l'identifiant du groupe de partitions de base de données.

6. Entrez la Capacité minimale (ACU). Utilisez une valeur d'au moins 16 ACU.
7. Entrez la Capacité maximale (ACU). Utilisez une valeur comprise entre 16 et 6 144 ACU.

Pour plus d'informations, consultez Corrélation de la capacité maximale du groupe de partitions de base de données avec le nombre de routeurs et de partitions créés.

8. Pour le déploiement d'un groupe de partitions de base de données, choisissez si vous souhaitez créer des instances de secours pour le groupe de partitions de base de données :

   - Aucune redondance de calcul : crée un groupe de partitions de base de données sans instance de secours pour chaque partition. C'est la valeur par défaut.
   - Redondance des calculs avec une seule cible de basculement : crée un groupe de partitions de base de données avec une instance de calcul de secours dans une zone de disponibilité (AZ) différente.
   - Redondance de calcul avec deux cibles de basculement : crée un groupe de partitions de base de données avec deux instances de calcul de secours situées dans deux zones de disponibilité différentes.

9. Choisissez si le groupe de partitions de base de données doit être accessible publiquement.

   > **Note**
   >
   > Ce paramètre ne peut plus être modifié une fois le groupe de partitions de base de données créé.

10. Choisissez Ajouter un groupe de partitions de base de données.

AWS CLI

Utilisez la commande `create-db-shard-group` de l'AWS CLI pour créer un nouveau groupe de partitions de base de données.

Les paramètres suivants sont obligatoires :

- `--db-cluster-identifier` : le cluster de bases de données auquel appartient le groupe de partitions de base de données.
- `--db-shard-group-identifier` : nom du groupe de partitions de base de données.

  L'identifiant du groupe de partitions de base de données respecte les contraintes suivantes :

  - Il doit être unique dans le Compte AWS et la Région AWS où vous le créez.
  - Il doit comporter entre 1 et 63 lettres, chiffres ou traits d'union.
  - Le premier caractère doit être une lettre.
  - Il ne peut pas se terminer par un trait d'union ou contenir deux traits d'union consécutifs.

  > **Important**
  >
  > Après avoir créé le groupe de partitions de base de données, vous ne pouvez pas modifier l'identifiant du cluster de bases de données ou l'identifiant du groupe de partitions de base de données.

- `--max-acu` : la capacité maximale du groupe de partitions de base de données. Utilisez une valeur comprise entre 16 et 6 144 ACU.

Les paramètres suivants sont facultatifs :

- `--compute-redundancy` : déterminer s'il faut créer des instances de secours pour le groupe de partitions de base de données. Ce paramètre peut présenter les valeurs suivantes :
  - `0` : crée un groupe de partitions de base de données sans instance de secours pour chaque partition. C'est la valeur par défaut.
  - `1` : crée un groupe de partitions de base de données avec une instance de calcul de secours dans une zone de disponibilité (AZ) différente.
  - `2` : crée un groupe de partitions de base de données avec deux instances de calcul de secours situées dans deux AZ différentes.

> **Note**
>
> Si vous définissez la redondance de calcul sur une valeur autre que zéro, le nombre total de nœuds sera doublé ou triplé. Des frais supplémentaires vous seront alors facturés.

- `--min-acu` : la capacité minimale de votre groupe de partitions de base de données. Il doit comporter au moins 16 ACU, qui est la valeur par défaut.
- `--publicly-accessible|--no-publicly-accessible` : déterminer s'il faut attribuer des adresses IP accessibles au public au groupe de partitions de base de données. L'accès au groupe de partitions de base de données est contrôlé par les groupes de sécurité utilisés par le cluster.

  La valeur par défaut est `--no-publicly-accessible`.

> **Note**
>
> Ce paramètre ne peut plus être modifié une fois le groupe de partitions de base de données créé.

L'exemple suivant illustre la création d'un groupe de partitions de base de données dans un cluster de bases de données Aurora.

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-db-cluster \
    --db-shard-group-identifier my-new-shard-group \
    --max-acu 1000
```

La sortie ressemble à l'exemple suivant.

```
{
    "Status": "CREATING",
    "Endpoint": "my-db-cluster.limitless-ckifpdyyyxxx.us-east-1.rds.amazonaws.com",
    "PubliclyAccessible": false,
    "DBClusterIdentifier": "my-db-cluster",
    "MaxACU": 1000.0,
    "DBShardGroupIdentifier": "my-new-shard-group",
    "DBShardGroupResourceId": "shardgroup-8986d309a93c4da1b1455add17abcdef",
    "ComputeRedundancy": 0
```

```
}
```

Création de tables Aurora PostgreSQL Limitless Database

Trois types de tables contiennent vos données dans Aurora PostgreSQL Limitless Database :

- Standard : constitue le type de table par défaut dans Aurora PostgreSQL Limitless Database. Vous créez des tables standard à l'aide de la commande CREATE TABLE et vous pouvez exécuter des opérations de langage de description des données (DDL) et de langage de manipulation des données (DML) sur ces tables.

  Les tables standard ne sont pas des tables distribuées. Elles sont stockées sur l'une des partitions choisies en interne par le système.
- Partitionnées : ces tables sont réparties sur plusieurs partitions. Les données sont réparties entre les partitions en fonction des valeurs des colonnes désignées dans la table. Cet ensemble de colonnes s'appelle une clé de partition.
- Référence : ces tables sont répliquées sur toutes les partitions. Elles sont utilisées pour les données de référence rarement modifiées, telles que les catalogues de produits et les codes postaux.

  Les requêtes de jointure entre les tables de référence et les tables partitionnées peuvent être exécutées sur des partitions, en vue d'éviter les mouvements de données inutiles entre les partitions et les routeurs.

Les tables sans limite peuvent être créées de deux façons :

- Création de tables sans limite à l'aide de variables : utilisez cette méthode lorsque vous souhaitez créer de nouvelles tables partitionnées et de référence.
- Conversion de tables standard en tables sans limite : utilisez cette méthode lorsque vous souhaitez convertir des tables standard existantes en tables partitionnées et en tables de référence.

Nous fournissons également des exemples de schémas pour Aurora PostgreSQL Limitless Database.

Création de tables sans limite à l'aide de variables

Vous pouvez utiliser des variables pour créer des tables partitionnées et des tables de référence en définissant le mode de création de table. Les tables que vous créez utiliseront ensuite ce mode jusqu'à ce que vous définissiez un autre mode.

Utilisez les variables suivantes pour créer des tables partitionnées et des tables de référence :

- `rds_aurora.limitless_create_table_mode` : définissez cette variable de session sur `sharded` ou `reference`. La valeur par défaut de cette variable est `standard`.
- `rds_aurora.limitless_create_table_shard_key` : définissez cette variable de session en lui attribuant un tableau de noms de colonnes qui serviront de clés de partition. Cette variable est ignorée lorsque `rds_aurora.limitless_create_table_mode` n'est pas défini sur `sharded`.

  Formatez la valeur sous la forme `untyped array literal`, comme lorsque vous insérez des valeurs littérales dans une colonne de tableau. Pour plus d'informations, consultez Tableaux dans la documentation PostgreSQL.
- `rds_aurora.limitless_create_table_collocate_with` : attribuez à cette variable de session le nom d'une table spécifique pour colocaliser les tables nouvellement créées avec celle-ci.

  Si deux tables ou plus sont partitionnées à l'aide de la même clé de partition, vous pouvez explicitement les aligner (colocaliser). Lorsque plusieurs tables sont colocalisées, les lignes présentant les mêmes valeurs de clé de partition sont stockées dans la même partition. La collocation permet de limiter certaines opérations à une seule partition, en vue d'améliorer les performances.

> **Note**
>
> Toutes les clés primaires et uniques doivent inclure la clé de partition. Autrement dit, la clé de partition est un sous-ensemble de la clé primaire ou unique.
> Les tables Limitless présentent certaines limites. Pour plus d'informations, consultez Limitations du langage DDL et autres informations relatives à Aurora PostgreSQL Limitless Database.

Rubriques

- Exemples d'utilisation de variables pour créer des tables sans limite

- Vues de tables Aurora PostgreSQL Limitless Database

Exemples d’utilisation de variables pour créer des tables sans limite

Les exemples suivants expliquent comment utiliser ces variables pour créer des tables partitionnées et des tables de référence.

Créez une table partitionnée nommée `items`, avec la clé de partition `id`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"id"}';
CREATE TABLE items(id int, val int, item text);
COMMIT;
```

Créez une table partitionnée nommée `items`, avec une clé de partition composée des colonnes `item_id` et `item_cat`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"item_id", "item_cat"}';
CREATE TABLE items(item_id int, item_cat varchar, val int, item text);
COMMIT;
```

Créez une table partitionnée nommée `item_description`, avec une clé de partition composée des colonnes `item_id` et `item_cat`, puis localisez-la avec la table `items` de l’exemple précédent.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='sharded';
SET LOCAL rds_aurora.limitless_create_table_shard_key='{"item_id", "item_cat"}';
SET LOCAL rds_aurora.limitless_create_table_collocate_with='items';
CREATE TABLE item_description(item_id int, item_cat varchar, color_id int);
COMMIT;
```

Créez une table de référence nommée `colors`.

```
BEGIN;
SET LOCAL rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE colors(color_id int primary key, color varchar);
COMMIT;
```

Pour rétablir la variable de session `rds_aurora.limitless_create_table_mode` sur `standard`, utilisez l’instruction suivante :

```
RESET rds_aurora.limitless_create_table_mode;
```

Une fois cette variable réinitialisée, les tables sont créées sous forme de tables standard, ce qui est la configuration par défaut. Pour plus d'informations sur les tables standard, consultez Conversion de tables standard en tables sans limite.

Vues de tables Aurora PostgreSQL Limitless Database

Vous pouvez trouver des informations sur les tables Limitless Database en utilisant les vues suivantes.

rds_aurora.limitless_tables

La vue `rds_aurora.limitless_tables` contient des informations sur les tables sans limite et leurs types.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_tables;

 table_gid | local_oid | schema_name | table_name  | table_status | table_type  |
 distribution_key
-----------+-----------+-------------+-------------+--------------+-------------
+------------------
         5 |     18635 | public      | standard    | active       | standard    |
         6 |     18641 | public      | ref         | active       | reference   |
         7 |     18797 | public      | orders      | active       | sharded     |
 HASH (order_id)
         2 |     18579 | public      | customer    | active       | sharded     |
 HASH (cust_id)
(4 rows)
```

rds_aurora.limitless_table_collocations

La vue `rds_aurora.limitless_table_collocations` contient des informations sur les tables partitionnées colocalisées. Par exemple, les tables `orders` et `customers` sont colocalisées et ont le même `collocation_id`. Les tables `users` et `followers` sont colocalisées et ont le même `collocation_id`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_table_collocations ORDER BY
 collocation_id;

 collocation_id | schema_name | table_name
----------------+-------------+------------
          16002 | public      | orders
          16002 | public      | customers
          16005 | public      | users
          16005 | public      | followers
(4 rows)
```

rds_aurora.limitless_table_collocation_distributions

`rds_aurora.limitless_table_collocation_distributions` affiche la diffusion des clés pour chaque collocation.

```
postgres_limitless=> SELECT * FROM
 rds_aurora.limitless_table_collocation_distributions ORDER BY collocation_id,
 lower_bound;

 collocation_id | subcluster_id |     lower_bound      |     upper_bound
----------------+---------------+----------------------+----------------------
          16002 |             6 | -9223372036854775808 | -4611686018427387904
          16002 |             5 | -4611686018427387904 |                    0
          16002 |             4 |                    0 |  4611686018427387904
          16002 |             3 |  4611686018427387904 |  9223372036854775807
          16005 |             6 | -9223372036854775808 | -4611686018427387904
          16005 |             5 | -4611686018427387904 |                    0
          16005 |             4 |                    0 |  4611686018427387904
          16005 |             3 |  4611686018427387904 |  9223372036854775807
(8 rows)
```

Conversion de tables standard en tables sans limite

Vous pouvez convertir des tables standard en tables partitionnées ou en tables de référence. Pendant la conversion, les données sont déplacées de la table standard vers la table distribuée, après quoi la table source est supprimée. Les données sont déplacées à l'aide de la commande `INSERT INTO SELECT FROM`.

Table des matières

Création de tables partitionnées

Vous créez des tables partitionnées en exécutant la procédure `rds_aurora.limitless_alter_table_type_sharded` sur des tables standard. La procédure utilise une table standard et une liste de colonnes, qu'elle utilise comme clé de partition pour distribuer la table. La procédure s'exécute de manière synchrone et applique un verrouillage `ACCESS EXCLUSIVE` sur la table.

Une fois la procédure terminée, la table standard source est supprimée et une table partitionnée portant le même nom est disponible.

La procédure `rds_aurora.limitless_alter_table_type_sharded` utilise la syntaxe suivante :

```
postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('schema.table',
 ARRAY['shard_key1', 'shard_key2', ... 'shard_keyn']);
```

La procédure nécessite les paramètres suivants :

- `schema` : le schéma de base de données qui contient la table à partitionner. Si le schéma n'est pas indiqué, la procédure utilise le `search_path`.
- `table` : la table à partitionner.
- `shard_key`*n* : un tableau de colonnes de la table à utiliser comme clé de partition.

  Les valeurs des clés de partition étant des littéraux de chaîne, elles sont sensibles à la casse. Si une clé de partition contient un guillemet simple (’), utilisez un autre guillemet simple pour

l'échapper. Par exemple, si une colonne de table est nommée `customer's id`, utilisez `customer''s id` comme clé de partition. Il n'est pas nécessaire d'échapper les barres obliques inverses (\) ni les guillemets doubles (").

> **Note**
>
> Toutes les clés primaires et uniques doivent inclure la clé de partition. Autrement dit, la clé de partition est un sous-ensemble de la clé primaire ou unique.
> Dans les tables partitionnées, la contrainte CHECK ne prend pas en charge les expressions.
> Pour plus d'informations, consultez Constaintes.

Pour créer une table partitionnée

L'exemple suivant montre comment créer la table partitionnée `customer` à l'aide de la clé de partition `customer_id`.

1. Créez la table standard.

```
CREATE TABLE customer (customer_id INT PRIMARY KEY NOT NULL, zipcode INT, email
 VARCHAR);
```

2. Convertissez la table standard en table partitionnée.

```
postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('public.customer',
 ARRAY['customer_id']);

postgres=> \d

                  List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
(6 rows)
```

Création de tables colocalisées

Si deux tables ou plus sont partitionnées à l'aide de la même clé de partition, vous pouvez explicitement les aligner (colocaliser). Lorsque plusieurs tables sont colocalisées, les lignes présentant les mêmes valeurs de clé de partition sont stockées dans la même partition. La collocation permet de limiter certaines opérations à une seule partition, en vue d'améliorer les performances.

La procédure `rds_aurora.limitless_alter_table_type_sharded` s'utilise la syntaxe suivante :

```
postgres=> CALL
 rds_aurora.limitless_alter_table_type_sharded('schema.collocated_table',
 ARRAY['shard_key1', 'shard_key2', ... 'shard_keyn'], 'schema.sharded_table');
```

La procédure nécessite les paramètres suivants :

- `schema` : le schéma de base de données qui contient les tables à colocaliser. Si le schéma n'est pas indiqué, la procédure utilise le `search_path`.
- `collocated_table` : la table à colocaliser.
- `shard_key`*n* : un tableau de colonnes de la table à utiliser comme clé de partition.

  La clé de partition doit être identique à celle de la table partitionnée d'origine, avec les mêmes noms et types de colonnes.
- `sharded_table` : la table partitionnée avec laquelle vous colocalisez la `collocated_table`.

Pour créer une table colocalisée

1. Créez la première table partitionnée en suivant la procédure décrite dans Création de tables partitionnées.
2. Créez la table standard pour la table colocalisée.

   ```
   CREATE TABLE mytable2 (customer_id INT PRIMARY KEY NOT NULL, column1 INT, column2
    VARCHAR);
   ```
3. Convertissez la table standard en table colocalisée.

   ```
   postgres=> CALL rds_aurora.limitless_alter_table_type_sharded('public.mytable2',
   ARRAY['customer_id'], 'public.customer');
   ```

```
postgres=> \d

                   List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
 public | mytable2     | partitioned table | postgres_limitless
 public | mytable2_fs1 | foreign table     | postgres_limitless
 public | mytable2_fs2 | foreign table     | postgres_limitless
 public | mytable2_fs3 | foreign table     | postgres_limitless
 public | mytable2_fs4 | foreign table     | postgres_limitless
 public | mytable2_fs5 | foreign table     | postgres_limitless
(12 rows)
```

Création de tables de référence

Vous créez des tables de référence en exécutant la procédure `rds_aurora.limitless_alter_table_type_reference` sur des tables standard. Cette procédure réplique une table donnée sur toutes les partitions du groupe de partitions de base de données et modifie le type de table en référence. La procédure s'exécute de manière synchrone et applique un verrouillage `ACCESS EXCLUSIVE` sur la table.

Une fois la procédure terminée, la table standard source est supprimée et une table de référence portant le même nom est disponible.

La procédure `rds_aurora.limitless_alter_table_type_reference` utilise la syntaxe suivante :

```
postgres=> CALL rds_aurora.limitless_alter_table_type_reference('schema.table');
```

La procédure stockée nécessite les paramètres suivants :

- `schema` : le schéma de base de données qui contient la table à répliquer. Si le schéma n'est pas indiqué, la procédure utilise le `search_path`.
- `table` : la table à répliquer.

> **Note**
>
> La table standard à partir de laquelle vous créez la table de référence doit disposer d'une clé primaire.
> Dans les tables de référence, la contrainte `CHECK` ne prend pas en charge les expressions.
> La fonction précédente, `limitless_table_alter_type_reference`, est obsolète.

Pour créer une référence de référence

L'exemple suivant illustre la création d'une table de référence `zipcodes`.

1. Créez la table standard.

```
CREATE TABLE zipcodes (zipcode INT PRIMARY KEY, details VARCHAR);
```

2. Convertissez la table standard en table de référence.

```
CALL rds_aurora.limitless_alter_table_type_reference('public.zipcodes');

postgres=> \d

                   List of relations
 Schema |     Name     |       Type        |       Owner
--------+--------------+-------------------+--------------------
 public | customer     | partitioned table | postgres_limitless
 public | customer_fs1 | foreign table     | postgres_limitless
 public | customer_fs2 | foreign table     | postgres_limitless
 public | customer_fs3 | foreign table     | postgres_limitless
 public | customer_fs4 | foreign table     | postgres_limitless
 public | customer_fs5 | foreign table     | postgres_limitless
 public | zipcodes     | foreign table     | postgres_limitless
(7 rows)
```

   La sortie montre la table partitionnée `customer` et la table de référence `zipcodes`.

Exemples de schéma Aurora PostgreSQL Limitless Database

Nous fournissons les exemples de schémas suivants pour Aurora PostgreSQL Limitless Database :

- Exemple de schéma Limitless E-Commerce
- Limitless pgbench

Vous pouvez utiliser ces schémas pour créer rapidement un exemple de base de données et charger des données dans des tables Aurora PostgreSQL Limitless Database. Pour plus d'informations, consultez le dépôt GitHub.

Chargement de données dans Aurora PostgreSQL Limitless Database

Vous pouvez charger des données dans Aurora PostgreSQL Limitless Database à l'aide de la commande `COPY` ou de l'utilitaire de chargement de données.

> **Note**
>
> Vous pouvez charger des données dans des tables standard, partitionnées et de référence.

Table des matières

Utilisation de la commande COPY avec Aurora PostgreSQL Limitless Database

Vous pouvez utiliser la fonctionnalité \copy de l'utilitaire `psql` pour importer et exporter des données dans/depuis Aurora PostgreSQL Limitless Database

Découvrez comment utiliser la commande COPY pour charger des données dans Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database est compatible avec la fonctionnalité \copy de l'utilitaire `psql` d'importation de données.

Dans Limitless Database comme dans Aurora PostgreSQL, les éléments suivants ne sont pas pris en charge :

- Accès SSH direct aux instances de base de données : vous ne pouvez pas copier un fichier de données (au format .csv, par exemple) sur l'hôte de l'instance de base de données et exécuter `COPY` à partir du fichier.
- Utilisation de fichiers locaux sur l'instance de base de données : utilisez `COPY ... FROM STDIN` et `COPY ... TO STDOUT`.

La commande `COPY` de PostgreSQL propose des options permettant d'utiliser des fichiers locaux (`FROM/TO`) et de transmettre des données via une connexion entre le client et le serveur (`STDIN/STDOUT`). Pour plus d'informations, consultez COPY dans la documentation de PostgreSQL.

La commande `\copy` de l'utilitaire `psql` de PostgreSQL fonctionne avec les fichiers locaux de l'ordinateur sur lequel vous exécutez le client `psql`. Il invoque la commande correspondante `COPY ... FROM STDIN` ou `COPY ... FROM STDOUT` sur le serveur distant (par exemple, Limitless Database) auquel vous vous connectez. Il lit les données à partir du fichier local vers `STDIN` ou écrit dans ce fichier à partir de `STDOUT`.

Fractionnement de données en plusieurs fichiers

Les données sont stockées sur plusieurs partitions dans Aurora PostgreSQL Limitless Database. Pour accélérer le chargement des données en utilisant `\copy`, vous pouvez les diviser en plusieurs fichiers. Importez-les ensuite indépendamment pour chaque fichier de données en exécutant des commandes `\copy` distinctes en parallèle.

Imaginons que vous ayez un fichier de données d'entrée au format CSV contenant 3 millions de lignes à importer. Vous pouvez diviser le fichier en segments contenant chacun 200 000 lignes (soit 15 segments) :

```
split -l200000 data.csv data_ --additional-suffix=.csv -d
```

Le résultat est une série de fichiers, de `data_00.csv` à `data_14.csv`. Vous pouvez ensuite importer des données à l'aide de 15 commandes `\copy` parallèles, par exemple :

```
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table from '/tmp/data_00.csv';" postgres_limitless &
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_01.csv';" postgres_limitless &
...
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_13.csv';" postgres_limitless &
psql -h dbcluster.limitless-111122223333.aws-region.rds.amazonaws.com -U username -c
 "\copy test_table FROM '/tmp/data_14.csv';" postgres_limitless
```

Cette méthode permet d'importer le même volume de données environ 10 fois plus vite qu'en utilisant une seule commande `\copy`.

Utilisation de la commande COPY pour copier des données Limitless Database dans un fichier

Vous pouvez utiliser la commande \copy pour copier des données d'une table sans limite vers un fichier, comme illustré dans l'exemple suivant :

```
postgres_limitless=> \copy test_table TO '/tmp/test_table.csv' DELIMITER ',' CSV
 HEADER;
```

Utilisation de l’utilitaire de chargement de données dans Aurora PostgreSQL Limitless Database

Aurora fournit un utilitaire permettant de charger des données directement dans Limitless Database à partir d’un cluster de bases de données Aurora PostgreSQL ou d’une instance de base de données RDS pour PostgreSQL.

Pour utiliser l’utilitaire de chargement de données, procédez comme suit :

1. Prérequis
2. Préparation de la base de données source
3. Préparation de la base de données de destination
4. Création des informations d’identification de base de données
5. L’un des éléments suivants :
   - Configuration de l’authentification de la base de données et de l’accès aux ressources à l’aide d’un script (recommandé)
   - Configuration manuelle de l’authentification de base de données et de l’accès aux ressources
6. Chargement de données à partir d’un cluster de bases de données Aurora PostgreSQL ou d’une instance de base de données RDS pour PostgreSQL

Limitations

L’utilitaire de chargement de données présente les limitations suivantes :

- Les types de données suivants ne sont pas pris en charge : `enum`, `ARRAY`, `BOX`, `CIRCLE`, `LINE`, `LSEG`, `PATH`, `PG_LSN`, `PG_SNAPSHOT`, `POLYGON`, `TSQUERY`, `TSVECTOR` et `TXID_SNAPSHOT`.
- Les zéros de tête (`0`) sont supprimés du type de données `VARBIT` lors du chargement.
- La migration des données échoue si les tables de destination contiennent des clés étrangères.
- Le chargement de données à partir de clusters de bases de données multi-AZ RDS pour PostgreSQL n’est pas pris en charge.

Prérequis

L’utilitaire de chargement de données présente les prérequis suivants :

- La base de données source utilise Aurora PostgreSQL ou RDS pour PostgreSQL version 11.x ou ultérieure.
- La base de données source se trouve dans les même Compte AWS et Région AWS que dans le groupe de partitions de base de données de destination.
- Le cluster de bases de données ou l'instance de base de données source présente l'état `available`.
- Les tables de la base de données source et de la base de données sans limite présentent les mêmes noms de tables, noms de colonnes et types de données de colonne.
- Les tables source et de destination possèdent des clés primaires qui utilisent les mêmes colonnes et les mêmes ordres de colonnes.
- Vous devez disposer d'un environnement permettant de vous connecter à une base de données sans limite pour exécuter des commandes de chargement de données. Les commandes disponibles sont les suivantes :
  - `rds_aurora.limitless_data_load_start`
  - `rds_aurora.limitless_data_load_cancel`
- Pour CDC :
  - La base de données source et le groupe de partitions de base de données de destination doivent utiliser le même groupe de sous-réseaux de base de données, le même groupe de sécurité VPC et le même port de base de données. Ces configurations concernent les connexions réseau à la fois à la base de données source et aux routeurs du groupe de partitions de base de données.
  - Vous devez activer la réplication logique sur la base de données source. L'utilisateur de la base de données source doit disposer des privilèges nécessaires pour lire la réplication logique.

Préparation de la base de données source

Pour accéder à la base de données source lors du chargement des données, vous devez autoriser le trafic réseau entrant vers celle-ci. Procédez comme suit.

Pour autoriser le trafic réseau vers la base de données source

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Accédez à la page Groupes de sécurité.
3. Choisissez ID du groupe de sécurité pour le groupe de sécurité utilisé par le cluster ou l'instance de base de données source.

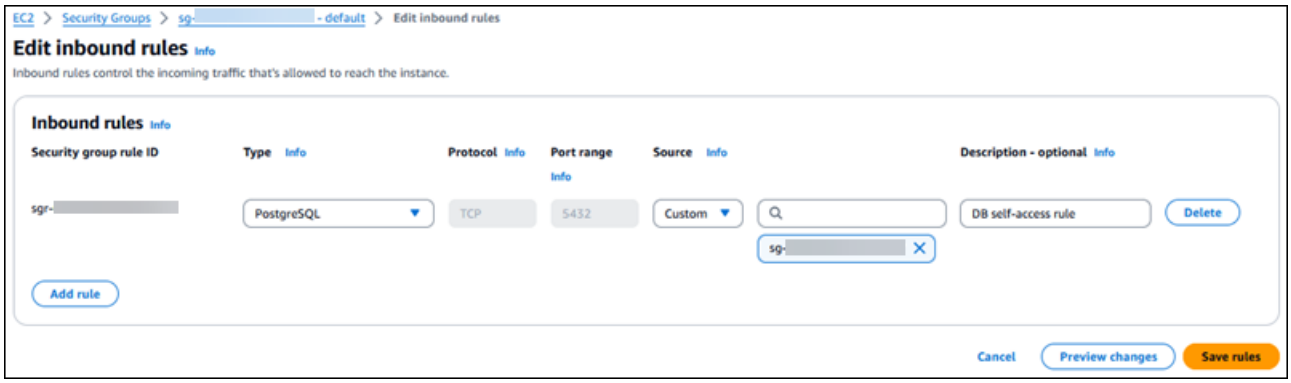
Son groupe de sécurité, par exemple, porte l'ID `sg-056a84f1712b77926`.

4. Dans l'onglet Règles entrantes :

   a. Choisissez Modifier les règles entrantes.

   b. Ajoutez une nouvelle règle entrante pour le cluster ou l'instance de base de données source :

      - Plage de ports : port de base de données pour la base de données source, généralement `5432`
      - ID du groupe de sécurité : `sg-056a84f1712b77926` dans cet exemple

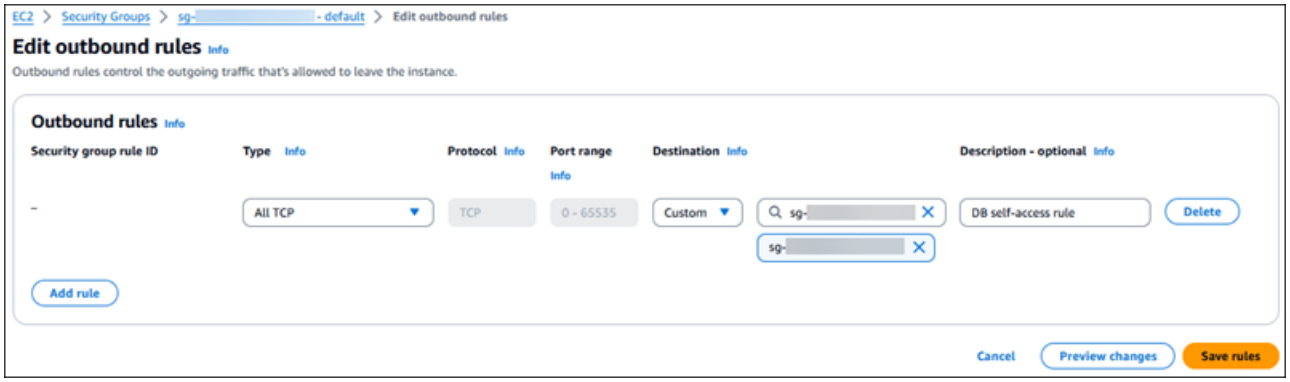


5. Dans l'onglet Règles sortantes :

   a. Choisissez Edit outbound rules (Modifier les règles sortantes).

   b. Ajoutez une nouvelle règle sortante pour le cluster ou l'instance de base de données source :

      - Port de base de données : `All traffic` (comprend les ports `0-65535`)
      - ID du groupe de sécurité : `sg-056a84f1712b77926` dans cet exemple



6. Connectez-vous à la AWS Management Console et ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
7. Accédez à la page Network ACLs (ACL réseau).
8. Ajoutez la configuration ACL réseau par défaut comme indiqué dans ACL réseau par défaut.

Préparation de la base de données de destination

Suivez les procédures décrites dans Création de tables Aurora PostgreSQL Limitless Database pour créer les tables de destination dans le groupe de partitions de base de données.

Vos tables de destination doivent utiliser les mêmes schémas, noms de tables et clés primaires que celles des tables sources.

Création des informations d'identification de base de données

Vous devez créer des utilisateurs de base de données dans les bases de données source et de destination, et leur accorder les privilèges nécessaires. Pour plus d'informations, consultez CREATE USER et GRANT dans la documentation de PostgreSQL.

Création des informations d'identification de base de données source

L'utilisateur de la base de données source est spécifié dans la commande permettant de lancer le chargement. Cet utilisateur doit disposer des privilèges nécessaires pour effectuer la réplication à partir de la base de données source.

1. Utilisez l'utilisateur principal de base de données (ou un autre utilisateur disposant du rôle `rds_superuser`) pour créer un utilisateur de base de données source bénéficiant de privilèges `LOGIN`.

```
CREATE USER source_db_username WITH PASSWORD 'source_db_user_password';
```

2. Accordez le rôle `rds_superuser` à l'utilisateur de la base de données source.

```
GRANT rds_superuser to source_db_username;
```

3. Si vous utilisez le mode `full_load_and_cdc`, accordez le rôle `rds_replication` à l'utilisateur de la base de données source. Le rôle `rds_replication` accorde les autorisations permettant de gérer des emplacements logiques et de diffuser les données à l'aide d'emplacements logiques.

```
GRANT rds_replication to source_db_username;
```

Création des informations d'identification de base de données de destination

L'utilisateur de la base de données de destination doit disposer de l'autorisation d'écriture sur les tables de destination du groupe de partitions de base de données.

1. Utilisez l'utilisateur principal de base de données (ou un autre utilisateur disposant du rôle `rds_superuser`) pour créer un utilisateur de base de données de destination bénéficiant de privilèges `LOGIN`.

```
CREATE USER destination_db_username WITH PASSWORD 'destination_db_user_password';
```

2. Accordez le rôle `rds_superuser` à l'utilisateur de la base de données de destination.

```
GRANT rds_superuser to destination_db_username;
```

Configuration de l’authentification de la base de données et de l’accès aux ressources à l’aide d’un script

Le script de configuration crée un rôle géré par le client AWS KMS key, un rôle Gestion des identités et des accès AWS (IAM) et deux secrets. AWS Secrets Manager

Pour utiliser le script de configuration, procédez comme suit :

1. Assurez-vous que vous l'avez AWS CLI installé et configuré avec vos Compte AWS informations d'identification.
2. Installez le processeur JSON `jq` en ligne de commande. Pour plus d’informations, consultez jqlang/jq.
3. Copiez le fichier data_loading_script.zip sur votre ordinateur et extrayez le fichier `data_load_aws_setup_script.sh`.
4. Modifiez le script pour remplacer les variables d’espace réservé par les valeurs appropriées pour les éléments suivants :
   - Votre Compte AWS
   - Le Région AWS
   - Informations d’identification de la base de données source
   - Informations d’identification de la base de données de destination
5. Ouvrez un terminal sur votre ordinateur et exécutez la commande suivante :

```
bash ./data_load_aws_setup_script.sh
```

Script de configuration pour l’utilitaire de chargement de données

Le texte du fichier `data_load_aws_setup_script.sh` est fourni ici à titre de référence.

```
#!/bin/bash
# Aurora Limitless data loading - AWS resources setup script #
# Set up the account credentials in advance. #
# Update the following script variables. #

###################################
#### Start of variable section ####

ACCOUNT_ID="12-digit_AWS_account_ID"
```

```
REGION="AWS_Region"
DATE=$(date +'%m%d%H%M%S')
RANDOM_SUFFIX="${DATE}"
SOURCE_SECRET_NAME="secret-source-${DATE}"
SOURCE_USERNAME="source_db_username"
SOURCE_PASSWORD="source_db_password"
DESTINATION_SECRET_NAME="secret-destination-${DATE}"
DESTINATION_USERNAME="destination_db_username"
DESTINATION_PASSWORD="destination_db_password"
DATA_LOAD_IAM_ROLE_NAME="aurora-data-loader-${RANDOM_SUFFIX}"
TMP_WORK_DIR="./tmp_data_load_aws_resource_setup/"

#### End of variable section ####
################################

# Main logic start
echo "DATE - [${DATE}]"
echo "RANDOM_SUFFIX - [${RANDOM_SUFFIX}]"
echo 'START!'

mkdir -p $TMP_WORK_DIR

# Create the symmetric KMS key for encryption and decryption.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_key_response.txt"
aws kms create-key --region $REGION | tee $TMP_FILE_PATH
KMS_KEY_ARN=$(cat $TMP_FILE_PATH | jq -r '.KeyMetadata.Arn')
aws kms create-alias \
    --alias-name alias/"${DATA_LOAD_IAM_ROLE_NAME}-key" \
    --target-key-id $KMS_KEY_ARN \
    --region $REGION

# Create the source secret.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_source_secret_response.txt"
aws secretsmanager create-secret \
    --name $SOURCE_SECRET_NAME \
    --kms-key-id $KMS_KEY_ARN \
    --secret-string "{\"username\":\"$SOURCE_USERNAME\",\"password\":\"$SOURCE_PASSWORD
\"}" \
    --region $REGION \
    | tee $TMP_FILE_PATH
SOURCE_SECRET_ARN=$(cat $TMP_FILE_PATH | jq -r '.ARN')

# Create the destination secret.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_destination_secret_response.txt"
```

```
aws secretsmanager create-secret \
    --name $DESTINATION_SECRET_NAME \
    --kms-key-id $KMS_KEY_ARN \
    --secret-string "{\"username\":\"$DESTINATION_USERNAME\",\"password\":
\"$DESTINATION_PASSWORD\"}" \
    --region $REGION \
    | tee $TMP_FILE_PATH
DESTINATION_SECRET_ARN=$(cat $TMP_FILE_PATH | jq -r '.ARN')

# Create the RDS trust policy JSON file.
# Use only rds.amazonaws.com for RDS PROD use cases.
TRUST_POLICY_PATH="${TMP_WORK_DIR}rds_trust_policy.json"
echo '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}' > $TRUST_POLICY_PATH

# Create the IAM role.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_create_iam_role_response.txt"
aws iam create-role \
    --role-name $DATA_LOAD_IAM_ROLE_NAME \
    --assume-role-policy-document "file://${TRUST_POLICY_PATH}" \
    --tags Key=assumer,Value=aurora_limitless_table_data_load \
    --region $REGION \
    | tee $TMP_FILE_PATH
IAM_ROLE_ARN=$(cat $TMP_FILE_PATH | jq -r '.Role.Arn')

# Create the permission policy JSON file.
PERMISSION_POLICY_PATH="${TMP_WORK_DIR}data_load_permission_policy.json"
permission_json_policy=$(cat &lt;&lt;EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "Ec2Permission",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "$SOURCE_SECRET_ARN",
                "$DESTINATION_SECRET_ARN"
            ]
        },
        {
            "Sid": "KmsPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "$KMS_KEY_ARN"
        },
        {
            "Sid": "RdsPermissions",
```

```
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBClusters",
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
EOF
)
echo $permission_json_policy > $PERMISSION_POLICY_PATH

# Add the inline policy.
aws iam put-role-policy \
    --role-name $DATA_LOAD_IAM_ROLE_NAME \
    --policy-name aurora-limitless-data-load-policy \
    --policy-document "file://${PERMISSION_POLICY_PATH}" \
    --region $REGION

# Create the key policy JSON file.
KEY_POLICY_PATH="${TMP_WORK_DIR}data_load_key_policy.json"
key_json_policy=$(cat &lt;&lt;EOF
{
    "Id": "key-aurora-limitless-data-load-$RANDOM_SUFFIX",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "$IAM_ROLE_ARN"
            },
            "Action": [
                "kms:Decrypt",
```

```
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "*"
        }
    ]
}
EOF
)
echo $key_json_policy > $KEY_POLICY_PATH

# Add the key policy.
TMP_FILE_PATH="${TMP_WORK_DIR}tmp_put_key_policy_response.txt"
sleep 10 # sleep 10 sec for IAM role ready
aws kms put-key-policy \
    --key-id $KMS_KEY_ARN \
    --policy-name default \
    --policy "file://${KEY_POLICY_PATH}" \
    --region $REGION \
    | tee $TMP_FILE_PATH

echo 'DONE!'

echo "ACCOUNT_ID : [${ACCOUNT_ID}]"
echo "REGION : [${REGION}]"
echo "RANDOM_SUFFIX : [${RANDOM_SUFFIX}]"
echo "IAM_ROLE_ARN : [${IAM_ROLE_ARN}]"
echo "SOURCE_SECRET_ARN : [${SOURCE_SECRET_ARN}]"
echo "DESTINATION_SECRET_ARN : [${DESTINATION_SECRET_ARN}]"

# Example of a successful run:
# ACCOUNT_ID : [012345678912]
# REGION : [ap-northeast-1]
# RANDOM_SUFFIX : [0305000703]
# IAM_ROLE_ARN : [arn:aws:iam::012345678912:role/aurora-data-loader-0305000703]
# SOURCE_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:012345678912:secret:secret-source-0305000703-yQDtow]
# DESTINATION_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:012345678912:secret:secret-destination-0305000703-5d5Jy8]

# If you want to manually clean up failed resource,
# please remove them in the following order:
# 1. IAM role.
```

```
 # aws iam delete-role-policy --role-name Test-Role --policy-name ExamplePolicy --
region us-east-1
# aws iam delete-role --role-name Test-Role --region us-east-1
# 2. Source and destination secrets.
# aws secretsmanager delete-secret --secret-id MyTestSecret --force-delete-without-
recovery --region us-east-1
# 3. KDM key.
# aws kms schedule-key-deletion --key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab --pending-window-in-days 7
 --region us-east-1
```

Sortie du script de configuration de l'utilitaire de chargement de données

L'exemple suivant présente la sortie générée par une exécution réussie du script.

```
% bash ./data_load_aws_setup_script.sh
DATE - [0305000703]
RANDOM_SUFFIX - [0305000703]
START!
{
    "KeyMetadata": {
        "AWSAccountId": "123456789012",
        "KeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
        "Arn": "arn:aws:kms:ap-
northeast-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab",
        "CreationDate": "2024-03-05T00:07:49.852000+00:00",
        "Enabled": true,
        "Description": "",
        "KeyUsage": "ENCRYPT_DECRYPT",
        "KeyState": "Enabled",
        "Origin": "AWS_KMS",
        "KeyManager": "CUSTOMER",
        "CustomerMasterKeySpec": "SYMMETRIC_DEFAULT",
        "KeySpec": "SYMMETRIC_DEFAULT",
        "EncryptionAlgorithms": [
            "SYMMETRIC_DEFAULT"
        ],
        "MultiRegion": false
    }
}
{
    "ARN": "arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
source-0305000703-yQDtow",
    "Name": "secret-source-0305000703",
```

```
    "VersionId": "a017bebe-a71b-4220-b923-6850c2599c26"
}
{
    "ARN": "arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
destination-0305000703-5d5Jy8",
    "Name": "secret-destination-0305000703",
    "VersionId": "32a1f989-6391-46b1-9182-f65d242f5eb6"
}
{
    "Role": {
        "Path": "/",
        "RoleName": "aurora-data-loader-0305000703",
        "RoleId": "AROAYPX63ITQOYORQSC6U",
        "Arn": "arn:aws:iam::123456789012:role/aurora-data-loader-0305000703",
        "CreateDate": "2024-03-05T00:07:54+00:00",
        "AssumeRolePolicyDocument": {
           "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "rds.amazonaws.com"
                        ]
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Tags": [
            {
                "Key": "assumer",
                "Value": "aurora_limitless_table_data_load"
            }
        ]
    }
}
DONE!
ACCOUNT_ID : [123456789012]
REGION : [ap-northeast-1]
RANDOM_SUFFIX : [0305000703]
IAM_ROLE_ARN : [arn:aws:iam::123456789012:role/aurora-data-loader-0305000703]
SOURCE_SECRET_ARN : [arn:aws:secretsmanager:ap-northeast-1:123456789012:secret:secret-
source-0305000703-yQDtow]
```

```
DESTINATION_SECRET_ARN : [arn:aws:secretsmanager:ap-
northeast-1:123456789012:secret:secret-destination-0305000703-5d5Jy8]
```

Nettoyage des ressources en échec

Si vous souhaitez nettoyer manuellement les ressources en échec, supprimez-les dans l'ordre suivant :

1. Rôle IAM, par exemple :

```
aws iam delete-role-policy \
--role-name Test-Role \
--policy-name ExamplePolicy

aws iam delete-role \
--role-name Test-Role
```

2. Secrets source et destination, par exemple :

```
aws secretsmanager delete-secret \
--secret-id MyTestSecret \
--force-delete-without-recovery
```

3. Clé KMS, par exemple :

```
aws kms schedule-key-deletion \
--key-id arn:aws:kms:us-west-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab
 \
--pending-window-in-days 7
```

Vous pouvez ensuite relancer le script.

Configuration manuelle de l’authentification de base de données et de l’accès aux ressources

Le processus manuel de configuration de l’authentification de la base de données et de l’accès aux ressources comprend les étapes suivantes :

1. Création du système géré par le client AWS KMS key
2. Ajout des stratégies d’autorisation de rôle IAM
3. Création des secrets de base de données
4. Création d’un rôle IAM
5. Mettre à jour le système géré par le client AWS KMS key

Ce processus est facultatif et exécute les mêmes tâches que dansConfiguration de l’authentification de la base de données et de l’accès aux ressources à l’aide d’un script. Nous vous recommandons d’utiliser le script.

Création du système géré par le client AWS KMS key

Suivez les procédures décrites dans Création de clés de chiffrement symétriques pour créer une clé KMS gérée par le client. Vous pouvez utiliser une clé existante si elle répond aux exigences suivantes :

Pour créer une clé KMS gérée par le client

1. Connectez-vous à la AWS KMS console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/km.
2. Accédez à la page Clés gérées par le client.
3. Choisissez Create key.
4. Sur la page Configurer la clé :
    a. Pour Type de clé, choisissez Symétrique.
    b. Pour Utilisation de la clé, choisissez Chiffrer et déchiffrer.
    c. Choisissez Suivant.
5. Sur la page Ajouter des étiquettes, entrez un Alias tel que **`limitless`**, puis choisissez Suivant.
6. Sur la page Définir les autorisations d’administration de clé, assurez-vous que la case Autoriser les administrateurs clés à supprimer cette clé est cochée, puis choisissez Suivant.

7. Sur la page Définir des autorisations d'utilisation de clé, choisissez Suivant.
8. Dans la page Vérification, choisissez Terminer.

   La stratégie de clé doit être mise à jour ultérieurement.

Tapez l'Amazon Resource Name (ARN) de la clé KMS qui doit être utilisée dans Ajout des stratégies d'autorisation de rôle IAM.

Pour plus d'informations sur l'utilisation de AWS CLI pour créer la clé KMS gérée par le client, voir create-key et create-alias.

Création des secrets de base de données

Pour permettre à l'utilitaire de chargement de données d'accéder aux tables de base de données source et de destination, vous devez créer deux secrets AWS Secrets Manager : un pour la base de données source et un pour la base de données de destination. Ces secrets stockent les noms d'utilisateur et les mots de passe permettant d'accéder aux bases de données source et de destination.

Suivez les procédures décrites dans Créer un secret AWS Secrets Manager pour créer les secrets de la paire clé-valeur.

Pour créer des secrets de base de données

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez Store a new secret (Stocker un nouveau secret).
3. Sur la page Choisir un type de secret :

   a. Pour Type de secret, choisissez Autre type de secret.
   b. Pour les Paires clé/valeur, choisissez l'onglet Texte brut.
   c. Entrez le code JSON suivant, où *sourcedbreader* et *sourcedbpassword* sont les informations d'identification de l'utilisateur de la base de données source Création des informations d'identification de base de données source.

```
{
    "username":"sourcedbreader",
    "password":"sourcedbpassword"
}
```

   d. Pour Clé de chiffrement, choisissez la clé KMS que vous avez créée dans Création du système géré par le client AWS KMS key, par exemple `limitless`.
   e. Choisissez Suivant.
4. Sur la page Configurer le secret, indiquez un Nom de secret, tel que **`source_DB_secret`** puis choisissez Suivant.
5. Sur la page Configurer la rotation - facultatif, choisissez Suivant.
6. Sur la page Review (Vérification), choisissez Store (Stocker).
7. Répétez la procédure pour le secret de base de données de destination :
   a. Entrez le code JSON suivant, où *`destinationdbwriter`* et *`destinationdbpassword`* sont les informations d'identification de l'utilisateur de la base de données de destination depuis Création des informations d'identification de base de données de destination.

```
{
    "username":"destinationdbwriter",
    "password":"destinationdbpassword"
}
```

   b. Saisissez un Nom de secret, par exemple **`destination_DB_secret`**.

Enregistrez ARNs les secrets à utiliserAjout des stratégies d'autorisation de rôle IAM.

Création d'un rôle IAM

Le chargement des données nécessite que vous donniez accès aux AWS ressources. Pour fournir un accès, vous créez le rôle IAM `aurora-data-loader` en suivant les procédures décrites dans Création d'un rôle pour la délégation d'autorisations à un utilisateur IAM.

Pour créer le rôle IAM

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Accédez à la page Rôles.
3. Choisissez Créer un rôle.
4. Sur la page Sélectionner une entité de confiance :
   a. Pour Type d'entité de confiance, choisissez Stratégie d'approbation personnalisée.

b. Entrez le code JSON suivant pour la stratégie d'approbation personnalisée :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "rds.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

c. Choisissez Suivant.

5. Sur la page Add permissions (Ajouter des autorisations), sélectionnez Next (Suivant).
6. Sur la page Nommer, vérifier et créer :

a. Pour Nom du rôle, entrez **`aurora-data-loader`** ou un autre nom de votre choix.

b. Choisissez Ajouter une balise et entrez la balise suivante :

- Clé : **`assumer`**
- Value (Valeur) : **`aurora_limitless_table_data_load`**

> **Important**
>
> Aurora PostgreSQL Limitless Database peut uniquement assumer un rôle IAM présentant cette balise.

c. Choisissez Créer un rôle.

Mettre à jour le système géré par le client AWS KMS key

Suivez les procédures décrites dans Modification d’une stratégie de clé pour ajouter le rôle IAM `aurora-data-loader` à la stratégie clé par défaut.

Pour ajouter le rôle IAM à la stratégie de clé

1. Connectez-vous à la AWS KMS console AWS Management Console et ouvrez-la à l'adresse https://console.aws.amazon.com/km.
2. Accédez à la page Clés gérées par le client.
3. Choisissez la clé KMS que vous avez créée dans Création du système géré par le client AWS KMS key, par exemple`limitless`.
4. Dans l’onglet Stratégie de clé, pour Utilisateurs de clés, sélectionnez Ajouter.
5. Dans la fenêtre Ajouter des utilisateurs clés, sélectionnez le nom du rôle IAM dans lequel vous avez crééCréation d’un rôle IAM, par exemple aurora-data-loader.
6. Choisissez Ajouter.

Ajout des stratégies d’autorisation de rôle IAM

Vous devez ajouter des stratégies d’autorisation au rôle IAM que vous avez créé. Cela permet à l’utilitaire de chargement de données Aurora PostgreSQL Limitless Database d’accéder aux ressources AWS associées pour établir des connexions réseau et récupérer les secrets des informations d’identification de la base de données source et de destination.

Pour plus d’informations, consultez Modification d’un rôle.

Pour ajouter des stratégies d’autorisation

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Accédez à la page Rôles.
3. Choisissez le rôle IAM que vous avez créé dansCréation d’un rôle IAM, par exemple aurora-data-loader.
4. Dans l’onglet Autorisations, sélectionnez Ajouter des autorisations, puis Créer une stratégie en ligne dans Stratégies d’autorisation.
5. Sur la page Spécifier les autorisations, choisissez l’éditeur JSON.

6. Copiez et collez le modèle suivant dans l'éditeur JSON, en remplaçant les espaces réservés par les secrets de votre base de données et la ARNs clé KMS.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Ec2Permission",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DescribeNetworkInterfacePermissions",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeRegions",
                "ec2:DescribeVpcs",
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkAcls"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SecretsManagerPermissions",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret"
            ],
            "Resource": [
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:source_DB_secret-ABC123",
                "arn:aws:secretsmanager:us-
east-1:123456789012:secret:destination_DB_secret-456DEF"
            ]
```

```
        },        {
            "Sid": "KmsPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:123456789012:key/aa11bb22-
####-####-####-fedcba123456"
        },
        {
            "Sid": "RdsPermissions",
            "Effect": "Allow",
            "Action": [
                "rds:DescribeDBClusters",
                "rds:DescribeDBInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

7. Vérifiez s'il y a des erreurs et corrigez-les.
8. Choisissez Suivant.
9. Sur la page Vérifier et créer, saisissez un Nom de stratégie, tel que **`data_loading_policy`**, puis choisissez Créer une stratégie.

Chargement de données à partir d’un cluster de bases de données Aurora PostgreSQL ou d’une instance de base de données RDS pour PostgreSQL

Après avoir terminé la configuration des ressources et de l’authentification, connectez-vous au point de terminaison du cluster et appelez la procédure stockée `rds_aurora.limitless_data_load_start` depuis une base de données sans limite, telle que `postgres_limitless`. La base de données sans limite est une base de données du groupe de partitions de base de données vers lequel vous souhaitez migrer les données.

Cette fonction se connecte de manière asynchrone en arrière-plan à la base de données source indiquée dans la commande, lit les données depuis la source avant de les charger sur les partitions. Pour de meilleures performances, les données sont chargées à l’aide de threads parallèles. La fonction récupère un instantané de table à un moment donné en exécutant une commande `SELECT` pour lire les données de la ou des tables spécifiées dans la commande.

Vous pouvez charger des données dans des tables partitionnées, de référence et standard.

Vous pouvez charger des données au niveau de la base de données, du schéma ou de la table lors des appels `rds_aurora.limitless_data_load_start`.

- Base de données : vous pouvez charger une base de données à la fois par appel, sans aucune limite concernant le nombre de schémas ou de tables qu’elle contient.
- Schéma : vous pouvez charger jusqu’à 15 schémas par appel, sans aucune limite concernant le nombre de tables dans chaque schéma.
- Table : vous pouvez charger jusqu’à 15 tables par appel.

> **Note**
>
> Cette fonctionnalité n’utilise pas les instantanés Amazon RDS ni l’isolation ponctuelle de la base de données. Pour assurer la cohérence entre les tables, nous vous recommandons de cloner la base de données source et d’utiliser cette base clonée comme source.

La procédure stockée utilise la syntaxe suivante :

```
CALL rds_aurora.limitless_data_load_start('source_type',
    'source_DB_cluster_or_instance_ID',
    'source_database_name',
    'streaming_mode',
    'data_loading_IAM_role_arn',
    'source_DB_secret_arn',
    'destination_DB_secret_arn',
    'ignore_primary_key_conflict_boolean_flag',
    'is_dry_run',
    (optional parameter) schemas/tables => ARRAY['name1', 'name2', ...]);
```

Les paramètres d'entrée sont les suivants :

- `source_type` : le type de source : `aurora_postgresql` or `rds_postgresql`.
- `source_DB_cluster_or_instance_ID` : l'identifiant du cluster de bases de données Aurora PostgreSQL source ou l'identifiant de l'instance de base de données RDS pour PostgreSQL.
- `source_database_name` : le nom de la base de données source, par exemple *`postgres`*
- `streaming_mode` : déterminer s'il faut inclure la capture de données modifiées (CDC) : `full_load` ou `full_load_and_cdc`
- `data_loading_IAM_role_arn` : l'Amazon Resource Name (ARN) du rôle IAM pour `aurora-data-loader`
- `source_DB_secret_arn` : l'ARN secret de la base de données source
- `destination_DB_secret_arn` : l'ARN secret de la base de données de destination
- `ignore_primary_key_conflict_boolean_flag` : déterminer s'il faut poursuivre l'exécution en cas de conflit de clé primaire :
  - S'il est défini sur `true`, le chargement des données ignore les nouvelles modifications apportées aux lignes présentant un conflit de clé primaire.
  - S'il est défini sur `false`, le chargement des données remplace les lignes existantes sur les tables de destination en cas de conflit de clé primaire.
- `is_dry_run` : déterminer s'il convient de tester la capacité de la tâche de chargement de données à se connecter aux bases de données source et de destination :
  - Si ce paramètre est défini sur `true`, les connexions sont testées sans chargement de données
  - Si ce paramètre est défini sur `false`, les données sont chargées

- (facultatif) `schemas` ou `tables` : un tableau de schémas ou de tables à charger. Vous pouvez spécifier chacune des valeurs suivantes :
  - Une liste de tables au format `tables => ARRAY['schema1.table1', 'schema1.table2', 'schema2.table1', ...]`
  - Une liste de schémas au format `schemas => ARRAY['schema1', 'schema2', ...]`

  Si ce paramètre n'est pas renseigné, la base de données source entière spécifiée est migrée.

Le paramètre de sortie correspond à l'ID de la tâche accompagné d'un message.

L'exemple suivant montre comment utiliser la procédure stockée `rds_aurora.limitless_data_load_start` pour charger des données à partir d'un cluster de bases de données Aurora PostgreSQL.

```
CALL rds_aurora.limitless_data_load_start('aurora_postgresql',
    'my-db-cluster',
    'postgres',
    'full_load_and_cdc',
    'arn:aws:iam::123456789012:role/aurora-data-loader-8f2c66',
    'arn:aws:secretsmanager:us-east-1:123456789012:secret:secret-source-8f2c66-EWrr0V',
    'arn:aws:secretsmanager:us-east-1:123456789012:secret:secret-destination-8f2c66-
d04fbD',
    'true',
    'false',
    tables => ARRAY['public.customer', 'public.order', 'public.orderdetails']);

INFO: limitless data load job id 1688761223647 is starting.
```

Surveillance du chargement des données

Aurora PostgreSQL Limitless Database propose plusieurs méthodes pour surveiller les tâches de chargement de données :

- Liste des tâches de chargement de données
- Affichage des détails des tâches de chargement de données à l'aide de l'ID de tâche
- Surveillance du groupe de journaux Amazon CloudWatch
- Surveillance des événements RDS

Liste des tâches de chargement de données

Vous pouvez vous connecter au point de terminaison du cluster et utiliser la vue `rds_aurora.limitless_data_load_jobs` pour répertorier les tâches de chargement de données.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_data_load_jobs LIMIT 6;

    job_id     |  status   | message |     source_db_identifier      | source_db_name |
 full_load_complete_time |
   progress_details                                                                 |
     start_time       |   last_updated_time    |  streaming_mode   | source_engine_type
 | ignore_primary_key_conflict | is_dryrun
---------------+-----------+---------+-------------------------------+---------------
+-------------------------
+-------------------------------------------------------------------------------------------------------------------------
+------------------------+-----------------------+------------------
+--------------------+-----------------------------+-----------
 1725697520693 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:48:15+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "9
 of 9 tables loaded", "COMPLETED_AT": "2024/09/07 08:48:15+00", "RECORDS_MIGRATED":
 600003}} | 2024-09-07 08:47:13+00 | 2024-09-07 08:48:15+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725696114225 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:24:20+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "3
 of 3 tables loaded", "COMPLETED_AT": "2024/09/07 08:24:20+00", "RECORDS_MIGRATED":
 200001}} | 2024-09-07 08:23:56+00 | 2024-09-07 08:24:20+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725696067630 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 08:23:45+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "6
 of 6 tables loaded", "COMPLETED_AT": "2024/09/07 08:23:45+00", "RECORDS_MIGRATED":
```

```
 400002}} | 2024-09-07 08:23:10+00 | 2024-09-07 08:23:45+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725694221753 | CANCELED  |         | persistent-kdm-auto-source-01 | postgres
    |                         | {}

           | 2024-09-07 07:31:18+00 | 2024-09-07 07:51:49+00 | full_load_and_cdc |
 aurora_postgresql  | t                           | f
 1725691698210 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 07:10:51+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "1
 of 1 tables loaded", "COMPLETED_AT": "2024/09/07 07:10:51+00", "RECORDS_MIGRATED":
 100000}} | 2024-09-07 07:10:42+00 | 2024-09-07 07:10:52+00 | full_load         |
 aurora_postgresql  | t                           | f
 1725691695049 | COMPLETED |         | persistent-kdm-auto-source-01 | postgres
    | 2024-09-07 07:10:48+00  | {"FULL_LOAD": {"STATUS": "COMPLETED", "DETAILS": "1
 of 1 tables loaded", "COMPLETED_AT": "2024/09/07 07:10:48+00", "RECORDS_MIGRATED":
 100000}} | 2024-09-07 07:10:41+00 | 2024-09-07 07:10:48+00 | full_load         |
 aurora_postgresql  | t                           | f
(6 rows)
```

Les enregistrements de tâches sont supprimés après 90 jours.

Affichage des détails des tâches de chargement de données à l'aide de l'ID de tâche

Si vous disposez de l'ID d'une tâche, vous pouvez vous connecter au point de terminaison du cluster et utiliser la vue `rds_aurora.limitless_data_load_job_details` pour consulter les détails de cette tâche de chargement de données, notamment le nom de la table, l'état de la tâche et le nombre de lignes chargées. L'ID de la tâche est disponible dans la réponse des fonctions qui lancent le chargement des données, ou dans la vue `rds_aurora.limitless_data_load_jobs`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_data_load_job_details WHERE
 job_id='1725696114225';

job_id        | destination_table_name | destination_schema_name | start_time
      | status    | full_load_rows | full_load_total_rows | full_load_complete_time |
 cdc_insert | cdc_update | cdc_delete
--------------+------------------------+-------------------------
+-------------------------+-----------+----------------+---------------------
+-------------------------+------------+------------+------------
1725696114225 | standard_1             | public                  | 2024-09-07
 08:23:57+00 | COMPLETED | 100000         | 100000              | 2024-09-07
 08:24:08+00  | 0          | 0          | 0
```

```
1725696114225 | standard_2             | public                  | 2024-09-07
 08:24:08+00 | COMPLETED | 100000        | 100000               | 2024-09-07
 08:24:17+00  | 0          | 0          | 0
1725696114225 | standard_3             | public                  | 2024-09-07
 08:24:18+00 | COMPLETED | 1             | 1                    | 2024-09-07
 08:24:20+00  | 0          | 0          | 0
1725696114225 | standard_4             | public                  | 2024-09-07
 08:23:58+00 | PENDING   | 0             | 0                    |
    | 0          | 0          | 0
(4 rows)
```

Les enregistrements de tâches sont supprimés après 90 jours.

Surveillance du groupe de journaux Amazon CloudWatch

Une fois que l'état de la tâche de chargement des données est passé à `RUNNING`, vous pouvez vérifier la progression de l'exécution à l'aide d'Amazon CloudWatch Logs.

Pour surveiller les flux de journaux CloudWatch

Connectez-vous à AWS Management Console et ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.

1. Accédez à Journaux, puis à Groupe de journaux.
2. Choisissez le groupe de journaux /aws/rds/aurora-limitless-database.
3. Recherchez le flux de journaux de votre tâche de chargement de données par job_id.

   Le flux du journaux présente le modèle Data-Load-Job-***job_id***.
4. Choisissez le flux de journaux pour afficher les événements du journal.

Chaque flux de journal affiche des événements contenant l'état de la tâche et le nombre de lignes chargées dans les tables de destination Aurora PostgreSQL Limitless Database. Si une tâche de chargement de données échoue, un journal d'erreurs est également créé, précisant l'état de l'échec ainsi que sa cause.

Les enregistrements de tâches sont supprimés après 90 jours.

Surveillance des événements RDS

La tâche de chargement de données publie également des événements RDS, notamment lorsqu'une tâche réussit, échoue ou est annulée. Vous pouvez afficher les événements depuis la base de données de destination.

Pour plus d'informations, consultez Événements de groupe de partitions de base de données.

Annulation du chargement de données

Pour annuler une tâche de chargement de données, appelez la procédure stockée `rds_aurora.limitless_data_load_cancel`, en indiquant l'ID de tâche comme paramètre d'entrée. Cette procédure stockée doit être appelée à partir de la base de données du groupe de partitions de base de données dans laquelle la tâche de chargement de données a été démarrée. Exemples :

```
CALL rds_aurora.limitless_data_load_cancel(12345);

INFO: limitless data load job with id 12345 is canceling without rollback.
```

Vous ne pouvez pas annuler une tâche de chargement de données qui n'existe pas ou qui n'est pas exécutée dans le même groupe de partitions de base de données.

L'utilitaire de chargement de données Aurora PostgreSQL Limitless Database laisse les données chargées dans les tables de destination sans restauration, comme le montre la réponse. Si vous ne souhaitez pas conserver les données chargées, nous vous recommandons de tronquer les tables de destination.

Interrogation d'Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database est compatible avec la syntaxe de requêtes PostgreSQL. Vous pouvez interroger votre Limitless Database à l'aide de `psql` ou de tout autre utilitaire de connexion compatible avec PostgreSQL. Pour exécuter des requêtes, vous vous connectez au point de terminaison sans limite comme indiqué dans Connexion de votre cluster de bases de données dans Aurora PostgreSQL Limitless Database.

Toutes les requêtes `SELECT` PostgreSQL sont prises en charge dans Aurora PostgreSQL Limitless Database. Toutefois, les requêtes sont effectuées sur deux couches :

1. Routeur auquel le client envoie la requête
2. Partitions où résident les données

Les performances varient selon la façon dont la base de données est interrogée, celle-ci devant être en mesure de gérer efficacement un grand nombre de requêtes simultanées sur différentes partitions. Les requêtes sont d'abord analysées dans la couche de transaction distribuée (routeur). Une phase d'analyse précède la planification de l'exécution de la requête et permet de déterminer où se trouvent les différentes relations concernées. Si toutes les relations concernées sont des tables partitionnées dont la clé de partition est filtrée sur la même partition, ou bien des tables de référence, alors la planification des requêtes est ignorée au niveau de la couche routeur et entièrement déléguée à la partition, qui se charge de la planification et de l'exécution. Ce processus réduit le nombre d'allers-retours entre les différents nœuds (routeur et partition), ce qui se traduit par une amélioration des performances dans la plupart des cas. Pour plus d'informations, consultez Requêtes à partition unique dans Aurora PostgreSQL Limitless Database.

> **Note**
>
> Dans certains cas particuliers, comme les produits cartésiens (jointures croisées), la requête est plus efficace lorsqu'elle extrait les données séparément de la partition.

Pour plus d'informations sur les plans d'exécution des requêtes, consultez EXPLAIN dans Référence Aurora PostgreSQL Limitless Database. Pour obtenir des informations générales sur les requêtes, consultez Requêtes dans la documentation PostgreSQL.

Rubriques

Requêtes à partition unique dans Aurora PostgreSQL Limitless Database

Une requête à partition unique est une requête qui peut être exécutée directement sur une partition, tout en conservant les propriétés ACID du SQL. Lorsque le planificateur de requêtes du routeur identifie une requête de ce type, il transmet la requête SQL complète à la partition concernée.

Cette optimisation réduit le nombre d'allers et retours réseau entre le routeur et la partition, améliorant ainsi les performances. Actuellement, cette optimisation est effectuée pour les requêtes `INSERT`, `SELECT`, `UPDATE` et `DELETE`.

Rubriques

Exemples de requêtes à partition unique

Dans les exemples suivants, nous utiliserons la table partitionnée `customers`, avec la clé de partition `customer_id` et la table de référence `zipcodes`.

SELECT

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) SELECT * FROM customers WHERE
 customer_id = 100;

                       QUERY PLAN
-------------------------------------------------------------
 Foreign Scan
   Output: customer_id, other_id, customer_name, balance
   Remote SQL:  SELECT customer_id,
     other_id,
```

```
     customer_name,
     balance
    FROM public.customers
   WHERE (customer_id = 100)
 Single Shard Optimized
(9 rows)
```

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) SELECT * FROM orders
    LEFT JOIN zipcodes ON orders.zipcode_id = zipcodes.zipcode_id
    WHERE customer_id = 11;

                                              QUERY PLAN

-----------------------------------------------------------------------------------------------
 Foreign Scan
   Output: customer_id, order_id, zipcode_id, customer_name, balance,
 zipcodes.zipcode_id, zipcodes.city
   Remote SQL:  SELECT orders.customer_id,
     orders.order_id,
     orders.zipcode_id,
     orders.customer_name,
     orders.balance,
     zipcodes.zipcode_id,
     zipcodes.city
    FROM (public.orders
      LEFT JOIN public.zipcodes ON ((orders.zipcode_id = zipcodes.zipcode_id)))
   WHERE (orders.customer_id = 11)
 Single Shard Optimized
(13 rows)
```

INSERT

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) INSERT INTO customers
    (customer_id, other_id, customer_name, balance)
    VALUES (1, 10, 'saikiran', 1000);

                      QUERY PLAN
------------------------------------------------------
 Insert on public.customers
   ->  Result
         Output: 1, 10, 'saikiran'::text, '1000'::real
 Single Shard Optimized
```

```
(4 rows)
```

UPDATE

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) UPDATE orders SET balance = balance +
 100
    WHERE customer_id = 100;

                                        QUERY PLAN

-----------------------------------------------------------------------------------------------
 Update on public.orders
   Foreign Update on public.orders_fs00002 orders_1
   ->  Foreign Update
         Remote SQL:  UPDATE public.orders SET balance = (balance + (100)::double
 precision)
   WHERE (customer_id = 100)
 Single Shard Optimized
(6 rows)
```

DELETE

```
postgres_limitless=> EXPLAIN (VERBOSE, COSTS OFF) DELETE FROM orders
    WHERE customer_id = 100 and balance = 0;

                             QUERY PLAN
---------------------------------------------------------------------
 Delete on public.orders
   Foreign Delete on public.orders_fs00002 orders_1
   ->  Foreign Delete
         Remote SQL:  DELETE FROM public.orders
   WHERE ((customer_id = 100) AND (balance = (0)::double precision))
 Single Shard Optimized
(6 rows)
```

Restrictions appliquées aux requêtes à partition unique

Les requêtes à partition unique sont soumises aux restrictions suivantes :

Fonctions

Si une requête à partition unique contient une fonction, elle n'est éligible à l'optimisation à partition unique que si l'une des conditions suivantes est remplie :

- La fonction est immuable. Pour plus d'informations, consultez Volatilité des fonctions.
- La fonction est mutable, mais elle est enregistrée dans la vue `rds_aurora.limitless_distributed_functions`. Pour plus d'informations, consultez Répartition des fonctions.

Vues

Si une requête contient une ou plusieurs vues, l'optimisation à partition unique est désactivée si l'une des conditions suivantes est remplie :

- N'importe quelle vue possède l'attribut `security_barrier`.
- Les objets utilisés dans la requête nécessitent plusieurs privilèges utilisateur. Par exemple, une requête contient deux vues, lesquelles sont exécutées par deux utilisateurs différents.

```
CREATE VIEW v1 AS SELECT customer_name FROM customers c WHERE c.customer_id =  1;
CREATE VIEW v2 WITH (security_barrier) AS SELECT customer_name FROM customers c
 WHERE c.customer_id =  1;

postgres_limitless=> EXPLAIN VERBOSE SELECT * FROM v1;
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
   Output: customer_name
   Remote Plans from Shard postgres_s3:
         Seq Scan on public.customers_ts00001 c  (cost=0.00..24.12 rows=6 width=32)
           Output: c.customer_name
           Filter: (c.customer_id = 1)
         Query Identifier: -6005737533846718506
   Remote SQL:  SELECT customer_name
    FROM ( SELECT c.customer_name
             FROM public.customers c
            WHERE (c.customer_id = 1)) v1
 Query Identifier: -5754424854414896228
(12 rows)


postgres_limitless=> EXPLAIN VERBOSE SELECT * FROM v2;
                                        QUERY PLAN
```

```
-------------------------------------------------------------------------------------------
 Foreign Scan on public.customers_fs00001 c  (cost=100.00..128.41 rows=7 width=32)
   Output: c.customer_name
   Remote Plans from Shard postgres_s3:
         Seq Scan on public.customers_ts00001 customers  (cost=0.00..24.12 rows=6
 width=32)
           Output: customers.customer_name
           Filter: (customers.customer_id = 1)
         Query Identifier: 4136563775490008117
   Remote SQL: SELECT customer_name FROM public.customers WHERE ((customer_id = 1))
 Query Identifier: 5056054318010163757
(9 rows)
```

Instructions PREPARE et EXECUTE

Aurora PostgreSQL Limitless Database prend en charge l'optimisation à partition unique pour les instructions préparées `SELECT`, `UPDATE` et `DELETE`.

Cependant, lorsque des instructions préparées sont utilisées pour `PREPARE` et `EXECUTE` alors que `plan_cache_mode` est défini sur `'force_generic_plan'`, le planificateur de requêtes rejette l'optimisation à partition unique pour cette requête.

PL/pgSQL

Les requêtes contenant des variables PL/pgSQL sont exécutées sous forme d'instructions préparées implicites. Si une requête contient des variables PL/pgSQL, le planificateur de requêtes rejette l'optimisation à partition unique.

L'optimisation est prise en charge dans le bloc PL/pgSQL si l'instruction ne contient aucune variable PL/pgSQL.

Jointures entièrement qualifiées (explicites)

L'optimisation à partition unique est basée sur l'élimination des partitions. L'optimiseur PostgreSQL élimine les partitions en fonction de conditions constantes. Si Aurora PostgreSQL Limitless Database constate que toutes les partitions et tables restantes résident sur la même partition, elle indique que la requête est éligible à l'optimisation à partition unique. Toutes les conditions du filtre doivent être explicites pour que l'élimination des partitions fonctionne. Aurora PostgreSQL Limitless Database ne peut pas éliminer de partitions en l'absence d'un ou de plusieurs prédicats de jointure ou de filtrage portant sur les clés de partition de chaque table partitionnée de l'instruction.

Supposons que nous ayons partitionné les tables `customers`, `orders` et `order_details` en fonction de la colonne `customer_id`. Dans ce schéma, l'application essaie de conserver toutes les données d'un client sur une seule partition.

Considérons la requête suivante :

```
SELECT * FROM
    customers c, orders o, order_details od
WHERE c.customer_id = o.customer_id
    AND od.order_id = o.order_id
    AND c.customer_id = 1;
```

Cette requête récupère toutes les données d'un client (`c.customer_id = 1`). Les données de ce client résident sur une seule partition, mais Aurora PostgreSQL Limitless Database ne considère pas cette requête comme une requête à partition unique. Le processus d'optimisation de la requête est le suivant :

1. L'optimiseur peut éliminer les partitions pour `customers` et `orders` selon les conditions suivantes :

   ```
   c.customer_id = 1
   c.customer_id = o.customer_id
   o.customer_id =  1 (transitive implicit condition)
   ```

2. L'optimiseur ne peut éliminer aucune partition pour `order_details`, car aucune condition constante n'est appliquée à la table.
3. L'optimiseur détermine qu'il a lu toutes les partitions depuis `order_details`. Par conséquent, la requête ne peut pas être qualifiée pour l'optimisation à partition unique.

Pour en faire une requête à partition unique, nous ajoutons la condition de jointure explicite suivante :

```
o.customer_id = od.customer_id
```

La requête modifiée se présente comme suit :

```
SELECT * FROM
    customers c, orders o,  order_details od
WHERE c.customer_id = o.customer_id
     AND o.customer_id = od.customer_id
     AND od. order_id = o. order_id
```

```
 AND c.customer_id =  1;
```

L'optimiseur peut désormais éliminer les partitions pour `order_details`. La nouvelle requête devient une requête à partition unique et est éligible à l'optimisation.

Configuration d'une clé de partition active

Cette fonctionnalité vous permet de définir une clé de partition unique lors de l'interrogation de la base de données, ce qui permet d'ajouter la clé de partition à toutes les requêtes `SELECT` et DML en tant que prédicat constant. Cette fonctionnalité est utile si vous avez migré vers Aurora PostgreSQL Limitless Database et que vous avez dénormalisé le schéma en ajoutant des clés de partition aux tables.

Vous pouvez ajouter automatiquement un prédicat de clé de partition à la logique SQL existante, sans modifier la sémantique des requêtes. L'ajout d'un prédicat de clé de partition actif n'est possible que sur les tables compatibles.

La fonctionnalité de clé de partition active utilise la variable `rds_aurora.limitless_active_shard_key`, dont la syntaxe est la suivante :

```
SET [session | local] rds_aurora.limitless_active_shard_key = '{"col1_value",
 "col2_value", ...}';
```

Quelques considérations concernant les clés de partition actives et les clés étrangères :

- Une table partitionnée peut être soumise à une contrainte de clé étrangère lorsque les tables parente et enfant sont colocalisées et que la clé étrangère constitue un sur-ensemble de la clé de partition.
- Une table partitionnée peut être soumise à une contrainte de clé étrangère vers une table de référence.
- Une table de référence peut être soumise à une contrainte de clé étrangère vers une autre table de référence.

Prenons l'exemple d'une table partitionnée `customers` sur la colonne `customer_id`.

```
BEGIN;
SET local rds_aurora.limitless_create_table_mode='sharded';
SET local rds_aurora.limitless_create_table_shard_key='{"customer_id"}';
CREATE TABLE customers(customer_id int PRIMARY KEY, name text , email text);
```

```
COMMIT;
```

Lorsqu'une clé de partition active est définie, les requêtes subissent les transformations suivantes.

SELECT

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
SELECT * FROM customers;

-- This statement is changed to:
SELECT * FROM customers WHERE customer_id = '123'::int;
```

INSERT

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
INSERT INTO customers(name, email) VALUES('Alex', 'alex@example.com');

-- This statement is changed to:
INSERT INTO customers(customer_id, name, email) VALUES('123'::int, 'Alex',
 'alex@example.com');
```

UPDATE

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
UPDATE customers SET email = 'alex_new_email@example.com';

-- This statement is changed to:
UPDATE customers SET email = 'alex_new_email@example.com' WHERE customer_id =
 '123'::int;
```

DELETE

```
SET rds_aurora.limitless_active_shard_key = '{"123"}';
DELETE FROM customers;

-- This statement is changed to:
DELETE FROM customers WHERE customer_id = '123'::int;
```

Jointures

Lorsque vous effectuez des opérations de jointure sur des tables ayant une clé de partition active, le prédicat de clé de partition est automatiquement ajouté à toutes les tables participant à la

jointure. Cet ajout automatique du prédicat de clé de partition se produit uniquement lorsque toutes les tables de la requête appartiennent au même groupe de collocation. Lorsque des tables issues de plusieurs groupes de colocalisation sont impliquées dans la requête, le système renvoie une erreur.

Supposons en outre l'existence des tables `orders` et `order_details`, qui sont colocalisées avec la table `customers`.

```
SET local rds_aurora.limitless_create_table_mode='sharded';
SET local rds_aurora.limitless_create_table_collocate_with='customers';
SET local rds_aurora.limitless_create_table_shard_key='{"customer_id"}';
CREATE TABLE orders (id int , customer_id int, total_amount int, date date);
CREATE TABLE order_details (id int , order_id int, customer_id int, product_name
 VARCHAR(100), price int);
COMMIT;
```

Récupérez les 10 dernières factures de commande du client portant l'ID 10.

```
SET rds_aurora.limitless_active_shard_key = '{"10"}';
SELECT * FROM customers, orders, order_details WHERE
    orders.customer_id = customers.customer_id AND
    order_details.order_id = orders.order_id AND
    customers.customer_id = 10
    order by order_date limit 10;
```

Cette requête est transformée comme suit :

```
SELECT * FROM customers, orders, order_details WHERE
    orders.customer_id = customers.customer_id AND
    orders.order_id = order_details.order_id AND
    customers.customer_id = 10 AND
    order_details.customer_id = 10 AND
    orders.customer_id = 10 AND
    ORDER BY "order_date" LIMIT 10;
```

Tables compatibles avec les clés de partition actives

Le prédicat de clé de partition est ajouté uniquement aux tables compatibles avec la clé de partition active. Une table est considérée comme compatible si le nombre de colonnes de sa clé de partition correspond à celui indiqué dans la variable `rds_aurora.limitless_active_shard_key`. Si la requête concerne des tables

incompatibles avec la clé de partition active, le système génère une erreur au lieu de poursuivre la requête.

Exemples :

```
-- Compatible table
SET rds_aurora.limitless_active_shard_key = '{"10"}';

-- The following query works because the customers table is sharded on one column.
SELECT * FROM customers;

-- Incompatible table
SET rds_aurora.limitless_active_shard_key = '{"10","20"}';

-- The following query raises a error because the customers table isn't sharded on
 two columns.
 SELECT * FROM customers;
```

Requêtes distribuées dans Aurora PostgreSQL Limitless Database

Les requêtes distribuées s'exécutent sur un routeur et plusieurs partitions. La requête est reçue par l'un des routeurs. Le routeur crée et gère la transaction distribuée, qui est envoyée aux partitions concernées. Les partitions créent une transaction locale à partir du contexte fourni par le routeur, puis exécutent la requête.

Lorsque la transaction est validée, le routeur applique, si besoin, un protocole de validation en deux phases optimisé et un contrôle de concurrence multi-version (MVCC) basé sur le temps pour fournir une sémantique ACID dans un système de base de données distribué.

Le MVCC basé sur le temps enregistre l'heure de validation de chaque transaction et utilise l'heure de démarrage de la transaction pour générer l'heure de l'instantané des données. Pour déterminer si une transaction est validée (et donc visible) à partir de l'instantané du lecteur, la base de données compare son heure de validation à celle de l'instantané. Si son heure de validation est antérieure à celle de l'instantané du lecteur, la transaction est visible ; sinon, elle est invisible. Ce protocole garantit que les données visibles sur Aurora PostgreSQL Limitless Database sont toujours fortement cohérentes.

Suivi des requêtes distribuées dans les journaux PostgreSQL dans Aurora PostgreSQL Limitless Database

Le suivi des requêtes distribuées est un outil permettant de suivre et de corréler les requêtes dans les journaux PostgreSQL dans Aurora PostgreSQL Limitless Database. Dans Aurora PostgreSQL, l'ID de transaction sert à reconnaître chaque transaction. Toutefois, dans Aurora PostgreSQL Limitless Database, le même ID de transaction peut apparaître sur plusieurs routeurs. C'est pourquoi il est préférable d'utiliser l'ID de suivi dans Limitless Database.

Voici les principaux cas d'utilisation :

- Utilisez la fonction `rds_aurora.limitless_get_last_trace_id()` pour trouver l'ID de suivi unique de la dernière requête exécutée dans la session en cours. Effectuez ensuite une recherche dans le groupe de journaux du cluster de bases de données dans Amazon CloudWatch Logs à l'aide de cet ID de suivi pour trouver tous les journaux associés.

  Vous pouvez utiliser les paramètres `log_min_messages` et `log_min_error_statement` conjointement pour contrôler le volume des journaux affichés et enregistrer une instruction contenant l'ID de suivi.
- Utilisez le paramètre `log_min_duration_statement` pour définir la durée d'exécution au-delà de laquelle toutes les requêtes enregistrent leur temps d'exécution et leur ID de suivi. Cette durée d'exécution peut ensuite être recherchée dans le groupe de journaux du cluster de bases de données de CloudWatch Logs afin d'identifier les nœuds constituant des goulots d'étranglement et d'évaluer les efforts d'optimisation du planificateur.

  Le paramètre `log_min_duration_statement` active l'ID de suivi pour tous les nœuds, quels que soient les valeurs des paramètres `log_min_messages` et `log_min_error_statement`.

Rubriques

ID de suivi

Cette fonctionnalité repose sur un identifiant unique, désigné sous le nom d'ID de suivi. L'ID de suivi est une chaîne composée de 31 chiffres, ajoutée aux lignes de journal de type STATEMENT dans

les journaux PostgreSQL. Il sert d'identifiant unique permettant de corréler les entrées associées à l'exécution d'une requête donnée. Par exemple, `1126253375719408504000000000011` et `1126253375719408495000000000090`.

L'ID de suivi est composé des éléments suivants :

- Identifiant de transaction : les 20 premiers chiffres, identifiant de manière unique la transaction.
- Identifiant de commande : les 30 premiers chiffres, indiquant une requête individuelle au sein d'une transaction.

  Si plus de 4 294 967 294 requêtes sont exécutées dans un bloc de transaction explicite, l'identifiant de commande revient à 1. Dans ce cas, vous êtes averti par le message `LOG` suivant dans le journal PostgreSQL :

  ```
  wrapping around the tracing ID back to 1 after running 4294967294 (4.2 billion or
   2^32-2) queries inside of an explicit transaction block
  ```

- Identifiant du type de nœud : le dernier chiffre indiquant si le nœud fonctionne en tant que routeur coordinateur (`1`) ou en tant que nœud participant (`0`).

Les exemples suivants illustrent les composants de l'ID de suivi :

- `1126253375719408504000000000011`:
  - Identifiant de transaction : `11262533757194085`04
  - Identifiant de commande : `1126253375719408504000000000001` indique la première commande du bloc de transactions
  - Identifiant du type de nœud : `1` indique un routeur coordinateur
- `1126253375719408495000000000090`:
  - Identifiant de transaction : `11262533757194084`95
  - Identifiant de commande : `1126253375719408495000000000009` indique la neuvième commande du bloc de transactions
  - Identifiant du type de nœud : `0` indique un nœud participant

Utilisation du suivi des requêtes

Effectuez les tâches suivantes pour activer le suivi des requêtes :

1. Vérifiez que le suivi est activé.

   Pour effectuer la vérification, utilisez la commande suivante :

   ```
   SHOW rds_aurora.limitless_log_distributed_trace_id;
   ```

   Elle est activée par défaut (on). S'il n'est pas activé, définissez-le à l'aide de la commande suivante :

   ```
   SET rds_aurora.limitless_log_distributed_trace_id = on;
   ```

2. Contrôlez le volume de journaux affichés en configurant le niveau de gravité des journaux.

   Le volume des journaux est contrôlé par le paramètre `log_min_messages`. Le paramètre `log_min_error_statement` est utilisé pour afficher la ligne `STATEMENT` contenant l'ID de suivi. Les deux sont définis sur `ERROR` par défaut. Pour effectuer la vérification, utilisez les commandes suivantes :

   ```
   SHOW log_min_messages;
   SHOW log_min_error_statement;
   ```

   Pour mettre à jour le niveau de gravité et afficher la ligne `STATEMENT` correspondant à la session en cours, utilisez les commandes suivantes avec l'un de ces niveaux de gravité :

   ```
   SET log_min_messages = 'DEBUG5 | DEBUG4 | DEBUG3 | DEBUG2 | DEBUG1 | INFO | NOTICE
    | WARNING | ERROR | LOG | FATAL | PANIC';
   SET log_min_error_statement = 'DEBUG5 | DEBUG4 | DEBUG3 | DEBUG2 | DEBUG1 | INFO
    | NOTICE | WARNING | ERROR | LOG | FATAL | PANIC';
   ```

   Exemples :

   ```
   SET log_min_messages = 'WARNING';
   SET log_min_error_statement = 'WARNING';
   ```

3. Activez l'affichage de l'ID de suivi dans les journaux pour les exécutions dépassant une durée spécifique.

   Le paramètre `log_min_duration_statement` peut être modifié en fonction de la durée minimale d'exécution d'une requête, au-delà de laquelle celle-ci enregistre une ligne de journal

indiquant sa durée d'exécution ainsi que les identifiants de suivi dans le cluster de bases de données. Ce paramètre est défini sur `-1` par défaut, ce qui signifie qu'il est désactivé. Pour effectuer la vérification, utilisez la commande suivante :

```
SHOW log_min_duration_statement;
```

En le définissant sur `0`, la durée d'exécution et l'ID de suivi sont enregistrés dans les journaux pour chaque requête du cluster de bases de données. Vous pouvez le définir sur `0` pour la session actuelle en utilisant la commande suivante :

```
SET log_min_duration_statement = 0;
```

4. Obtenez l'ID de suivi.

   Après avoir exécuté une requête (même à l'intérieur d'un bloc de transaction explicite), appelez la fonction `rds_aurora.limitless_get_last_trace_id` pour obtenir l'ID de suivi de la dernière requête exécutée :

   ```
   SELECT * FROM rds_aurora.limitless_get_last_trace_id();
   ```

   Cette fonction renvoie l'identifiant de transaction et l'identifiant de commande. Elle ne renvoie pas l'identifiant du type de nœud.

   ```
   => SELECT * FROM customers;
    customer_id | fname | lname
   -------------+-------+-------
   (0 rows)

   => SELECT * FROM rds_aurora.limitless_get_last_trace_id();
    transaction_identifier |       command_identifier
   ------------------------+--------------------------------
    10104661421959001813   | 10104661421959001813000000001
   (1 row)
   ```

   La fonction renvoie une ligne vide pour les requêtes non distribuées, puisqu'aucun ID de suivi ne leur est attribué.

   ```
   => SET search_path = public;
   SET
   ```

```
=> SELECT * FROM rds_aurora.limitless_get_last_trace_id();
 transaction_identifier | command_identifier
------------------------+--------------------
                        |
(1 row)
```

> **Note**
>
> Pour les requêtes VACUUM et ANALYZE, l'instruction de durée n'est pas enregistrée avec l'ID de suivi. Par conséquent, `limitless_get_last_trace_id()` ne renvoie pas l'ID de suivi. Si une opération VACUUM ou ANALYZE s'exécute pendant une longue durée, vous pouvez utiliser la requête suivante pour obtenir l'ID de suivi correspondant à cette opération :
>
> ```
> SELECT * FROM rds_aurora.limitless_stat_activity
> WHERE distributed_tracing_id IS NOT NULL;
> ```
>
> Si le serveur s'arrête avant que vous n'ayez récupéré le dernier ID de suivi, vous devrez effectuer une recherche manuelle dans les journaux PostgreSQL afin d'identifier les ID de suivi figurant dans les entrées précédant la panne.

5. Recherchez l'ID de suivi dans les journaux du cluster de bases de données à l'aide de CloudWatch.

   Utilisez CloudWatch Insights pour interroger le groupe de journaux du cluster de bases de données, comme illustré dans les exemples suivants.

   - Requête permettant d'afficher un identifiant de transaction spécifique ainsi que toutes les commandes exécutées à l'intérieur de celle-ci :

     ```
     fields @timestamp, @message
     | filter @message like /10104661421959001813/
     | sort @timestamp desc
     ```

   - Requête permettant d'afficher un identifiant de commande spécifique :

     ```
     fields @timestamp, @message
     | filter @message like /10104661421959001813000000001/
     | sort @timestamp desc
     ```

6. Examinez tous les journaux du cluster de bases de données produits par la requête distribuée.

Exemples de journaux

Les exemples suivants illustrent l'utilisation du suivi des requêtes.

Corrélation des journaux pour les requêtes sujettes aux erreurs

Dans cet exemple, la commande `TRUNCATE` est exécutée sur la table `customers`, alors que cette table n'existe pas.

Sans suivi des requêtes

Fichier journal PostgreSQL sur le routeur coordinateur :

```
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:DETAIL: relation
 "public.customers" does not exist
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:CONTEXT: remote
 SQL command: truncate public.customers;
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[27503]:STATEMENT:
 truncate customers;
```

Fichier journal PostgreSQL sur une partition participante :

```
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[ 27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]: postgres@postgres_limitless:[ 27503]:STATEMENT:
 truncate customers;
```

Il s'agit de journaux types. Ils ne disposent pas des identifiants précis nécessaires pour corréler facilement les requêtes au sein du cluster de bases de données.

Avec suivi des requêtes

Fichier journal PostgreSQL sur le routeur coordinateur :

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:ERROR: failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:DETAIL: relation
 "public.customers" does not exist
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:CONTEXT: remote SQL
 command: truncate public.customers;
```

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:STATEMENT: /* tid =
 1126253375719408502700000000011 */ truncate customers;
```

Fichier journal PostgreSQL sur une partition participante :

```
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:ERROR:  failed to
 execute remote query
2023-09-26 04:03:19 UTC:[local]:postgres@postgres_limitless:[27503]:STATEMENT:  /* tid
 = 1126253375719408502700000000010 */ truncate customers;
```

En présence d'un suivi des requêtes, chaque ligne de journal est associée à un identifiant unique composé de 31 chiffres. Ici, `1126253375719408502700000000011` et `1126253375719408502700000000010` représentent les ID de suivi pour les nœuds coordinateur et participant, respectivement.

- Identifiant de transaction : `11262533757194085027`
- Identifiant de commande : `1126253375719408502700000000001`
- Identifiant du type de nœud : le dernier chiffre `1` ou `0`, indique respectivement un routeur coordinateur et un nœud participant.

Corrélation des journaux pour déterminer la durée d'exécution de la requête sur différents nœuds

Dans cet exemple, le paramètre `log_min_duration_statement` a été défini sur `0` pour afficher la durée de toutes les requêtes.

Sans suivi des requêtes

```
2024-01-15 07:28:46 UTC:[local]:postgres@postgres_limitless:[178322]:LOG: duration:
 12.779 ms statement: select * from customers;
```

Avec suivi des requêtes

Fichier journal PostgreSQL sur le routeur coordinateur :

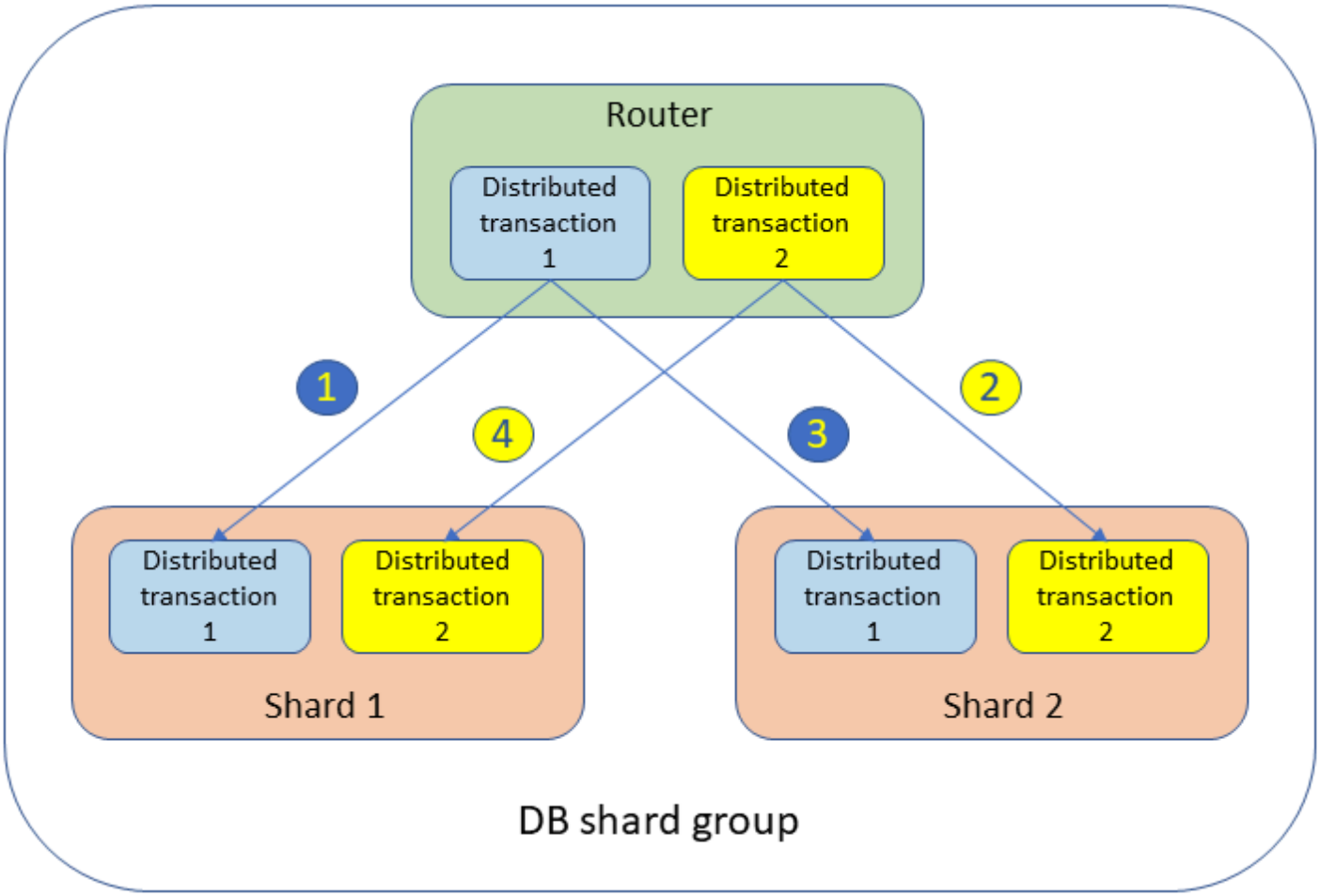
```
2024-01-15 07:32:08 UTC:[local]:postgres@postgres_limitless:[183877]:LOG: duration:
 12.618 ms statement: /* tid = 045766956624049708840000000011 */ select * from
 customers;
```

Fichier journal PostgreSQL sur une partition participante :

```
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.279 ms statement: /* tid = 045766956624049708840000000010 */ START
 TRANSACTION ISOLATION LEVEL READ COMMITTED
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.249 ms parse <unnamed>: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.398 ms bind <unnamed>/c1: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.019 ms execute <unnamed>/c1: SELECT customer_id, fname, lname FROM
 public.customers
2024-01-15 07:32:08 UTC:localhost(46358):postgres@postgres_limitless:[183944]:LOG:
 duration: 0.073 ms statement: /* tid = 045766956624049708840000000010 */ COMMIT
 TRANSACTION
```

Blocages distribués dans Aurora PostgreSQL Limitless Database

Dans un groupe de partitions de base de données, des blocages peuvent survenir entre des transactions distribuées entre différents routeurs et partitions. Par exemple, deux transactions distribuées simultanées couvrant deux partitions sont exécutées, comme illustré ci-après.



Les transactions verrouillent les tables et créent des événements d'attente dans les deux partitions comme suit :

1. Transaction distribuée 1 :

   ```
   UPDATE table SET value = 1 WHERE key = 'shard1_key';
   ```

   Cette transaction conserve un verrou sur la partition 1.

2. Transaction distribuée 2 :

```
UPDATE table SET value = 2 WHERE key = 'shard2_key';
```

Cette transaction conserve un verrou sur la partition 2.

3. Transaction distribuée 1 :

```
UPDATE table SET value = 3 WHERE key = 'shard2_key';
```

La transaction distribuée 1 est en attente sur la partition 2.

4. Transaction distribuée 2 :

```
UPDATE table SET value = 4 WHERE key = 'shard1_key';
```

La transaction distribuée 2 est en attente sur la partition 1.

Dans ce scénario, ni la partition 1 ni la partition 2 ne détectent le problème : la transaction 1 attend la transaction 2 sur la partition 2, tandis que la transaction 2 attend la transaction 1 sur la partition 1. D'un point de vue global, la transaction 1 attend la transaction 2, et la transaction 2 attend la transaction 1. Cette situation dans laquelle deux transactions sur deux partitions différentes s'attendent l'une l'autre est appelée blocage distribué.

Aurora PostgreSQL Limitless Database peut détecter et résoudre automatiquement les blocages distribués. Un routeur du groupe de partitions de base de données est averti lorsqu'une transaction met trop de temps à acquérir une ressource. Le routeur recevant la notification commence à collecter les informations nécessaires auprès de l'ensemble des routeurs et des partitions du groupe de partitions de base de données. Le routeur met ensuite fin aux transactions impliquées dans un blocage distribué, jusqu'à ce que les autres transactions du groupe de partitions de base de données puissent se poursuivre sans se bloquer mutuellement.

Lorsque le routeur met fin à une transaction impliquée dans un blocage distribué, le message d'erreur ci-dessous est envoyé :

```
ERROR: aborting transaction participating in a distributed deadlock
```

Le paramètre de `rds_aurora.limitless_distributed_deadlock_timeout` du cluster de bases de données définit la durée pendant laquelle chaque transaction attend une ressource avant de demander au routeur de vérifier s'il existe un blocage distribué. Vous pouvez augmenter la valeur

du paramètre si votre charge de travail est moins susceptible de rencontrer des situations de blocage. La valeur par défaut est de `1000` millisecondes (1 seconde).

Le cycle de blocage distribué est publié dans les journaux PostgreSQL lorsqu'un blocage entre nœuds est détecté et résolu. Les informations relatives à chaque processus impliqué dans le blocage sont les suivantes :

- Nœud coordinateur qui a lancé la transaction
- ID de transaction virtuelle (xid) de la transaction sur le nœud coordinateur, au format *`backend_id`*`/`*`backend_local_xid`*
- ID de session distribuée de la transaction

Gestion d’Aurora PostgreSQL Limitless Database

Les rubriques suivantes décrivent comment gérer vos clusters de bases de données Aurora PostgreSQL Limitless Database.

Rubriques

Considérations relatives à la taille des bases de données et des tables

Dans Aurora PostgreSQL Limitless Database, chaque partition contient une table partitionnée, elle-même subdivisée en plusieurs tranches de table. Le nombre de tranches varie en fonction du nombre de partitions disponibles dans le groupe de partitions de base de données. Chaque tranche de table peut atteindre une taille maximale de 32 Tio, tandis que chaque partition dispose d’une capacité maximale de 128 Tio. La taille des tables de référence est limitée à 32 Tio pour l’ensemble du groupe de partitions de base de données.

> **Note**
>
> La capacité maximale de chaque nœud (routeur ou partition) est de 128 Tio, car il s’agit de la capacité maximale d’un cluster de bases de données Aurora PostgreSQL.

Le nombre maximum de relations par base de données (y compris les tables, les vues et les index) dans Aurora PostgreSQL et Aurora PostgreSQL Limitless Database est de 1 431 650 303.

Pour plus d’informations, consultez Annexe K. Limites de PostgreSQL dans la documentation PostgreSQL et Limites de taille Amazon Aurora.

Récupération de l’espace de table via une opération de vacuum

PostgreSQL Multiversion Concurrency Control (MVCC) permet de préserver l’intégrité des données en enregistrant une copie interne des lignes mises à jour ou supprimées jusqu’à ce qu’une transaction soit validée ou annulée. Ces copies, également appelées tuples, peuvent provoquer un gonflement de la table si elles ne sont pas nettoyées régulièrement. Les instances PostgreSQL ordonnent les transactions selon leurs ID de transaction, et PostgreSQL utilise un modèle MVCC basé sur les ID de transaction pour déterminer la visibilité des tuples et assurer l’isolation des transactions. Chaque transaction établit un instantané des données, et chaque tuple possède une version. L’instantané et la version sont tous deux basés sur l’ID de transaction.

Pour nettoyer les données, l’utilitaire `VACUUM`exécute quatre fonctions clés dans PostgreSQL :

- `VACUUM` : supprime les versions de ligne expirées, rendant ainsi l’espace disponible pour une réutilisation.
- `VACUUM FULL` : permet une défragmentation complète en supprimant les lignes inactives et en compactant les tables, en réduisant la taille et en augmentant l’efficacité.
- `VACUUM FREEZE` : protège contre les problèmes de bouclage des ID de transaction en marquant les anciennes versions de lignes comme étant bloquées.
- `VACUUM ANALYZE` : supprime les versions des lignes inactives et met à jour les statistiques de planification des requêtes de la base de données. Il s’agit d’une combinaison des fonctions `VACUUM` et `ANALYZE`. Pour plus d’informations sur le fonctionnement de `ANALYZE` dans Aurora PostgreSQL Limitless Database, consultez ANALYSE.

Comme dans MVCC, l’opération de vacuum dans Aurora PostgreSQL est basée sur l’ID de transaction. Si une transaction est en cours lorsque le l’opération de vacuum commence, les lignes qui sont toujours visibles pour cette transaction ne sont pas supprimées.

Pour plus d’informations sur l’utilitaire `VACUUM`, consultez VACUUM dans la documentation PostgreSQL. Pour plus d’informations sur la prise en charge de `VACUUM` dans Aurora PostgreSQL Limitless Database, consultez VACUUM.

Rubriques

- AUTOVACUUM
- Vacuum basé sur le temps dans Aurora PostgreSQL Limitless Database
- Utilisation des statistiques de base de données pour l’opération de vacuum

- Différences dans le comportement des opérations de vacuum entre Aurora PostgreSQL et Aurora PostgreSQL Limitless Database

AUTOVACUUM

Aurora PostgreSQL utilise les utilitaires `VACUUM` et `AUTOVACUUM` pour supprimer les tuples inutiles. Le mécanisme sous-jacent de l'`AUTOVACUUM` et du `VACUUM` manuel sont les mêmes ; la seule différence réside dans l'automatisation.

`AUTOVACUUM` dans Aurora PostgreSQL et Aurora PostgreSQL Limitless Database est une combinaison des utilitaires `VACUUM` et `ANALYZE`. `AUTOVACUUM` détermine les bases de données et les tables à nettoyer, selon une règle prédéfinie, telle que le pourcentage de tuples inactifs et le nombre d'insertions.

Par exemple, `AUTOVACUUM` « se réveille » périodiquement pour effectuer un nettoyage. L'intervalle est contrôlé par le paramètre `autovacuum_naptime`. La valeur par défaut est de 1 minute. Les valeurs par défaut des paramètres de configuration `AUTOVACUUM` et `VACUUM` sont les mêmes dans Aurora PostgreSQL Limitless Database et Aurora PostgreSQL.

Le démon `AUTOVACUUM`, s'il est activé, émet automatiquement des commandes `ANALYZE` chaque fois que le contenu d'une table a suffisamment changé. Dans Aurora PostgreSQL Limitless Database, `AUTOVACUUM` émet une command `ANALYZE` sur les routeurs et les partitions.

Pour plus d'informations sur le démon `AUTOVACUUM` et les paramètres de stockage des tables associés à `AUTOVACUUM`, consultez Le démon autovacuum et les Paramètres de stockage dans la documentation PostgreSQL.

Vacuum basé sur le temps dans Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database est un système distribué, ce qui signifie que plusieurs instances peuvent être impliquées dans une transaction. Par conséquent, la visibilité basée sur l'ID de transaction ne s'applique pas. Aurora PostgreSQL Limitless Database utilise à la place un mécanisme de visibilité basé sur le temps, car les ID de transaction ne sont pas unifiés entre les instances, tandis que le temps, lui, peut l'être. Chaque instantané de transaction et chaque version de tuple se réfèrent au temps plutôt qu'à l'ID de transaction. Plus précisément, chaque instantané de transaction est associé à une heure de début, et chaque tuple enregistre une heure de création (au moment d'une `INSERT` ou d'une `UPDATE`) ainsi qu'une heure de suppression (au moment d'une `DELETE`).

Pour maintenir la cohérence des données entre les instances du groupe de partitions de base de données, Aurora PostgreSQL Limitless Database doit s'assurer que l'opération de vacuum ne supprime aucun tuple encore visible pour les transactions actives du groupe de partitions de base de données. Par conséquent, les opérations de vacuum sont également basées sur le temps dans Aurora PostgreSQL Limitless Database. D'autres aspects de `VACUUM` demeurent inchangés, notamment le fait que, pour exécuter `VACUUM` sur une table donnée, l'utilisateur doit disposer d'un accès à cette table.

> **Note**
>
> Il est vivement déconseillé de laisser les transactions ouvertes pendant de longues périodes. Les opérations de vacuum basées sur le temps consomme plus de mémoire que celles qui sont basées sur l'ID de transaction.

L'exemple suivant illustre le fonctionnement des opérations de vacuum basées sur le temps.

1. Une table client est répartie sur quatre partitions.
2. La transaction 1 commence par une lecture répétable et ne cible qu'une seule partition (partition 1). Cette transaction reste ouverte.

   La transaction 1 est plus ancienne que toute autre transaction lancée après elle.
3. La transaction 2 démarre plus tard et supprime tous les tuples d'une table, puis valide la transaction.
4. Si l'`AUTOVACUUM` ou le `VACUUM` manuel tente de nettoyer les tuples inactifs (résultant de la transaction 2), rien n'est supprimé.

   Cela est vrai non seulement pour la partition 1, mais également pour les partitions 2 à 4, car la transaction 1 peut toujours avoir besoin d'accéder à ces tuples. Ils sont toujours visibles pour la transaction 1 grâce au MVCC.

La dernière étape est réalisée par le biais d'une synchronisation, afin que toutes les partitions soient informées de la transaction 1, même si celle-ci ne les concerne pas toutes.

Utilisation des statistiques de base de données pour l'opération de vacuum

Pour obtenir des informations sur les tuples susceptibles de nécessiter un nettoyage, utilisez la vue limitless_stat_all_tables, qui fonctionne de la même manière que pg_stat_all_tables. L'exemple suivant illustre l'interrogation de la vue.

```
SELECT * FROM rds_aurora.limitless_stat_all_tables WHERE relname LIKE '%customer%';
```

De même, pour les statistiques de base de données, utilisez limitless_stat_database au lieu de pg_stat_database, et limitless_stat_activity plutôt que pg_stat_activity.

Pour vérifier l'utilisation du disque de la table, utilisez la fonction limitless_stat_relation_sizes, qui fonctionne de la même manière que pg_relation_size. Les exemples suivants illustrent l'interrogation de la fonction.

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public','customer');
```

Pour suivre la progression d'une opération VACUUM sur Aurora PostgreSQL Limitless Database, utilisez la vue limitless_stat_progress_vacuum au lieu de pg_stat_progress_vacuum. L'exemple suivant illustre l'interrogation de la vue.

```
SELECT * FROM rds_aurora.limitless_stat_progress_vacuum;
```

Pour plus d'informations, consultez Vues d'Aurora PostgreSQL Limitless Database et Fonctions d'Aurora PostgreSQL Limitless Database.

Différences dans le comportement des opérations de vacuum entre Aurora PostgreSQL et Aurora PostgreSQL Limitless Database

Les différences suivantes distinguent également Aurora PostgreSQL et Aurora PostgreSQL Limitless Database en ce qui concerne le processus de vacuum :

- Aurora PostgreSQL exécute des opérations VACUUM sur les ID de transaction jusqu'à la plus ancienne transaction en cours. S'il n'y a aucune transaction en cours dans la base de données, VACUUM exécute l'opération jusqu'à la dernière transaction.
- Aurora PostgreSQL Limitless Database synchronise le plus ancien instantané toutes les 10 secondes. Par conséquent, VACUUM peut ne pas exécuter l'opération sur les transactions ayant été effectuées au cours des 10 dernières secondes.

Pour plus d’informations sur la prise en charge de `VACUUM` dans Aurora PostgreSQL Limitless Database, consultez VACUUM dans le Référence Aurora PostgreSQL Limitless Database.

Surveillance d’Aurora PostgreSQL Limitless Database

Vous pouvez utiliser Amazon CloudWatch, Enhanced Monitoring et Performance Insights pour surveiller Aurora PostgreSQL Limitless Database. De nouvelles fonctions et vues statistiques, ainsi que des événements d’attente, sont également disponibles dans Aurora PostgreSQL Limitless Database et peuvent être utilisés à des fins de surveillance et de diagnostic.

Rubriques

Surveillance d’Aurora PostgreSQL Limitless Database avec Amazon CloudWatch

Les métriques CloudWatch pour Aurora PostgreSQL Limitless Database sont présentées selon les dimensions suivantes :

- DBShardGroup
- DBShardGroupRouterAggregation
- DBShardGroupInstance
- DBClusterIdentifier

Pour plus d’informations sur les métriques CloudWatch, consultez Surveillance des métriques Amazon Aurora avec Amazon CloudWatch.

Métriques DBShardGroup

Pour consulter les métriques `DBShardGroup` relatives à Aurora PostgreSQL Limitless Database dans la console CloudWatch, choisissez RDS, puis DBShardGroup.

Vous pouvez suivre les métriques CloudWatch suivantes :

- `DBShardGroupACUUtilization` : utilisation de l’unité de capacité Aurora (ACU) en pourcentage calculé à partir de la `DBShardGroupCapacity` divisée par `DBShardGroupMaxACU`.
- `DBShardGroupCapacity` : nombre d’ACU consommées par les instances d’écriture du groupe de partitions de base de données.
- `DBShardGroupComputeRedundancyCapacity` : nombre d’ACU consommées par les instances de secours du groupe de partitions de base de données.
- `DBShardGroupMaxACU` ; nombre maximum d’ACU configurées pour le groupe de partitions de base de données.
- `DBShardGroupMinACU` : nombre minimum d’ACU requis par le groupe de partitions de base de données.

La clé de dimension `DBShardGroupIdentifier` est disponible pour agréger les métriques `DBShardGroup`.

Métriques DBShardGroupRouterAggregation

Pour consulter les métriques `DBShardGroupRouterAggregation` relatives à Aurora PostgreSQL Limitless Database dans la console CloudWatch, choisissez RDS, puis DBShardGroupRouterAggregation.

Vous pouvez suivre les métriques CloudWatch suivantes :

- `CommitThroughput` : le nombre moyen d'opérations de validation par seconde sur tous les nœuds de routeur du groupe de partitions de base de données.
- `DatabaseConnections` : la somme de toutes les connexions entre tous les nœuds de routeur du groupe de partitions de base de données.

Métriques DBShardGroupInstance

Une métrique DBShardGroupInstance est l'instance de base de données individuelle au sein de chaque partition ou sous-cluster de routeur.

Pour consulter les métriques `DBShardGroupInstance` relatives à Aurora PostgreSQL Limitless Database dans la console CloudWatch, choisissez RDS, puis DBShardGroupInstance.

Vous pouvez suivre les métriques CloudWatch suivantes :

- `ACUUtilization` : le pourcentage calculé comme la métrique `ServerlessDatabaseCapacity` divisée par le nombre maximal d'ACU attribuées au sous-cluster.
- `AuroraReplicaLag` : pour les clusters Limitless où la redondance de calcul est activée, cette valeur indique la latence lors de la réplication des mises à jour depuis l'instance principale du sous-cluster.
- `AuroraReplicaLagMaximum` : pour les clusters Limitless où la redondance de calcul est activée, cette valeur indique la latence maximale lors de la réplication des mises à jour depuis l'instance principale du sous-cluster. Lorsque les réplicas en lecture sont supprimés ou renommés, il peut y avoir un pic temporaire de latence de réplication, le temps que l'ancienne ressource soit recyclée. Utilisez cette métrique pour déterminer si un basculement s'est produit en raison d'une latence de réplication élevée sur l'un de ses lecteurs.
- `AuroraReplicaLagMinimum` : pour les clusters Limitless où la redondance de calcul est activée, cette valeur indique la latence minimale lors de la réplication des mises à jour depuis l'instance principale du sous-cluster.

- `BufferCacheHitRatio` : le pourcentage de données et d'index fournis à partir du cache mémoire d'une instance (par opposition au volume de stockage).
- `CommitLatency` : la durée moyenne nécessaire au moteur et au stockage pour effectuer les opérations de validation pour un nœud particulier (routeur ou partition).
- `CommitThroughput` : nombre moyen d'opérations de validation par seconde.
- `CPUUtilization`— Utilisation de l'UC en pourcentage de la valeur d'ACU maximale attribuée au sous-cluster.
- `FreeableMemory` : la quantité de mémoire inutilisée qui est disponible lorsque le groupe de partitions est mis à l'échelle jusqu'à sa capacité maximale. Cette valeur est déterminée par les ACU attribuées au groupe de partitions. Pour chaque ACU dont la capacité actuelle est inférieure à la capacité maximale, cette valeur augmente d'environ 2 Gio. Ainsi, cette métrique ne tend pas vers zéro tant que le groupe de partitions de base de données n'a pas été augmenté verticalement jusqu'à sa limite maximale.
- `MaximumUsedTransactionIDs` : l'âge de l'ID de transaction non vidée le plus ancien, en transactions. Si cette valeur atteint 2 146 483 648 (2^31 - 1 000 000), la base de données est forcée à passer en mode de lecture seule afin d'éviter le bouclage des ID de transaction. Pour plus d'informations, consultez Prévention des échecs de bouclage de l'ID de transaction dans la documentation PostgreSQL.
- `NetworkReceiveThroughput` : quantité de débit réseau reçue des clients par chaque instance du groupe de partitions de base de données. Ce débit n'inclut pas le trafic réseau entre les instances du groupe de partitions de base de données et le volume de cluster.
- `NetworkThroughput` : le débit réseau agrégé (transmis et reçu) entre les clients et les routeurs, ainsi que les routeurs et les partitions du groupe de partitions de base de données. Ce débit n'inclut pas le trafic réseau entre les instances du groupe de partitions de base de données et le volume de cluster.
- `NetworkTransmitThroughput` : quantité de débit réseau envoyée aux clients par chaque instance du groupe de partitions de base de données. Ce débit n'inclut pas le trafic réseau entre les instances du groupe de partitions de base de données et le volume de cluster.
- `ReadIOPS` : le nombre moyen d'opérations d'entrée/sortie de lecture sur le disque par seconde (IOPS).
- `ReadLatency` : le temps moyen nécessaire pour chaque opération d'entrée/sortie (E/S) de lecture sur le disque.
- `ReadThroughput` : le nombre moyen d'octets lus sur le disque par seconde.

- `ServerlessDatabaseCapacity` : la capacité actuelle de la partition de base de données ou du sous-cluster de routeurs au sein du groupe de partitions de base de données.
- `StorageNetworkReceiveThroughput` : quantité de débit réseau reçue du sous-système de stockage Aurora par chaque instance du groupe de partitions de base de données.
- `StorageNetworkThroughput` : le débit réseau agrégé à la fois transmis vers et reçu depuis le sous-système de stockage Aurora, par chaque instance du groupe de partitions de base de données.
- `StorageNetworkTransmitThroughput` : la quantité de débit réseau envoyée au sous-système de stockage Aurora par chaque instance du groupe de partitions de base de données.
- `SwapUsage` : la quantité d'espace d'échange utilisé par le groupe de partitions de base de données.
- `TempStorageIOPS` : le nombre moyen d'opérations d'E/S réalisées sur le stockage local attaché à l'instance de base de données. Cette valeur prend en compte les opérations d'E/S de lecture et d'écriture.

  La métrique `TempStorageIOPS` peut être utilisée avec `TempStorageThroughput` pour diagnostiquer les rares cas où l'activité du réseau pour les transferts entre vos instances de base de données et vos périphériques de stockage locaux est responsable d'augmentations de capacité inattendues.
- `TempStorageThroughput` : le volume de données transférées depuis et vers le stockage local associé à un router ou à une partition.
- `WriteIOPS` : le nombre moyen d'IOPS d'écriture sur le disque.
- `WriteLatency` : le temps moyen nécessaire pour chaque opération d'entrée/sortie (E/S) d'écriture sur le disque.
- `WriteThroughput` : le nombre moyen d'octets écrits sur le disque par seconde.

Les clés de dimension suivantes sont disponibles pour agréger les métriques `DBShardGroupInstance` :

- `DBClusterIdentifier` : le cluster de bases de données Aurora PostgreSQL.
- `DBShardGroupIdentifier` : le groupe de partitions de base de données auquel appartient l'instance.
- `DBShardGroupSubClusterType` : le type de nœud, soit `Distributed Transaction Router` (routeur), soit `Data Access Shard` (partition).

- `DBShardGroupSubClusterIdentifier` : le nom du routeur ou de la partition auxquels appartient l'instance.

Voici des exemples d'agrégation de métriques CloudWatch :

- `CPUUtilization` totale de toutes les instances appartenant à une partition ou à un routeur donné(e) dans un groupe de partitions de base de données.
- `CPUUtilization` totale de toutes les instances d'un groupe de partitions de base de données.

Métriques DBClusterIdentifier

Pour consulter les métriques `DBClusterIdentifier` relatives à Aurora PostgreSQL Limitless Database dans la console CloudWatch, choisissez RDS, puis DBClusterIdentifier.

Lorsque vous utilisez Aurora PostgreSQL Limitless Database, le nombre d'opérations d'entrée/sortie (E/S) peut être supérieur à celui d'un cluster de bases de données Aurora. Vous pouvez suivre les métriques CloudWatch suivantes pour votre cluster Limitless Database :

- `VolumeReadIops` : le nombre d'opérations d'E/S de lecture facturées depuis un volume de cluster, rapportées par intervalles de 5 minutes.
- `VolumeWriteIops` : le nombre d'opérations d'E/S d'écriture disque sur le volume de cluster, rapportées par intervalles de 5 minutes.

Aurora PostgreSQL Limitless Database utilise la configuration de stockage en cluster. Aurora I/O-Optimized. Avec Aurora I/O-Optimized, vous payez un tarif mensuel fixe pour toutes vos opérations d'E/S, au lieu d'être facturé par million de requêtes E/S. Pour plus d'informations, consultez Configurations de stockage pour les clusters de bases de données Amazon Aurora.

Vous pouvez également bénéficier d'une capacité de stockage plus importante qu'avec un cluster de bases de données Aurora. Vous pouvez suivre les métriques CloudWatch suivantes pour le stockage :

- `BackupRetentionPeriodStorageUsed` : l'utilisation totale de stockage de sauvegarde continue facturée pour votre cluster Aurora PostgreSQL Limitless Database.
- `SnapshotStorageUsed` : l'utilisation totale de stockage d'instantanés facturée pour votre cluster Aurora PostgreSQL Limitless Database.

- `TotalBackupStorageBilled` : la somme des coûts liés à la rétention des sauvegardes automatiques et aux instantanés du cluster de bases de données.

  Pour plus d'informations sur les coûts de stockage des sauvegardes, consultez Comprendre l'utilisation du stockage de sauvegarde Amazon Aurora.
- `VolumeBytesUsed` : la quantité de stockage utilisée par votre cluster Aurora PostgreSQL Limitless Database, rapportée par intervalles de 5 minutes.

Surveillance d'Aurora PostgreSQL Limitless Database avec CloudWatch Database Insights

Le mode standard de Database Insights est requis pour activer Aurora PostgreSQL Limitless Database. Vous pouvez l'utiliser pour surveiller la charge de base de données (DB Load) de vos instances de base de données Limitless Database en temps réel. La charge de la base de données mesure le niveau d'activité de la session de votre base de données. Vous pouvez utiliser Database Insights pour analyser et résoudre les problèmes liés aux performances de vos instances de base de données Aurora PostgreSQL Limitless Database à grande échelle.

Pour plus d'informations sur CloudWatch Database Insights, consultez les rubriques suivantes.

- Surveillance des bases de données Amazon Aurora à l'aide de CloudWatch Database Insights
- CloudWatch Database Insights dans le Guide de l'utilisateur Amazon CloudWatch
- Démarrer avec CloudWatch Database Insights dans le Guide de l'utilisateur Amazon CloudWatch
- Configuration de votre base de données pour surveiller les requêtes SQL lentes avec Database Insights pour Amazon Aurora

Pour en savoir plus sur l'activation du mode Avancé ou du mode Standard de Database Insights, consultez les rubriques suivantes.

Rubriques

- Activation du mode Avancé de Database Insights pour Aurora PostgreSQL Limitless Database
- Activation du mode Standard de Database Insights pour Aurora PostgreSQL Limitless Database

Activation du mode Avancé de Database Insights pour Aurora PostgreSQL Limitless Database

Pour activer le mode Avancé de Database Insights pour Aurora PostgreSQL Limitless Database, suivez les procédures suivantes.

Activation du mode Avancé de Database Insights lors de la création d'un cluster de bases de données pour Aurora PostgreSQL Limitless Database

Activez le mode Avancé de Database Insights lors de la création d'une base de données pour Aurora PostgreSQL Limitless Database.

Console

Dans la console, vous pouvez activer le mode Avancé de Database Insights lorsque vous créez un cluster de bases de données. Les paramètres de Database Insights s'appliquent à toutes les instances de base de données de votre cluster de bases de données.

Pour activer le mode Avancé de Database Insights lors de la création d'un cluster de bases de données à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez Create database (Créer une base de données).
4. Dans la section Database Insights, sélectionnez le mode Avancé. Choisissez ensuite les options suivantes :
   - Conservation – Durée de conservation des données de Performance Insights. La période de conservation du mode Avancé de Database Insights doit être comprise entre 15 et 24 mois.
   - AWS KMS key – Spécifiez votre clé KMS. Performance Insights chiffre toutes les données potentiellement sensibles à l'aide votre clé KMS. Les données sont chiffrées en transit et au repos. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.
5. Choisissez Create database (Créer une base de données).

AWS CLI

Pour activer le mode Avancé de Database Insights lors de la création d'un cluster de bases de données, appelez la commande create-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :

- `--db-cluster-identifier` : l'identifiant du cluster de bases de données.
- `--database-insights-mode advanced` pour activer le mode Avancé de Database Insights.
- `--engine` : le cluster de bases de données doit utiliser le moteur de base de données `aurora-postgresql`.
- `--engine-version` : le cluster de bases de données doit utiliser l'une des versions du moteur de base de données.

  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type` : le cluster de bases de données doit utiliser la configuration de stockage du cluster de bases de données `aurora-iopt1`.
- `--cluster-scalability-type` : spécifie le mode de capacité de mise à l'échelle du cluster de bases de données Aurora. Lorsqu'il est défini sur `limitless`, le cluster fonctionne comme une base de données Aurora PostgreSQL Limitless Database. Lorsqu'il est défini sur `standard` (valeur par défaut), le cluster utilise la création d'instance de base de données normale.

  > **Note**
  >
  > Ce paramètre ne peut plus être modifié une fois le cluster de bases de données créé.

- `--master-username` : le nom de l'utilisateur principal du cluster de bases de données.
- `--master-user-password` : le mot de passe de l'utilisateur principal.
- `--enable-performance-insights` pour activer Performance Insights pour Database Insights.
- `--performance-insights-retention-period` : période de conservation des données de votre cluster de bases de données. Pour activer Database Insights, la période de conservation doit être d'au moins 465 jours.
- `--monitoring-interval` : l'intervalle, en secondes, entre les points lorsque des métriques Enhanced Monitoring sont collectées pour le cluster de bases de données. Cette valeur ne peut pas être `0`.
- `--monitoring-role-arn` : l'Amazon Resource Name (ARN) du rôle IAM qui autorise RDS à envoyer des métriques Enhanced Monitoring à Amazon CloudWatch Logs.
- `--enable-cloudwatch-logs-exports` : vous devez exporter les journaux `postgresql` vers CloudWatch Logs.

Dans l'exemple suivant, le mode Avancé de Database Insights est activé lors de la création d'un cluster de bases de données.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
```

```
--db-cluster-identifier my-limitless-cluster \
--database-insights-mode advanced \
--engine aurora-postgresql \
--engine-version 16.6-limitless \
--storage-type aurora-iopt1 \
--cluster-scalability-type limitless \
--master-username myuser \
--master-user-password mypassword \
--enable-performance-insights \
--performance-insights-retention-period 465 \
--monitoring-interval 5 \
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
--enable-cloudwatch-logs-exports postgresql
```

Pour Windows :

```
aws rds create-db-cluster ^
--db-cluster-identifier my-limitless-cluster ^
--database-insights-mode advanced ^
--engine aurora-postgresql ^
--engine-version 16.6-limitless ^
--storage-type aurora-iopt1 ^
--cluster-scalability-type limitless ^
--master-username myuser ^
--master-user-password mypassword ^
--enable-performance-insights ^
--performance-insights-retention-period 465 ^
--monitoring-interval 5 ^
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole ^
--enable-cloudwatch-logs-exports postgresql
```

RDS API

Pour activer le mode Avancé de Database Insights lors de la création d'un cluster de bases de données , définissez les paramètres suivants pour votre opération CreateDBCluster de l'API Amazon RDS.

- `DatabaseInsightsMode` sur `advanced`
- `Engine` sur `aurora-postgresql`
- `EngineVersion` sur une version de moteur disponible pour Limitless Database
- `StorageType` sur `aurora-iopt1`

- `ClusterScalabilityType` sur `limitless`
- `MasterUsername`
- `MasterUserPassword`
- `EnablePerformanceInsights` sur `True`
- `PerformanceInsightsRetentionPeriod` sur au moins `465` jours
- `MonitoringInterval` sur une valeur qui n'est pas `0`
- `MonitoringRoleArn` sur l'Amazon Resource Name (ARN) du rôle IAM qui autorise RDS à envoyer des métriques Enhanced Monitoring à Amazon CloudWatch Logs.

Activation du mode Avancé de Database Insights lors de la modification d'un cluster de bases de données pour Aurora PostgreSQL Limitless Database

Activation de Database Insights lors de la modification d'une base de données pour Aurora PostgreSQL Limitless Database.

> **Note**
>
> Pour activer Database Insights, il est nécessaire que chaque instance de base de données d'un cluster de bases de données ait les mêmes paramètres Performance Insights et Enhanced Monitoring.

Console

Dans la console, vous pouvez activer le mode Avancé de Database Insights lorsque vous modifiez un cluster de bases de données. Les paramètres de Database Insights s'appliquent à toutes les instances de base de données de votre cluster de bases de données.

Pour activer le mode Avancé de Database Insights lors de la modification d'un cluster de bases de données à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez un cluster de bases de données, puis Modifier.

4. Dans la section Database Insights, sélectionnez le mode Avancé. Choisissez ensuite les options suivantes :

   - Conservation – Durée de conservation des données de Performance Insights. La période de conservation du mode Avancé de Database Insights doit être comprise entre 15 et 24 mois.
   - AWS KMS key – Spécifiez votre clé KMS. Performance Insights chiffre toutes les données potentiellement sensibles à l'aide votre clé KMS. Les données sont chiffrées en transit et au repos. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.

5. Choisissez Continuer.
6. Pour Scheduling of Modifications (Planification des modifications), choisissez Appliquer immédiatement. Si vous choisissez Appliquer pendant la fenêtre de maintenance planifiée suivante, votre base de données ignore ce paramètre et active immédiatement le mode Avancé de Performance Insights.
7. Choisissez Modifier le cluster.

AWS CLI

Pour activer le mode Avancé de Database Insights lors de la modification d'un cluster de bases de données, appelez la commande modify-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :

- `--database-insights-mode advanced` pour activer le mode Avancé de Database Insights.
- `--db-cluster-identifier` : l'identifiant du cluster de bases de données.
- `--enable-performance-insights` pour activer Performance Insights pour Database Insights.
- `--performance-insights-retention-period` : période de conservation des données de votre cluster de bases de données. Pour activer le mode Avancé de Database Insights, la période de conservation doit être d'au moins 465 jours.

Dans l'exemple suivant, le mode Avancé Database Insights est activé lors de la modification d'un cluster de bases de données.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --database-insights-mode advanced \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 465
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --database-insights-mode advanced ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 465
```

RDS API

Pour activer le mode Avancé de Database Insights lorsque vous modifiez un cluster de bases de données, renseignez les paramètres suivants pour l'opération ModifyDBCluster de l'API Amazon RDS.

- `DatabaseInsightsMode` sur `advanced`
- `EnablePerformanceInsights` sur `True`
- `PerformanceInsightsRetentionPeriod` sur au moins `465` jours

Activation du mode Standard de Database Insights pour Aurora PostgreSQL Limitless Database

Pour activer le mode Standard de Database Insights pour Aurora PostgreSQL Limitless Database, suivez les procédures suivantes.

Activation du mode Standard de Database Insights lors de la création d'un cluster de bases de données pour Aurora PostgreSQL Limitless Database

Activez le mode Standard de Database Insights lors de la création d'une base de données pour Aurora PostgreSQL Limitless Database.

Console

Dans la console, vous pouvez activer le mode Standard de Database Insights lorsque vous créez un cluster de bases de données. Les paramètres de Database Insights s'appliquent à toutes les instances de base de données de votre cluster de bases de données.

Pour activer le mode Standard de Database Insights lors de la création d'un cluster de bases de données à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez Create database (Créer une base de données).
4. Dans la section Database Insights, sélectionnez le mode Standard. Choisissez ensuite les options suivantes :
   - Conservation – Durée de conservation des données de Performance Insights. Vous devez définir une période de conservation d'au moins 31 jours pour pouvoir créer un cluster de bases de données Aurora PostgreSQL Limitless Database.
   - AWS KMS key – Spécifiez votre clé KMS. Performance Insights chiffre toutes les données potentiellement sensibles à l'aide votre clé KMS. Les données sont chiffrées en transit et au repos. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.
5. Choisissez Create database (Créer une base de données).

AWS CLI

Pour activer le mode Standard de Database Insights lors de la création d'un cluster de bases de données, appelez la commande create-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :

- `--db-cluster-identifier` : l'identifiant du cluster de bases de données.
- `--database-insights-mode standard` pour activer le mode Standard de Database Insights.
- `--engine` : le cluster de bases de données doit utiliser le moteur de base de données `aurora-postgresql`.
- `--engine-version` : le cluster de bases de données doit utiliser l'une des versions du moteur de base de données.

  - `16.4-limitless`
  - `16.6-limitless`
- `--storage-type` : le cluster de bases de données doit utiliser la configuration de stockage du cluster de bases de données `aurora-iopt1`.
- `--cluster-scalability-type` : spécifie le mode de capacité de mise à l'échelle du cluster de bases de données Aurora. Lorsqu'il est défini sur `limitless`, le cluster fonctionne comme une base de données Aurora PostgreSQL Limitless Database. Lorsqu'il est défini sur `standard` (valeur par défaut), le cluster utilise la création d'instance de base de données normale.

  > **Note**
  >
  > Ce paramètre ne peut plus être modifié une fois le cluster de bases de données créé.

- `--master-username` : le nom de l'utilisateur principal du cluster de bases de données.
- `--master-user-password` : le mot de passe de l'utilisateur principal.
- `--enable-performance-insights` pour activer Performance Insights pour Database Insights.
- `--performance-insights-retention-period` : période de conservation des données de votre cluster de bases de données. Vous devez définir une période de conservation d'au moins 31 jours pour pouvoir créer un cluster de bases de données Aurora PostgreSQL Limitless Database.
- `--monitoring-interval` : l'intervalle, en secondes, entre les points lorsque des métriques Enhanced Monitoring sont collectées pour le cluster de bases de données. Cette valeur ne peut pas être `0`.
- `--monitoring-role-arn` : l'Amazon Resource Name (ARN) du rôle IAM qui autorise RDS à envoyer des métriques Enhanced Monitoring à Amazon CloudWatch Logs.
- `--enable-cloudwatch-logs-exports` : vous devez exporter les journaux `postgresql` vers CloudWatch Logs.

Dans l'exemple suivant, le mode Standard de Database Insights est activé lors de la création d'un cluster de bases de données.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
--db-cluster-identifier my-limitless-cluster \
--database-insights-mode standard \
--engine aurora-postgresql \
--engine-version 16.6-limitless \
--storage-type aurora-iopt1 \
--cluster-scalability-type limitless \
--master-username myuser \
--master-user-password mypassword \
--enable-performance-insights \
--performance-insights-retention-period 31 \
--monitoring-interval 5 \
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole \
--enable-cloudwatch-logs-exports postgresql
```

Pour Windows :

```
aws rds create-db-cluster ^
--db-cluster-identifier my-limitless-cluster ^
--database-insights-mode standard ^
--engine aurora-postgresql ^
--engine-version 16.6-limitless ^
--storage-type aurora-iopt1 ^
--cluster-scalability-type limitless ^
--master-username myuser ^
--master-user-password mypassword ^
--enable-performance-insights ^
--performance-insights-retention-period 31 ^
--monitoring-interval 5 ^
--monitoring-role-arn arn:aws:iam::123456789012:role/EMrole ^
--enable-cloudwatch-logs-exports postgresql
```

RDS API

Pour activer le mode Standard de Database Insights lorsque vous créez un cluster de bases de données , définissez les paramètres suivants pour votre opération CreateDBCluster de l'API Amazon RDS.

- `DatabaseInsightsMode` sur `standard`
- `Engine` sur `aurora-postgresql`
- `EngineVersion` vers une version de moteur disponible pour Limitless Database

- `StorageType` sur `aurora-iopt1`
- `ClusterScalabilityType` sur `limitless`
- `MasterUsername`
- `MasterUserPassword`
- `EnablePerformanceInsights` sur `True`
- `PerformanceInsightsRetentionPeriod` sur au moins `31` jours
- `MonitoringInterval` à une valeur qui n'est pas `0`
- `MonitoringRoleArn` à l'Amazon Resource Name (ARN) du rôle IAM qui autorise RDS à envoyer des métriques Enhanced Monitoring à Amazon CloudWatch Logs.

Activation du mode Standard de Database Insights lors de la modification d'un cluster de bases de données pour Aurora PostgreSQL Limitless Database

Activation de Database Insights lors de la modification d'une base de données pour Aurora PostgreSQL Limitless Database.

> **Note**
>
> Pour activer Database Insights, il est nécessaire que chaque instance de base de données d'un cluster de bases de données ait les mêmes paramètres Performance Insights et Enhanced Monitoring.

Console

Dans la console, vous pouvez activer le mode Standard de Database Insights lorsque vous créez un cluster de bases de données. Les paramètres de Database Insights s'appliquent à toutes les instances de base de données de votre cluster de bases de données.

Pour activer le mode Standard de Database Insights lors de la modification d'un cluster de bases de données à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez un cluster de bases de données, puis Modifier.

4. Dans la section Database Insights, sélectionnez le mode Standard. Choisissez ensuite les options suivantes :

   - Conservation – Durée de conservation des données de Performance Insights. Vous devez définir une période de conservation d'au moins 31 jours pour pouvoir créer un cluster de bases de données Aurora PostgreSQL Limitless Database.
   - AWS KMS key – Spécifiez votre clé KMS. Performance Insights chiffre toutes les données potentiellement sensibles à l'aide votre clé KMS. Les données sont chiffrées en transit et au repos. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.

5. Choisissez Continuer.
6. Pour Scheduling of Modifications (Planification des modifications), choisissez Appliquer immédiatement. Si vous choisissez Appliquer lors de la prochaine fenêtre de maintenance planifiée, votre base de données ignore ce paramètre et active immédiatement le mode Standard de Database Insights.
7. Choisissez Modifier le cluster.

AWS CLI

Pour activer le mode Standard de Database Insights lors de la modification d'un cluster de bases de données, appelez la commande modify-db-cluster de l'AWS CLI et fournissez les valeurs suivantes :

- `--db-cluster-identifier` : l'identifiant du cluster de bases de données.
- `--database-insights-mode standard` pour activer le mode Standard de Database Insights.
- `--enable-performance-insights` pour activer Performance Insights pour Database Insights.
- `--performance-insights-retention-period` : période de conservation des données de votre cluster de bases de données. Pour activer le mode Standard de Database Insights, la période de conservation doit être d'au moins 31 jours.

Dans l'exemple suivant, le mode Standard de Database Insights est activé lors de la modification d'un cluster de bases de données.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --database-insights-mode standard \
    --db-cluster-identifier sample-db-identifier \
    --enable-performance-insights \
    --performance-insights-retention-period 31
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --database-insights-mode standard ^
    --db-cluster-identifier sample-db-identifier ^
    --enable-performance-insights ^
    --performance-insights-retention-period 31
```

RDS API

Pour activer le mode Standard de Database Insights lorsque vous modifiez un cluster de bases de données, fournissez les paramètres suivants pour l'opération ModifyDBCluster de l'API Amazon RDS.

- `DatabaseInsightsMode` sur `standard`
- `EnablePerformanceInsights` sur `True`
- `PerformanceInsightsRetentionPeriod` sur au moins `31` jours

Surveillance d'Aurora PostgreSQL Limitless Database avec Amazon CloudWatch Logs

L'exportation des journaux PostgreSQL vers CloudWatch Logs est requise dans le cadre de l'activation d'Aurora PostgreSQL Limitless Database. Vous pouvez accéder à ces journaux et les analyser dans CloudWatch Logs Insights, de la même manière que vous accédez aux journaux PostgreSQL d'un cluster de bases de données Aurora PostgreSQL standard. Pour plus d'informations, consultez Analyse des journaux PostgreSQL à l'aide de CloudWatch Logs Insights.

Le nom du groupe de journaux pour le cluster de bases de données est le même que dans Aurora PostgreSQL :

```
/aws/rds/cluster/DB_cluster_ID/postgresql
```

Le nom du groupe de journaux pour le groupe de partitions de base de données prend la forme suivante :

```
/aws/rds/cluster/DB_cluster_ID/DB_shard_group_ID/postgresql
```

Il existe des flux de journaux pour chaque nœud (routeur ou partition). Leurs noms présentent le format suivant :

```
[DistributedTransactionRouter|
DataAccessShard]/node_cluster_serial_ID-node_instance_serial_ID/n
```

Exemples :

- Routeur – `DistributedTransactionRouter/6-6.2`
- Partition – `DataAccessShard/22-22.0`

> **Note**
>
> Vous ne pouvez pas afficher les fichiers journaux PostgreSQL du groupe de partitions de bases de données directement dans la console RDS, l'AWS CLI ou l'API RDS, comme c'est possible pour un cluster de bases de données. Vous devez utiliser CloudWatch Logs Insights pour les afficher.

Surveillance d’Aurora PostgreSQL Limitless Database à l’aide d’Enhanced Monitoring

La fonctionnalité Enhanced Monitoring est requise pour activer Aurora PostgreSQL Limitless Database. Vous pouvez l’utiliser pour surveiller le système d’exploitation de vos instances de base de données Limitless Database en temps réel.

Aurora publie les métriques Enhanced Monitoring dans CloudWatch Logs. Parmi les principales métriques disponibles figurent les connexions à la base de données, l’utilisation du stockage et la latence des requêtes. Ces métriques peuvent vous aider à identifier les goulots d’étranglement au niveau des performances.

Pour plus d’informations sur les métriques de surveillance améliorée, consultez Métriques de système d’exploitation pour Aurora.

Surveillance d'Aurora PostgreSQL Limitless Database avec Performance Insights

Utilisez Performance Insights pour surveiller votre cluster Aurora PostgreSQL Limitless Database. Performance Insights fonctionne de la même manière pour Aurora PostgreSQL Limitless Database que pour les clusters de bases de données Aurora standard. Cependant, dans Aurora PostgreSQL Limitless Database, le suivi des métriques s'effectue au niveau du groupe de partitions.

Les deux principales métriques à surveiller dans Performance Insights sont les suivantes :

- Charge de la base de données : mesure le niveau d'activité de votre base de données. La métrique clé de Performance Insights est `DBLoad`, qui est collectée toutes les secondes.

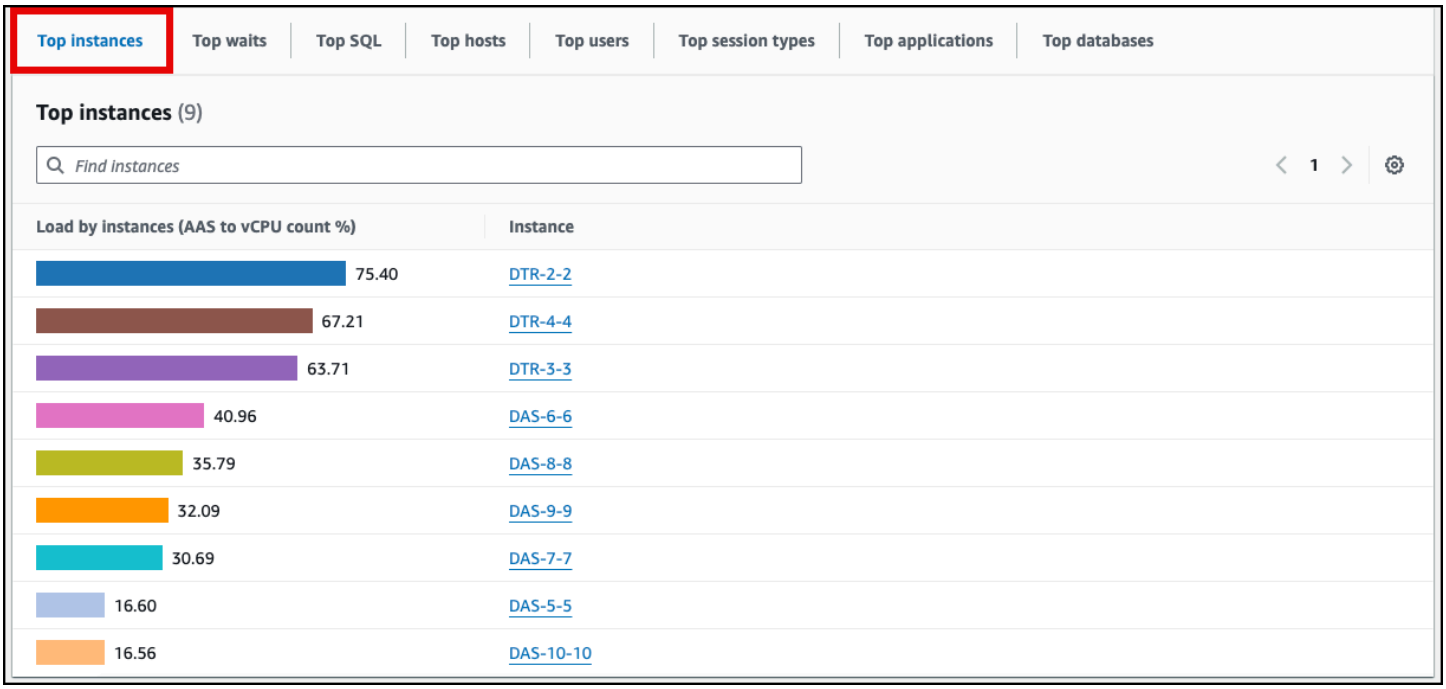
  L'unité de la métrique `DBLoad` dans Performance Insights est le nombre moyen de sessions actives (AAS). Pour obtenir les sessions actives en moyenne, Performance Insights échantillonne le nombre de sessions exécutant simultanément une requête. L'AAS correspond au nombre total de sessions divisé par le nombre total d'échantillons pour une période déterminée. Pour plus d'informations sur `DBLoad` et AAS, consultez Charge de base de données.
- Utilisation maximale de l'UC : la puissance de calcul maximale disponible pour votre base de données. Pour voir si les sessions actives dépassent l'utilisation maximale de l'UC, examinez leur relation sur la ligne `Max vCPU`. La valeur `Max vCPU` est déterminée par le nombre de cœurs de vCPU (UC virtuelle) pour votre instance de base de données. Pour plus d'informations sur `Max vCPU`, consultez Utilisation maximale de l'UC.

En outre, vous pouvez « découper » la métrique `DBLoad` en dimensions, qui représentent des sous-catégories de cette métrique. Les dimensions les plus utiles sont les suivantes :

- Instances principales : affichent la charge de base de données relative pour vos instances (partitions et routeurs) par ordre décroissant.
- Événement d'attente : indiquent que les requêtes SQL attendent la réalisation d'événements spécifiques avant de pouvoir s'exécuter. Les événements d'attente révèlent les endroits où l'activité est freinée.
- Principaux éléments SQL : indiquent les requêtes qui contribuent le plus à la charge de la base de données.

Pour plus d'informations sur les dimensions de Performance Insights, consultez Dimensions.

La figure suivante montre la dimension Instances principales d’un groupe de partitions de base de données.



Rubriques

- [Analyse de la charge de la base de données pour la base de données Aurora PostgreSQL Limitless à l’aide du tableau de bord de Performance Insights]

Analyse de la charge de la base de données pour la base de données Aurora PostgreSQL Limitless à l'aide du tableau de bord de Performance Insights

Avec Performance Insights, vous pouvez suivre les métriques au niveau du groupe de partitions et au niveau de l'instance pour une base de données Aurora PostgreSQL Limitless. Lorsque vous analysez la charge de base de données pour une base de données Aurora PostgreSQL Limitless, vous pouvez si vous le souhaitez comparer la charge de base de données pour chaque partition et routeur au nombre maximal de vCPU.

> **Note**
>
> Performance Insights et Enhanced Monitoring sont toujours activés dans la base de données Aurora PostgreSQL Limitless. La période de conservation minimale des données Performance Insights pour Limitless Database est de 31 jours (1 mois).

La vue Absolue indique le nombre moyen de sessions actives (AAS) et le nombre estimé de vCPU. La vue Relative montre le rapport entre l'AAS et le vCPU estimé.

Rubriques

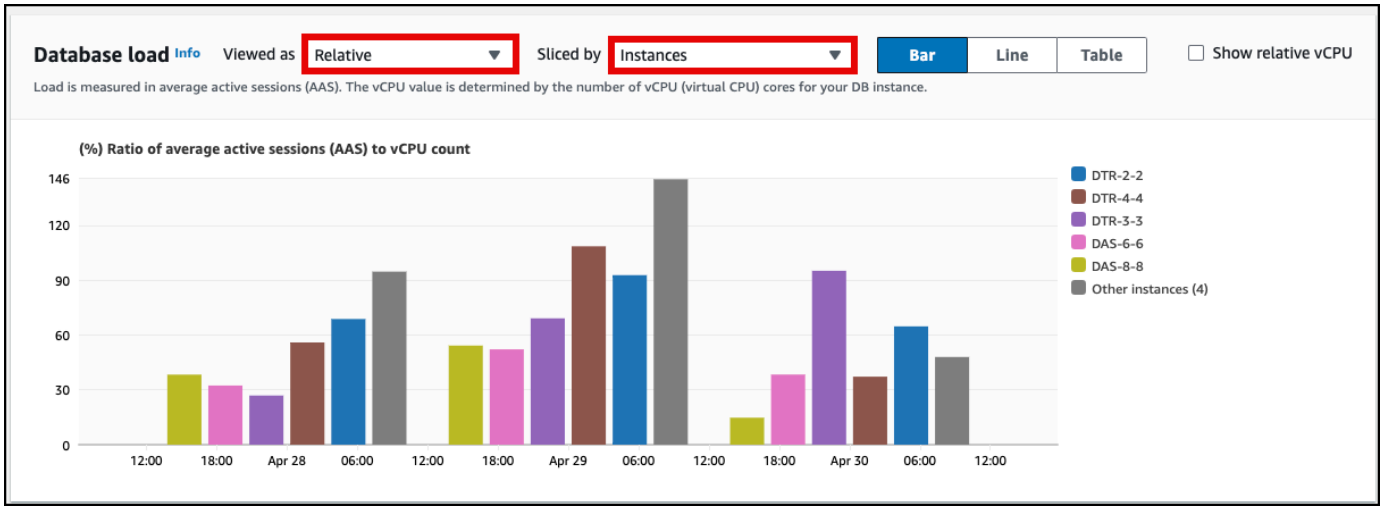
Analyse de la charge de la base de données relative pour la base de données Aurora PostgreSQL Limitless à l'aide du tableau de bord de Performance Insights

Vous pouvez si vous le souhaitez être améliorer les performances de votre base de données Aurora PostgreSQL Limitless en suivant la charge de base de données relative. Pour analyser la charge de base de données relative par instance pour votre base de données Aurora PostgreSQL Limitless, utilisez la procédure suivante.

Pour analyser la charge de base de données relative pour la base de données Aurora PostgreSQL Limitless à l'aide de la console

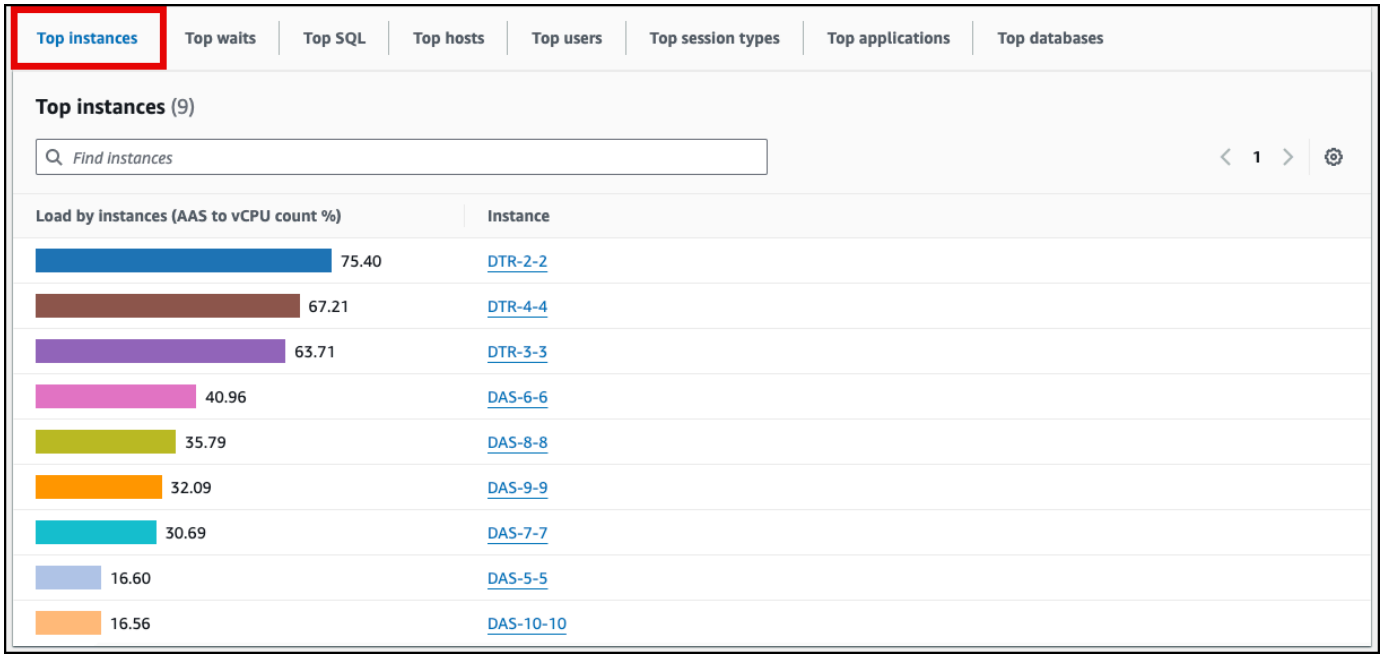
1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Performance Insights.
3. Choisissez une base de données Aurora PostgreSQL Limitless. Le tableau de bord de Performance Insights s'affiche pour cette base de données Aurora PostgreSQL Limitless.
4. Dans la section Charge de base de données, choisissez Instances pour Divisé par. Pour connaître le ratio entre le nombre moyen de sessions actives (AAS) et le nombre de cœurs de vCPU pour toutes les instances de votre base de données Aurora PostgreSQL Limitless, choisissez Relative for Considéré comme.

   Le graphique Nombre moyen de sessions actives affiche la charge de base de données pour les instances de votre base de données d'Aurora PostgreSQL Limitless.



5. Pour afficher les principales instances, cliquez sur l'onglet Les meilleures instances.

   Dans l'exemple suivant, l'instance avec la charge de base de données la plus élevée est `DTR-2-2`.



6. (Facultatif) Pour analyser la charge de base de données d'une instance dans votre base de données Aurora PostgreSQL Limitless, choisissez le nom de l'instance dans la colonne Instances. Pour afficher la charge de base de données pour `DTR-2-2`, choisissez `DTR-2-2` dans la colonne Instances.

> **Note**
>
> Vous pouvez consulter les métriques Performance Insights uniquement pour les instances d'une base de données Aurora PostgreSQL Limitless.

Analyse de la charge de la base de données en temps d'attente pour la base de données Aurora PostgreSQL Limitless à l'aide du tableau de bord de Performance Insights

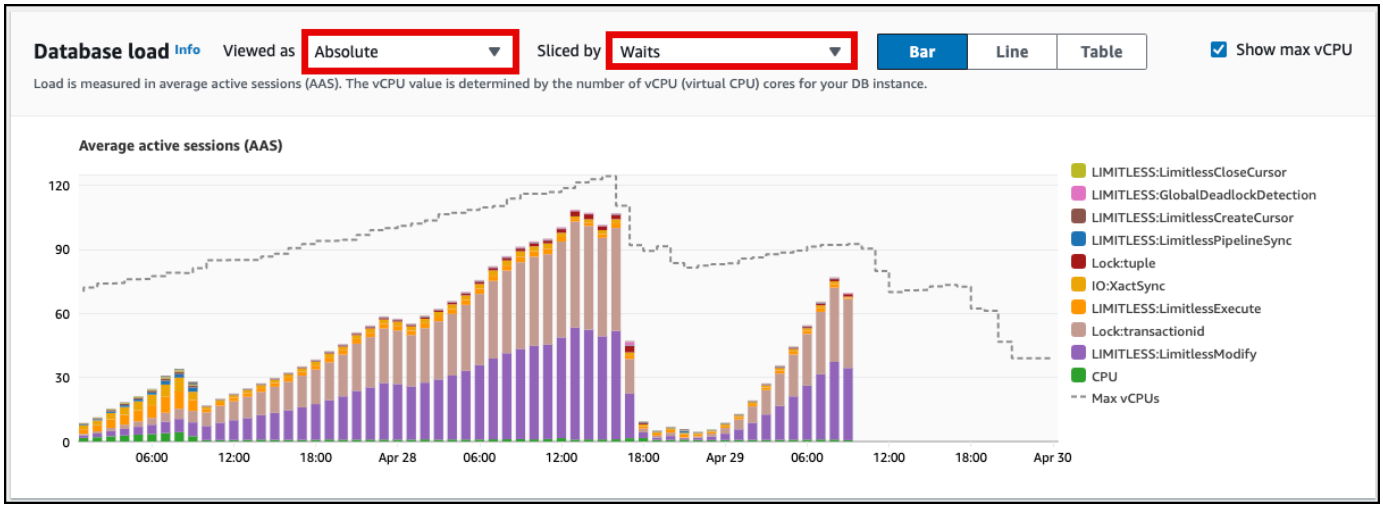
Vous souhaiterez peut-être améliorer les performances de votre base de données Aurora PostgreSQL Limitless en suivant les événements d'attente. Pour analyser les événements de charge de base de données par attente pour votre base de données Aurora PostgreSQL Limitless, suivez la procédure suivante.

Pour analyser la charge de base de données par temps d'attente pour la base de données Aurora PostgreSQL Limitless à l'aide de la console

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Performance Insights.

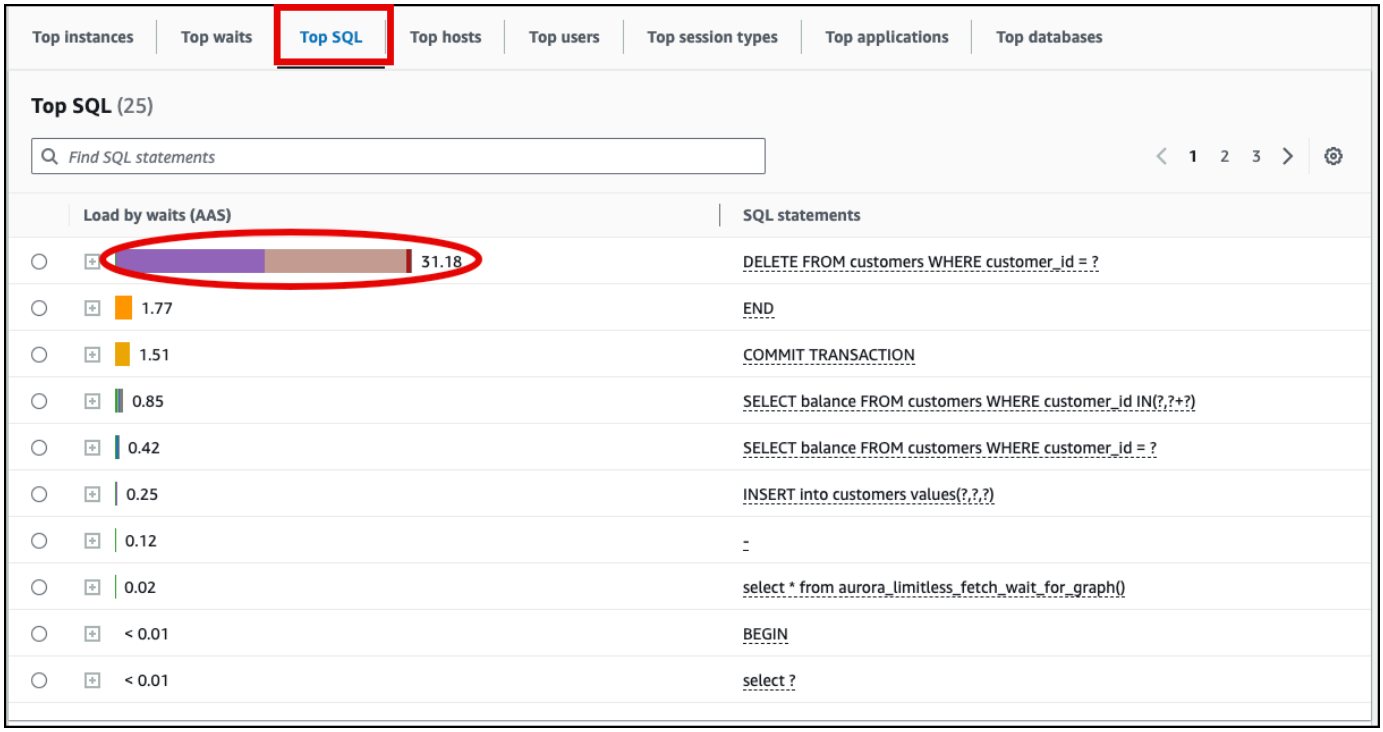
3. Choisissez une base de données Aurora PostgreSQL Limitless. Le tableau de bord de Performance Insights s'affiche pour cette base de données Aurora PostgreSQL Limitless.
4. Dans la section Charge de base de données, choisissez Attentes pour Divisé par. Pour afficher le nombre d'AAS et le nombre estimé de vCPU, choisissez Absolues pour Considéré comme.

   Le graphique Nombre moyen de sessions actives affiche la charge de base de données pour les instances de votre base de données d'Aurora PostgreSQL Limitless.



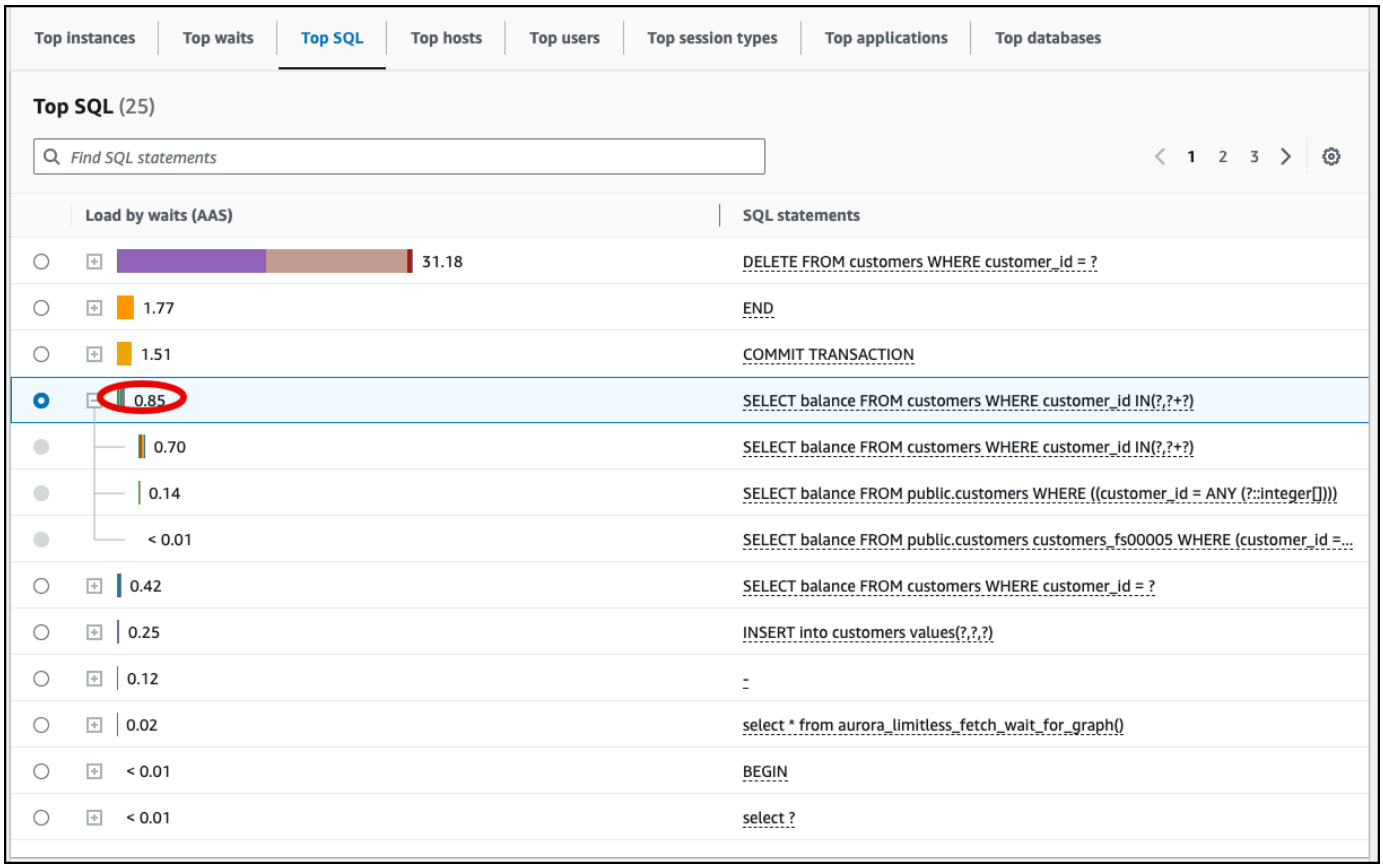
5. Faites défiler jusqu'à l'onglet Top SQL (Principaux éléments SQL).

   Dans l'exemple suivant, l'instruction SQL dont la charge par attentes est la plus élevée est l'instruction `DELETE`.



6. Choisissez l'instruction SQL pour afficher ses instructions de composants.

   Dans l'exemple suivant, l'instruction SELECT comporte 3 instructions de composants.



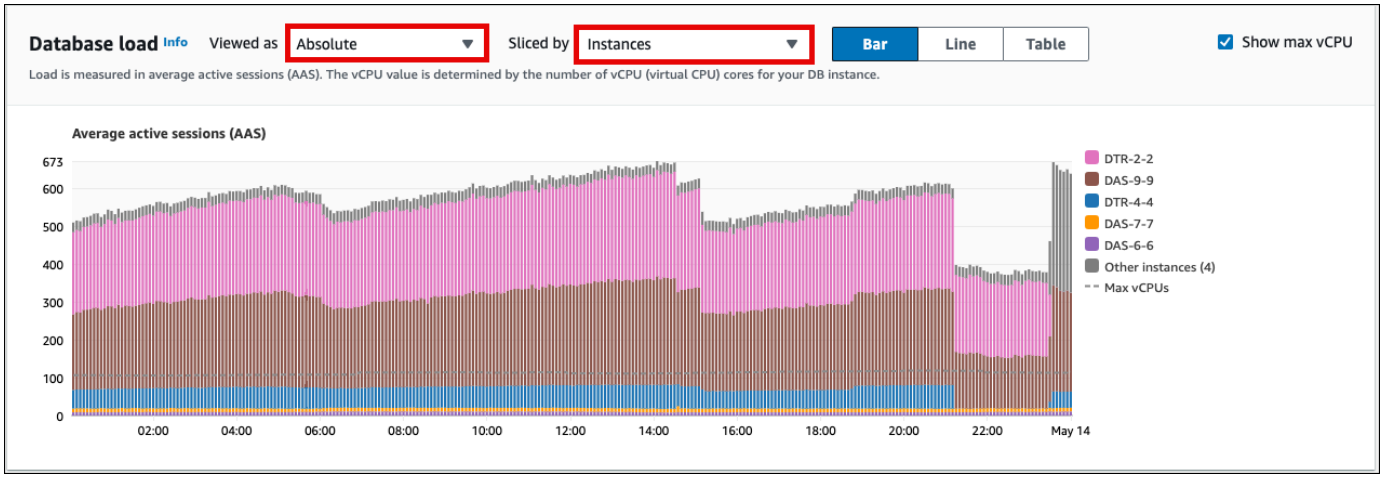
Analyse de la répartition de la charge de la base de données Aurora PostgreSQL Limitless à l'aide du tableau de bord de Performance Insights

Vous souhaiterez peut-être équilibrer la répartition de la charge pour les instances de votre base de données Aurora PostgreSQL Limitless. Pour analyser la répartition de la charge des instances sur une base de données Aurora PostgreSQL Limitless, suivez la procédure suivante.

Pour analyser la répartition de la charge des instances sur une base de données Aurora PostgreSQL Limitless à l'aide de la console

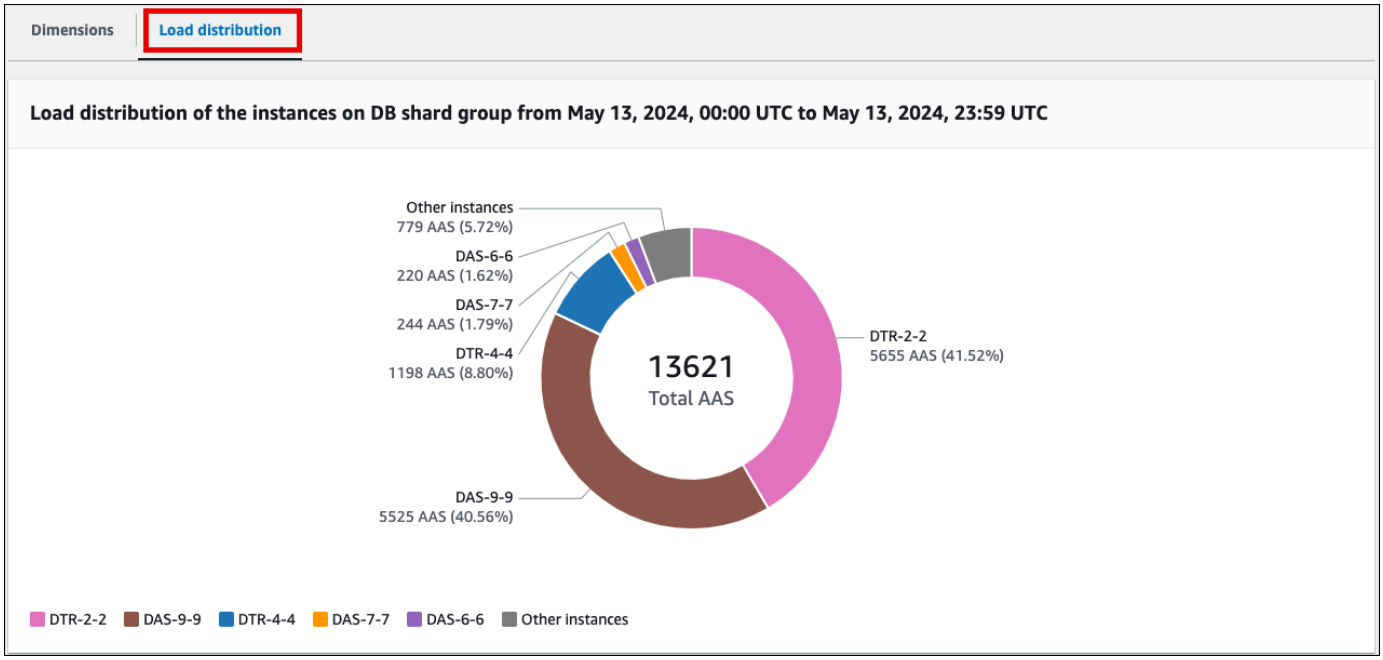
1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Performance Insights.
3. Choisissez une base de données Aurora PostgreSQL Limitless. Le tableau de bord de Performance Insights s'affiche pour cette base de données Aurora PostgreSQL Limitless.
4. Dans la section Charge de base de données, choisissez Instances pour Divisé par. Pour afficher le nombre d'AAS et le nombre estimé de vCPU pour toutes les instances de votre base de données Aurora PostgreSQL Limitless, choisissez Absolues pour Considéré comme.

   Le graphique Nombre moyen de sessions actives affiche la charge de base de données pour les instances de votre base de données d'Aurora PostgreSQL Limitless.



5. Pour voir un graphique de répartition de la charge des instances dans votre base de données Aurora PostgreSQL Limitless, choisissez l'onglet Répartition de charge.

   Dans l'exemple suivant, l'instance avec la charge de base de données la plus élevée est `DTR-2-2`.


Dimensions
Load distribution
Load distribution of the instances on DB shard group from May 13, 2024, 00:00 UTC to May 13, 2024, 23:59 UTC
Other instances
779 AAS (5.72%)
DAS-6-6
220 AAS (1.62%)
DAS-7-7
244 AAS (1.79%)
DTR-4-4
1198 AAS (8.80%)
13621
Total AAS
DTR-2-2
5655 AAS (41.52%)
DAS-9-9
5525 AAS (40.56%)
DTR-2-2
DAS-9-9
DTR-4-4
DAS-7-7
DAS-6-6
Other instances

Surveillance d’Aurora PostgreSQL Limitless Database avec Amazon GuardDuty RDS Protection

Amazon GuardDuty est un service de détection des menaces qui vous aide à protéger vos comptes, vos conteneurs, vos charges de travail et les données de votre environnement AWS. Grâce à des modèles de machine learning (ML) et à des fonctions de détection des anomalies et des menaces, GuardDuty surveille en continu diverses sources de journaux et les activités d’exécution afin d’identifier et de hiérarchiser les risques de sécurité potentiels ainsi que les activités malveillantes dans votre environnement.

GuardDuty RDS Protection analyse et établit un profil pour les événements de connexion afin de détecter les menaces d’accès potentielles à vos bases de données Amazon Aurora. Lorsque vous activez la protection RDS, GuardDuty consomme les événements de connexion RDS de vos bases de données Aurora. RDS Protection surveille ces événements et en établit le profil pour détecter les menaces potentielles de l’intérieur ou des acteurs externes.

Pour plus d’informations sur GuardDuty RDS Protection dans Aurora, consultez Surveillance des menaces avec Amazon GuardDuty RDS Protection pour Amazon Aurora.

Pour plus d’informations sur l’activation de GuardDuty RDS Protection, consultez GuardDuty RDS Protection dans le Guide de l’utilisateur Amazon GuardDuty.

Fonctions et vues dans Aurora PostgreSQL Limitless Database

Aurora PostgreSQL Limitless Database propose des fonctions et des vues supplémentaires. Elles sont basées sur les fonctions et les vues Aurora PostgreSQL correspondantes.

> **Note**
>
> Certaines statistiques peuvent produire des résultats incohérents si des transactions sont en cours d'exécution.

Rubriques

Fonctions d'Aurora PostgreSQL Limitless Database

Le tableau suivant présente les nouvelles fonctions d'Aurora PostgreSQL Limitless Database.

> **Note**
>
> Les fonctions répertoriées dans ce tableau se trouvent dans le schéma `rds_aurora`. Lorsque vous utilisez une fonction Limitless Database, assurez-vous d'inclure le nom d'objet entièrement qualifié : `rds_aurora.`*`object_name`*.

Fonctions d'Aurora PostgreSQL Limitless Database	Fonction d'Aurora PostgreSQL correspondante
limitless_backend_dsid	pg_backend_pid
limitless_cancel_session	pg_cancel_backend
limitless_stat_clear_snapshot	pg_stat_clear_snapshot
limitless_stat_database_size	pg_database_size

Fonctions d'Aurora PostgreSQL Limitless Database	Fonction d'Aurora PostgreSQL correspondante
limitless_stat_get_snapshot_timestamp	pg_stat_get_snapshot_timestamp
limitless_stat_prepared_xacts	pg_prepared_xacts
limitless_stat_relation_sizes	pg_indexes_size, pg_relation_size, pg_table_size, pg_total_relation_size
limitless_stat_reset	pg_stat_reset
limitless_stat_statements_reset	pg_stat_statements_reset
limitless_stat_system_waits	aurora_stat_system_waits
limitless_terminate_session	pg_terminate_backend
limitless_wait_report	aurora_wait_report

Les exemples suivants fournissent des détails sur les fonctions d'Aurora PostgreSQL Limitless Database. Pour plus d'informations sur les fonctions PostgreSQL, consultez Fonctions et opérateurs dans la documentation PostgreSQL.

limitless_backend_dsid

La fonction `limitless_backend_dsid` renvoie l'ID de la session distribuée pour la session en cours. Une session distribuée s'exécute sur un routeur d'un groupe de partitions de base de données et implique des processus dorsaux sur une ou plusieurs partitions du groupe de partitions de base de données.

L'exemple suivant montre comment utiliser la fonction `limitless_backend_dsid`.

```
SELECT rds_aurora.limitless_backend_dsid();

limitless_backend_dsid
------------------------
8CACD7B04D0FC2A5
(1 row)
```

limitless_cancel_session

La fonction `limitless_cancel_session` se comporte comme `pg_cancel_backend`, à la différence qu'elle essaie d'annuler tous les processus dorsaux liés à l'ID de session distribuée spécifié en leur envoyant un `SIGINT` (signal d'interruption).

Le paramètre d'entrée est le suivant :

- `distributed_session_id` (texte) : l'ID de la session distribuée à annuler.

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.
- `pid` (texte) : l'ID du processus dorsal.
- `success` (booléen) : indique si l'annulation s'est déroulée avec succès.

L'exemple suivant montre comment utiliser la fonction `limitless_cancel_session`.

```
SELECT * FROM rds_aurora.limitless_cancel_session('940CD5C81E3C796B');

 subcluster_id |  pid  | success
---------------+-------+---------
             1 | 26920 | t
(1 row)
```

limitless_stat_clear_snapshot

La fonction `limitless_stat_clear_snapshot` supprime l'instantané des statistiques actuelles ou les informations mises en cache sur tous les nœuds.

L'exemple suivant montre comment utiliser la fonction `limitless_stat_clear_snapshot`.

```
SELECT rds_aurora.limitless_stat_clear_snapshot();
```

limitless_stat_database_size

La fonction `limitless_stat_database_size` renvoie les tailles d'une base de données dans le groupe de partitions de base de données.

Le paramètre d'entrée est le suivant :

- `dbname` (nom) : la base de données dont vous souhaitez obtenir les tailles.

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.
- `subcluster_type` (texte) : le type de sous-cluster auquel appartient ce processus : `router` ou `shard`.
- `db_size` : la taille de la base de données de ce sous-cluster en octets.

L'exemple suivant montre comment utiliser la fonction `limitless_stat_database_size`.

```
SELECT * FROM rds_aurora.limitless_stat_database_size('postgres_limitless');

 subcluster_id | subcluster_type | db_size
---------------+-----------------+----------
             1 | router          |  8895919
             2 | router          |  8904111
             3 | shard           | 21929391
             4 | shard           | 21913007
             5 | shard           | 21831087
(5 rows)
```

limitless_stat_get_snapshot_timestamp

La fonction `limitless_stat_get_snapshot_timestamp` renvoie l'horodatage de l'instantané des statistiques en cours, ou NULL si aucun instantané n'a été créé. Un instantané est créé lors de la première consultation des statistiques cumulées dans une transaction, lorsque `stats_fetch_consistency` est défini sur `snapshot`. Renvoie une vue consolidée des horodatages des instantanés de tous les nœuds. Les colonnes `subcluster_id` et `subcluster_type` indiquent de quel nœud proviennent les données.

L'exemple suivant montre comment utiliser la fonction `limitless_stat_get_snapshot_timestamp`.

```
SELECT * FROM rds_aurora.limitless_stat_get_snapshot_timestamp();

 subcluster_id | subcluster_type | snapshot_timestamp
---------------+-----------------+--------------------
             1 | router          |
             2 | router          |
             3 | shard           |
             4 | shard           |
             5 | shard           |
(5 rows)
```

limitless_stat_prepared_xacts

La fonction `limitless_stat_prepared_xacts` renvoie des informations sur les transactions sur tous les nœuds actuellement préparés pour une validation en deux phases. Pour plus d'informations, consultez pg_prepared_xacts dans la documentation PostgreSQL.

L'exemple suivant montre comment utiliser la fonction `limitless_stat_prepared_xacts`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_prepared_xacts;

 subcluster_id | subcluster_type | transaction_id |             gid              |
         prepared            |  owner_id  |    database_id
---------------+-----------------+----------------+------------------------------
+-------------------------------+------------+--------------------
 8             | shard           |        5815978 | 7_4599899_postgres_limitless |
 2024-09-03 15:51:17.659603+00 | auroraperf | postgres_limitless
 12            | shard           |        4599138 | 7_4599899_postgres_limitless |
 2024-09-03 15:51:17.659637+00 | auroraperf | postgres_limitless
(2 rows)
```

limitless_stat_relation_sizes

La fonction `limitless_stat_relation_sizes` renvoie les différentes tailles d'une table dans le groupe de partitions de base de données.

Les paramètres d'entrée sont les suivants :

- `relnspname` (nom) : le nom du schéma contenant la table.
- `relname` (nom) : le nom de la table.

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.
- `subcluster_type` (texte) : le type de sous-cluster auquel appartient ce processus : `router` ou `shard`.
- `main_size` : la taille en octets du jeu de données principal de ce nœud.
- `fsm_size` : la taille en octets de la carte de l'espace libre de la table dans ce nœud.
- `vm_size` : la taille en octets de la carte de visibilité de la table dans ce nœud.
- `init_size` : la taille en octets de l'initialisation de la table dans ce nœud.
- `toast_size` : la taille en octets de la table Toast associée à la table de ce fork.

- `index_size` : la taille en octets de tous les indexes de la table dans ce nœud.
- `total_size` : la taille en octets de tous les segments de la table dans ce nœud.

L'exemple suivant montre comment utiliser la fonction `limitless_stat_relation_sizes` (certaines colonnes ont été omises).

```
SELECT * FROM rds_aurora.limitless_stat_relation_sizes('public','customers');

 subcluster_id | subcluster_type | main_size | fsm_size | vm_size | toast_size |
 table_size | total_size
---------------+-----------------+-----------+----------+---------+------------
+------------+------------
             1 | router          |         0 |        0 |       0 |          0 |
   0 |          0
             2 | router          |         0 |        0 |       0 |          0 |
   0 |          0
             3 | shard           |   4169728 |  4177920 | 1392640 |    1392640 |
 11132928 |   11132928
             4 | shard           |   4169728 |  4177920 | 1392640 |    1392640 |
 11132928 |   11132928
             5 | shard           |   3981312 |  4227072 | 1409024 |    1409024 |
 11026432 |   11026432
(5 rows)
```

limitless_stat_reset

La fonction `limitless_stat_reset` remet à zéro (0) l'ensemble des compteurs de statistiques de la base de données actuelle. Si `track_functions` est activé, la colonne `stats_reset` dans `limitless_stat_database` indique la dernière fois que les statistiques ont été réinitialisées pour la base de données. Par défaut, `limitless_stat_reset` ne peut être exécuté que par un super-utilisateur. Les autres utilisateurs peuvent obtenir une autorisation en utilisant le privilège `EXECUTE`.

L'exemple suivant montre comment utiliser la fonction `limitless_stat_reset`.

```
SELECT tup_inserted, tup_deleted FROM pg_stat_database
WHERE datname = 'postgres_limitless';

 tup_inserted | tup_deleted
--------------+-------------
          896 |           0
(1 row)
```

```
SELECT rds_aurora.limitless_stat_reset();

limitless_stat_reset
---------------------
(1 row)

SELECT tup_inserted, tup_deleted FROM pg_stat_database
WHERE datname = 'postgres_limitless';

tup_inserted | tup_deleted
-------------+-------------
           0 |           0
(1 row)
```

limitless_stat_statements_reset

La fonction `limitless_stat_statements_reset` ignore les statistiques collectées jusqu'à présent par `limitless_stat_statements` correspondant aux paramètres spécifiés `username`, `dbname`, `distributed_query_id` et `queryid`. Si aucun paramètre n'est spécifié, la valeur par défaut `""` ou `0` (non valide) est utilisée pour chacun d'entre eux, et les statistiques correspondant aux autres paramètres sont réinitialisées. Si aucun paramètre n'est spécifié, ou si tous les paramètres spécifiés sont `""` ou `0` (non valides), la fonction ignore toutes les statistiques. Si toutes les statistiques de la vue `limitless_stat_statements` sont supprimées, la fonction réinitialise également les statistiques de la vue `limitless_stat_statements_info`.

Les paramètres d'entrée sont les suivants :

- `username` (nom) : l'utilisateur qui a interrogé l'instruction.
- `dbname` (nom) : la base de données dans laquelle la requête a été exécutée.
- `distributed_query_id` (bigint) : l'ID de requête de la requête parent provenant du nœud coordinateur. Cette colonne est `NULL` s'il s'agit de la requête parent. Le nœud coordinateur transmet l'ID de requête distribué aux nœuds participants. Ainsi, pour les nœuds participants, les valeurs de l'ID de requête distribuée et de l'ID de requête sont différentes.
- `queryid` (bigint) : l'ID de requête de l'instruction.

L'exemple suivant montre comment utiliser la fonction `limitless_stat_statements_reset` pour réinitialiser toutes les statistiques collectées par `limitless_stat_statements`.

```
SELECT rds_aurora.limitless_stat_statements_reset();
```

limitless_stat_system_waits

La fonction `limitless_stat_system_waits` renvoie une vue consolidée des données d'événements d'attente provenant de `aurora_stat_system_waits`, qui rapporte l'activité d'attente à l'échelle du système pour une instance, à partir de tous les nœuds. Les colonnes `subcluster_id` et `subcluster_type` indiquent de quel nœud proviennent les données.

L'exemple suivant montre comment utiliser la fonction `limitless_stat_system_waits`.

```
postgres_limitless=> SELECT *
FROM rds_aurora.limitless_stat_system_waits() lssw, pg_catalog.aurora_stat_wait_event()
 aswe
WHERE lssw.event_id=aswe.event_id and aswe.event_name='LimitlessTaskScheduler';

 subcluster_id | subcluster_type | type_id | event_id  | waits  |  wait_time   |
 event_name
---------------+-----------------+---------+-----------+--------+--------------
+------------------------
             1 | router          |      12 | 201326607 | 677068 | 616942216307 |
 LimitlessTaskScheduler
             2 | router          |      12 | 201326607 | 678586 | 616939897111 |
 LimitlessTaskScheduler
             3 | shard           |      12 | 201326607 | 756640 | 616965545172 |
 LimitlessTaskScheduler
             4 | shard           |      12 | 201326607 | 755184 | 616958057620 |
 LimitlessTaskScheduler
             5 | shard           |      12 | 201326607 | 757522 | 616963183539 |
 LimitlessTaskScheduler
(5 rows)
```

limitless_terminate_session

La fonction `limitless_terminate_session` se comporte comme `pg_terminate_backend`, à la différence qu'elle essaie d'interrompre tous les processus dorsaux liés à l'ID de session distribuée spécifié en leur envoyant un `SIGTERM` (signal de fin).

Le paramètre d'entrée est le suivant :

- `distributed_session_id` (texte) : l'ID de la session distribuée à terminer.

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.

- `pid` (texte) : l'ID du processus dorsal.
- `success` (booléen) : indique si le processus s'est bien terminé.

L'exemple suivant montre comment utiliser la fonction `limitless_terminate_session`.

```
SELECT * FROM rds_aurora.limitless_terminate_session('940CD5C81E3C796B');

 subcluster_id |  pid  | success
---------------+-------+---------
             1 | 26920 | t
(1 row)
```

limitless_wait_report

La fonction `limitless_wait_report` renvoie l'activité des événements d'attente sur une période donnée, en provenance de l'ensemble des nœuds. Les colonnes `subcluster_id` et `subcluster_type` indiquent de quel nœud proviennent les données.

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.
- `subcluster_type` (texte) : le type de sous-cluster auquel appartient ce processus : `router` ou `shard`.

Les autres colonnes sont les mêmes que dans `aurora_wait_report`.

L'exemple suivant montre comment utiliser la fonction `limitless_wait_report`.

```
postgres_limitless=> select * from rds_aurora.limitless_wait_report();

 subcluster_id | subcluster_type | type_name | event_name | waits | wait_time |
 ms_per_wait | waits_per_xact | ms_per_xact
---------------+-----------------+-----------+------------+-------+-----------
+-------------+--------------+ +-------------
             1 | router          | Client    | ClientRead |    57 | 741550.14 |
 13009.652 |           0.19 |    2505.237
             5 | shard           | Client    | ClientRead |    54 | 738897.68 |
 13683.290 |           0.18 |    2496.276
             4 | shard           | Client    | ClientRead |    54 | 738859.53 |
 13682.584 |           0.18 |    2496.147
             2 | router          | Client    | ClientRead |    53 | 719223.64 |
 13570.257 |           0.18 |    2429.810
```

```
             3 | shard           | Client    | ClientRead |    54 | 461720.40 |
8550.378 |           0.18 |    1559.86
```

Vues d’Aurora PostgreSQL Limitless Database

Le tableau suivant présente les nouvelles vues d’Aurora PostgreSQL Limitless Database.

> **Note**
>
> Les vues répertoriées dans ce tableau se trouvent dans le schéma `rds_aurora`. Lorsque vous utilisez une vue Limitless Database, assurez-vous d’inclure le nom d’objet entièrement qualifié : `rds_aurora.`*`object_name`*.

Vue d’Aurora PostgreSQL Limitless Database	Vue d’Aurora PostgreSQL correspondante
limitless_database	pg_database
limitless_locks	pg_locks
limitless_stat_activity	pg_stat_activity
limitless_stat_all_indexes	pg_stat_all_indexes
limitless_stat_all_tables	pg_stat_all_tables
limitless_stat_database	pg_stat_database
limitless_stat_progress_vacuum	pg_stat_progress_vacuum
limitless_stat_statements	pg_stat_statements
limitless_stat_subclusters	Aucun
limitless_stat_statements_info	pg_stat_statements_info
limitless_statio_all_indexes	pg_statio_all_indexes
limitless_statio_all_tables	pg_statio_all_tables
limitless_tables	pg_tables
limitless_table_collocations	Aucun

Vue d'Aurora PostgreSQL Limitless Database	Vue d'Aurora PostgreSQL correspondante
limitless_table_collocation_distributions	Aucun

Les exemples suivants fournissent des détails sur les vues d'Aurora PostgreSQL Limitless Database. Pour plus d'informations sur les vues de PostgreSQL, consultez Affichage des statistiques dans la documentation PostgreSQL.

> **Note**
>
> Certaines vues de statistiques peuvent produire des résultats incohérents si des transactions sont en cours d'exécution.

limitless_database

Cette vue contient des informations sur les bases de données disponibles dans le groupe de partitions de base de données. Exemples :

```
postgres_limitless=> SELECT subcluster_id, subcluster_type, oid, datname, datacl
 FROM rds_aurora.limitless_database;

 subcluster_id | subcluster_type |  oid  |      datname       |
                                 datacl

---------------+-----------------+-------+--------------------
+-------------------------------------------------------------------------------------------
 2             | router          |     4 | template0          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
 2             | router          |     5 | postgres           |
 2             | router          | 16384 | rdsadmin           |
 {rdsadmin=CTc/rdsadmin,rds_aurora_limitless_metadata_admin=c/
rdsadmin,rds_aurora_limitless_heat_mgmt_admin=c/rdsadmin}
 2             | router          | 16477 | postgres_limitless |
 2             | router          |     1 | template1          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
 6             | shard           |     4 | template0          | {=c/
rdsadmin,rdsadmin=CTc/rdsadmin}
```

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster (nœud)
- `subcluster_type` (texte) : le type de sous-cluster (nœud), de routeur ou de partition

Les autres colonnes sont les mêmes que dans `pg_database`.

limitless_locks

Cette vue contient une ligne par processus et par nœud. Elle fournit un accès aux informations sur les verrous détenus par les processus actifs du serveur de base de données.

Example de créer un verrou avec deux transactions

Dans cet exemple, nous exécutons deux transactions simultanément sur deux routeurs.

```
# Transaction 1 (run on router 1)
BEGIN;
SET search_path = public;
SELECT * FROM customers;
INSERT INTO customers VALUES (400,'foo','bar');

# Transaction 2 (run on router 2)
BEGIN;
SET search_path = public;
ALTER TABLE customers ADD COLUMN phone VARCHAR;
```

La première transaction est exécutée. Les transactions suivantes doivent attendre que la première transaction soit terminée. La deuxième transaction est donc bloquée par un verrou. Pour en vérifier la cause racine, nous exécutons une commande en joignant `limitless_locks` à `limitless_stat_activity`.

```
# Run on router 2
SELECT distributed_session_id, state, usename, query, query_start
FROM rds_aurora.limitless_stat_activity
WHERE distributed_session_id in (
SELECT distributed_session_id
FROM rds_aurora.limitless_locks
WHERE relname = 'customers'
);

 distributed_session_id | state               | usename                | query
                                  | query_start
------------------------+---------------------+-------------------------
+---------------------------------- -------------+-------------------------------
```

```
 47BDE66E9A5E8477       | idle in transaction | limitless_metadata_admin | INSERT INTO
 customers VALUES (400,'foo','bar'); | 2023-04-13 17:44:45.152244+00
 2AD7F370202D0FA9       | active              | limitless_metadata_admin | ALTER TABLE
 customers ADD COLUMN phone VARCHAR; | 2023-04-13 17:44:55.113388+00
 47BDE66E9A5E8477       |                     | limitless_auth_admin     |
 <insufficient privilege>                        |
 2AD7F370202D0FA9       |                     | limitless_auth_admin     |
 <insufficient privilege>                        |
 47BDE66E9A5E8477       |                     | limitless_auth_admin     |
 <insufficient privilege>                        |
 2AD7F370202D0FA9       |                     | limitless_auth_admin     |
 <insufficient privilege>                        |
(6 rows)
```

Example de créer un verrou de manière explicite

Dans cet exemple, nous créons un verrou de manière explicite, puis nous utilisons la vue `limitless_locks` pour afficher les verrous (certaines colonnes sont omises).

```
BEGIN;
SET search_path = public;
LOCK TABLE customers IN ACCESS SHARE MODE;
SELECT * FROM rds_aurora.limitless_locks WHERE relname = 'customers';

 subcluster_id | subcluster_type | distributed_session_id | locktype |      datname
   | relnspname |  relname  | virtualtransaction |  pid  |      mode
---------------+-----------------+------------------------+----------
+--------------------+------------+ ----------+--------------------+-------
+-----------------
             1 | router          | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 28/600787          | 59564 |
 AccessShareLock
             2 | router          | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 28/600405          | 67130 |
 AccessShareLock
             3 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 15/473401          | 27735 |
 AccessShareLock
             4 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/473524          | 27734 |
 AccessShareLock
```

```
             5 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/472935          | 27737 |
 AccessShareLock
             6 | shard           | 7207702F862FC937       | relation |
 postgres_limitless | public     | customers | 13/473015          | 48660 |
 AccessShareLock
(6 rows)
```

limitless_stat_activity

Cette vue contient une ligne par processus et par nœud. Elle peut être utilisée pour suivre l'état général du système et les processus de triage qui prennent du temps. Exemples :

```
postgres=# SELECT
    subcluster_id,
    subcluster_type,
    distributed_session_id,
    distributed_session_state,
    datname,
    distributed_query_id,
    is_sso_query
FROM
    rds_aurora.limitless_stat_activity
WHERE
    distributed_session_id in ('D2470C97E3D07E06', '5A3CD7B8E5FD13FF')
    order by  distributed_session_id;

 subcluster_id | subcluster_type | distributed_session_id | distributed_session_state |
      datname       | distributed_query_id | is_sso_query
---------------+-----------------+------------------------+---------------------------
+--------------------+----------------------+-------------
 2             | router          | 5A3CD7B8E5FD13FF       | coordinator               |
 postgres_limitless |                      | f
 3             | shard           | 5A3CD7B8E5FD13FF       | participant               |
 postgres_limitless |  6808291725541680947 |
 4             | shard           | 5A3CD7B8E5FD13FF       | participant               |
 postgres_limitless |  6808291725541680947 |
 2             | router          | D2470C97E3D07E06       | coordinator               |
 postgres_limitless |                      | t
 3             | shard           | D2470C97E3D07E06       | participant               |
 postgres_limitless |  4058400544464210222 |
(5 rows)
```

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.
- `subcluster_type` (texte) : le type de sous-cluster auquel appartient ce processus : `router` ou `shard`.
- `distributed_session_id` (texte) : l'ID de la session distribuée à laquelle appartient ce processus.
- `distributed_session_state` (texte) : précise si le processus est un coordinateur, un participant ou un processus indépendant/non distribué (indiqué comme `NULL`).
- `datname` (texte) : la base de données à laquelle ce processus est connecté.
- `distributed_query_id` (bigint) : l'ID de requête de la requête parent provenant du nœud coordinateur. Cette colonne est `NULL` s'il s'agit de la requête parent. Le nœud coordinateur transmet l'ID de requête distribué aux nœuds participants. Ainsi, pour les nœuds participants, les valeurs de l'ID de requête distribuée et de l'ID de requête sont différentes.
- `is_sso_query` (texte) : indique si la requête est optimisée pour une seule partition ou non.

Les autres colonnes sont les mêmes que dans `pg_stat_activity`.

limitless_stat_all_indexes

Cette vue contient des statistiques d'utilisation sur les index du groupe de partitions de base de données. Exemples :

```
postgres_limitless=> SELECT schemaname, relname, indexrelname, idx_scan
  FROM rds_aurora.limitless_stat_all_indexes
  WHERE relname LIKE 'orders_ts%' ORDER BY indexrelname LIMIT 10;

 schemaname |    relname     |    indexrelname     | idx_scan
------------+----------------+---------------------+----------
 ec_sample  | orders_ts00001 | orders_ts00001_pkey |   196801
 ec_sample  | orders_ts00002 | orders_ts00002_pkey |   196703
 ec_sample  | orders_ts00003 | orders_ts00003_pkey |   196376
 ec_sample  | orders_ts00004 | orders_ts00004_pkey |   197966
 ec_sample  | orders_ts00005 | orders_ts00005_pkey |   195301
 ec_sample  | orders_ts00006 | orders_ts00006_pkey |   195673
 ec_sample  | orders_ts00007 | orders_ts00007_pkey |   194475
 ec_sample  | orders_ts00008 | orders_ts00008_pkey |   191694
 ec_sample  | orders_ts00009 | orders_ts00009_pkey |   193744
 ec_sample  | orders_ts00010 | orders_ts00010_pkey |   195421
(10 rows)
```

limitless_stat_all_tables

Cette vue contient des statistiques sur toutes les tables de la base de données actuelle du groupe de partitions de base de données. Elle est particulièrement utile pour le suivi des processus de vacuum et des opérations de manipulation de données (DML). Exemples :

```
postgres_limitless=> SELECT subcluster_id, subcluster_type, relname,
 n_ins_since_vacuum, n_tup_ins, last_vacuum
  FROM rds_aurora.limitless_stat_all_tables
  WHERE relname LIKE 'orders_ts%' ORDER BY relname LIMIT 10;

 subcluster_id | subcluster_type |    relname     | n_ins_since_vacuum | n_tup_ins |
 last_vacuum
---------------+-----------------+----------------+--------------------+-----------
+-------------
 5             | shard           | orders_ts00001 |              34779 |    196083 |
 5             | shard           | orders_ts00002 |              34632 |    194721 |
 5             | shard           | orders_ts00003 |              34950 |    195965 |
 5             | shard           | orders_ts00004 |              34745 |    197283 |
 5             | shard           | orders_ts00005 |              34879 |    195754 |
 5             | shard           | orders_ts00006 |              34340 |    194605 |
 5             | shard           | orders_ts00007 |              33779 |    192203 |
 5             | shard           | orders_ts00008 |              33826 |    191293 |
 5             | shard           | orders_ts00009 |              34660 |    194117 |
 5             | shard           | orders_ts00010 |              34569 |    195560 |
(10 rows)
```

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.
- `subcluster_type` (texte) : le type de sous-cluster auquel appartient ce processus : `router` ou `shard`.
- `relname` (nom) : le nom de la table.

Les autres colonnes sont les mêmes que dans `pg_stat_all_tables`.

limitless_stat_database

Cette vue contient des statistiques sur toutes les tables du groupe de partitions de base de données. Renvoie une ligne par base de données et par nœud. Exemples :

```
postgres_limitless=> SELECT
```

```
    subcluster_id,
    subcluster_type,
    datname,
    blks_read,
    blks_hit
FROM
    rds_aurora.limitless_stat_database
WHERE
    datname='postgres_limitless';
 subcluster_id | subcluster_type |      datname       | blks_read | blks_hit
---------------+-----------------+--------------------+-----------+----------
             1 | router          | postgres_limitless |       484 | 34371314
             2 | router          | postgres_limitless |       673 | 33859317
             3 | shard           | postgres_limitless |      1299 | 17749550
             4 | shard           | postgres_limitless |      1094 | 17492849
             5 | shard           | postgres_limitless |      1036 | 17485098
             6 | shard           | postgres_limitless |      1040 | 17437257
(6 rows)
```

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.
- `subcluster_type` (texte) : le type de sous-cluster auquel appartient ce processus : `router` ou `shard`.
- `datname` (nom) : le nom de la base de données.

Les autres colonnes sont les mêmes que dans `pg_stat_database`.

limitless_stat_progress_vacuum

Cette vue contient des informations sur les opérations de vacuum en cours. Exemples :

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_progress_vacuum;

-[ RECORD 1 ]----------+-----------------
subcluster_id          | 3
subcluster_type        | shard
distributed_session_id | A56D96E2A5C9F426
pid                    | 5270
datname                | postgres
nspname                | public
relname                | customer_ts2
phase                  | vacuuming heap
```

```
heap_blks_total        | 130500
heap_blks_scanned      | 100036
heap_blks_vacuumed     | 0
index_vacuum_count     | 0
max_dead_tuples        | 11184810
num_dead_tuples        | 0

-[ RECORD 2 ]----------+------------------
subcluster_id          | 3
subcluster_type        | shard
distributed_session_id | 56DF26A89EC23AB5
pid                    | 6854
datname                | postgres
nspname                | public
relname                | sales_ts1
phase                  | vacuuming heap
heap_blks_total        | 43058
heap_blks_scanned      | 24868
heap_blks_vacuumed     | 0
index_vacuum_count     | 0
max_dead_tuples        | 8569523
num_dead_tuples        | 0
```

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.
- `subcluster_type` (texte) : le type de sous-cluster auquel appartient ce processus : `router` ou `shard`.
- `distributed_session_id` (texte) : l'identifiant de la session qui a lancé l'opération de vacuum.
- `datname` (nom) : la base de données dans laquelle le vacuum est effectué.
- `nspname` (name) : le nom du schéma de la table en cours de vacuum. La valeur est `null` si la table en cours de vacuum ne se trouve pas dans la même base de données que celle à laquelle l'utilisateur est connecté.
- `relname` (nom) : le nom de la table en cours de vacuum. La valeur est `null` si la table en cours de vacuum ne se trouve pas dans la même base de données que celle à laquelle l'utilisateur est connecté.

Les autres colonnes sont les mêmes que dans `pg_stat_progress_vacuum`.

limitless_stat_statements

Cette vue permet de suivre la planification et d'exécuter les statistiques de toutes les instructions SQL exécutées sur tous les nœuds.

> **Note**
>
> Vous devez installer l'extension pg_stat_statements pour utiliser la vue `limitless_stat_statements`.
>
> ```
> -- CREATE EXTENSION must be run by a superuser
> CREATE EXTENSION pg_stat_statements;
>
> -- Verify that the extension is created on all nodes in the DB shard group
> SELECT distinct node_id
>     FROM rds_aurora.limitless_stat_statements
>     LIMIT 10;
> ```

L'exemple suivant illustre l'utilisation de la vue `limitless_stat_statements`.

```
postgres_limitless=> SELECT
 subcluster_id,
 subcluster_type,
 distributedqueryid,
 username,
 dbname,
 sso_calls
FROM
 rds_aurora.limitless_stat_statements;

 subcluster_id | subcluster_type |  distributedqueryid  |              username
       |       dbname       | sso_calls
---------------+-----------------+----------------------
+--------------------------------------+--------------------+-----------
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
```

```
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         0
 2             | router          |                      | postgres
       | postgres_limitless |         1
 3             | shard           | -7975178695405682176 | postgres
       | postgres_limitless |
[...]
```

Les paramètres de sortie sont les suivants :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.
- `subcluster_type` (texte) : le type de sous-cluster auquel appartient ce processus : `router` ou `shard`.
- `distributedqueryid` (bigint) : l'ID de requête de la requête parent provenant du nœud coordinateur. Cette colonne est NULL s'il s'agit de la requête parent. Le nœud coordinateur transmet l'ID de requête distribué aux nœuds participants. Ainsi, pour les nœuds participants, les valeurs de l'ID de requête distribuée et de l'ID de requête sont différentes.
- `username` (nom) : l'utilisateur qui a interrogé l'instruction.
- dbname (nom) : la base de données dans laquelle la requête a été exécutée.
- `sso_calls` (nom) : le nombre de fois où l'instruction a été optimisée pour une seule partition.

Les autres colonnes sont les mêmes que dans pg_stat_statements.

limitless_stat_statements_info

Cette vue contient les statistiques de la vue `limitless_stat_statements`. Chaque ligne contient les données de la vue pg_stat_statements_info pour chaque nœud. La colonne `subcluster_id` identifie chaque nœud.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_statements_info;

 subcluster_id | subcluster_type | dealloc |          stats_reset
---------------+-----------------+---------+-------------------------------
             1 | router          |       0 | 2023-06-30 21:22:09.524781+00
             2 | router          |       0 | 2023-06-30 21:21:40.834111+00
             3 | shard           |       0 | 2023-06-30 21:22:10.709942+00
             4 | shard           |       0 | 2023-06-30 21:22:10.740179+00
             5 | shard           |       0 | 2023-06-30 21:22:10.774282+00
             6 | shard           |       0 | 2023-06-30 21:22:10.808267+00
```

```
(6 rows)
```

Le paramètre de sortie est le suivant :

- `subcluster_id` (texte) : l'ID du sous-cluster auquel appartient ce processus.

Les autres colonnes sont les mêmes que dans pg_stat_statements_info.

limitless_stat_subclusters

Cette vue contient les statistiques du réseau entre les routeurs et les autres nœuds. Elle contient une ligne par paire composée d'un routeur et d'un autre nœud, par exemple :

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_stat_subclusters;

 orig_subcluster | orig_instance_az | dest_subcluster | dest_instance_az |
 latency_us |       latest_collection       | failed_requests | received_bytes |
 sent_bytes | same_az_requests | cross_az_requests |     stat_reset_timestamp
-----------------+------------------+-----------------+------------------
+------------+-------------------------------+-----------------+---------------
+------------+------------------+-------------------+------------------------------
 3               | us-west-2b       | 2               | us-west-2a       |
 847 | 2024-10-07 17:25:39.518617+00 |               0 |       35668633 |   92090171
 |                0 |           302787 | 2024-10-05 12:39:55.239675+00
 3               | us-west-2b       | 4               | us-west-2b       |
 419 | 2024-10-07 17:25:39.546376+00 |               0 |      101190464 |  248795719
 |           883478 |                 0 | 2024-10-05 12:39:55.231218+00
 3               | us-west-2b       | 5               | us-west-2c       |
    1396 | 2024-10-07 17:25:39.52122+00  |               0 |       72864849 |
 172086292 |                0 |           557726 | 2024-10-05 12:39:55.196412+00
 3               | us-west-2b       | 6               | us-west-2c       |
 729 | 2024-10-07 17:25:39.54828+00  |               0 |       35668584 |   92090171
 |                0 |           302787 | 2024-10-05 12:39:55.247334+00
 3               | us-west-2b       | 7               | us-west-2a       |
    1702 | 2024-10-07 17:25:39.545307+00 |               0 |       71699576 |
 171634844 |                0 |           556278 | 2024-10-05 12:39:52.715168+00
 2               | us-west-2a       | 3               | us-west-2b       |
 868 | 2024-10-07 17:25:40.293927+00 |               0 |       35659611 |   92011872
 |                0 |           302817 | 2024-10-05 12:39:54.420758+00
 2               | us-west-2a       | 4               | us-west-2b       |
 786 | 2024-10-07 17:25:40.296863+00 |               0 |      102437253 |  251838024
 |                0 |           895060 | 2024-10-05 12:39:54.404081+00
```

```
2               | us-west-2a       | 5               | us-west-2c       |
   1232 | 2024-10-07 17:25:40.292021+00 |               0 |       71990027 |
168828110 |               0 |           545453 | 2024-10-05 12:39:36.769549+00
```

Les paramètres de sortie sont les suivants :

- `orig_subcluster` (texte) : l'ID du routeur d'où proviennent les communications
- `orig_subcluster_az` (texte) : la zone de disponibilité (AZ) du routeur d'origine
- `dest_subcluster` (texte) : l'ID du nœud de destination
- `dest_subcluster_az` (texte) : la dernière zone de disponibilité collectée du nœud de destination
- `latency_us` (bigint) : la dernière latence réseau collectée entre les nœuds, en microsecondes. La valeur est `0` si le nœud est inaccessible.
- `latest_collection` (horodatage) : l'horodatage de la dernière collecte des informations de zone de disponibilité et de latence pour le nœud de destination.
- `failed_requests` (bigint) : le nombre cumulé de demandes internes ayant échoué
- `received_bytes` (bigint) : le nombre cumulé estimé d'octets reçus de ce nœud
- `sent_bytes` (bigint) : le nombre cumulé estimé d'octets envoyés à ce nœud
- `same_az_requests` (bigint) : le nombre cumulé de demandes de base de données internes adressées à ce nœud lorsqu'il se trouve dans la même zone de disponibilité que le routeur d'origine
- `cross_az_requests` (bigint) : le nombre cumulé de demandes de base de données internes adressées à ce nœud lorsqu'il se trouve dans une zone de disponibilité différente de celle du routeur d'origine
- `stat_reset_timestamp` (horodatage) : l'horodatage auquel les statistiques cumulées pour cette vue ont été réinitialisées pour la dernière fois

limitless_statio_all_indexes

Cette vue contient les statistiques d'entrée/sortie (E/S) pour tous les index du groupe de partitions de base de données. Exemples :

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_statio_all_indexes WHERE
 relname like'customers_ts%';

 subcluster_id | subcluster_type | schemaname |      relname      |
 indexrelname             | idx_blks_read | idx_blks_hit
```

```
--------------+-----------------+------------+------------------
+-------------------------------------+ --------------+-------------
             3 | shard           | public     | customers_ts00002 |
 customers_ts00002_customer_name_idx |             1 |           0
             3 | shard           | public     | customers_ts00001 |
 customers_ts00001_customer_name_idx |             1 |           0
             4 | shard           | public     | customers_ts00003 |
 customers_ts00003_customer_name_idx |             1 |           0
             4 | shard           | public     | customers_ts00004 |
 customers_ts00004_customer_name_idx |             1 |           0
             5 | shard           | public     | customers_ts00005 |
 customers_ts00005_customer_name_idx |             1 |           0
             5 | shard           | public     | customers_ts00006 |
 customers_ts00006_customer_name_idx |             1 |           0
             6 | shard           | public     | customers_ts00007 |
 customers_ts00007_customer_name_idx |             1 |           0
             6 | shard           | public     | customers_ts00008 |
 customers_ts00008_customer_name_idx |             1 |           0
(8 rows)
```

limitless_statio_all_tables

Cette vue contient les statistiques d'entrée/sortie (E/S) pour toutes les tables du groupe de partitions de base de données. Exemples :

```
postgres_limitless=> SELECT
    subcluster_id,
    subcluster_type,
    schemaname,
    relname,
    heap_blks_read,
    heap_blks_hit
FROM
    rds_aurora.limitless_statio_all_tables
WHERE
    relname LIKE 'customers_ts%';

 subcluster_id | subcluster_type | schemaname |      relname      | heap_blks_read |
 heap_blks_hit
---------------+-----------------+------------+-------------------+----------------
+---------------
             3 | shard           | public     | customers_ts00002 |            305 |
      57780
```

```
             3 | shard           | public     | customers_ts00001 |            300 |
      56972
             4 | shard           | public     | customers_ts00004 |            302 |
      57291
             4 | shard           | public     | customers_ts00003 |            302 |
      57178
             5 | shard           | public     | customers_ts00006 |            300 |
      56932
             5 | shard           | public     | customers_ts00005 |            302 |
      57386
             6 | shard           | public     | customers_ts00008 |            300 |
      56881
             6 | shard           | public     | customers_ts00007 |            304 |
      57635
(8 rows)
```

limitless_tables

Cette vue contient des informations sur les tables d'Aurora PostgreSQL Limitless Database.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_tables;

 table_gid | local_oid | schema_name | table_name  | table_status | table_type  |
 distribution_key
-----------+-----------+-------------+-------------+--------------+-------------
+------------------
         5 |     18635 | public      | placeholder | active       | placeholder |
         6 |     18641 | public      | ref         | active       | reference   |
         7 |     18797 | public      | orders      | active       | sharded     | HASH
 (order_id)
         2 |     18579 | public      | customer    | active       | sharded     | HASH
 (cust_id)
(4 rows)
```

limitless_table_collocations

Cette vue contient des informations sur les tables partitionnées colocalisées.

Dans l'exemple suivant, les tables `orders` et `customers` sont colocalisées, tout comme les tables `users` et `followers`. Les tables colocalisées présentent le même `collocation_id`.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_table_collocations ORDER BY
 collocation_id;
```

```
 collocation_id | schema_name | table_name
----------------+-------------+------------
              2 | public      | orders
              2 | public      | customers
              5 | public      | users
              5 | public      | followers
(4 rows)
```

limitless_table_collocation_distributions

Cette vue affiche la diffusion des clés pour chaque collocation.

```
postgres_limitless=> SELECT * FROM
 rds_aurora.limitless_table_collocation_distributions ORDER BY collocation_id,
 lower_bound;

 collocation_id | subcluster_id |     lower_bound      |     upper_bound
----------------+---------------+----------------------+----------------------
              2 |             6 | -9223372036854775808 | -4611686018427387904
              2 |             5 | -4611686018427387904 |                    0
              2 |             4 |                    0 |  4611686018427387904
              2 |             3 |  4611686018427387904 |  9223372036854775807
              5 |             6 | -9223372036854775808 | -4611686018427387904
              5 |             5 | -4611686018427387904 |                    0
              5 |             4 |                    0 |  4611686018427387904
              5 |             3 |  4611686018427387904 |  9223372036854775807
(8 rows)
```

Événements d’attente dans Aurora PostgreSQL Limitless Database

Un événement d’attente dans Aurora PostgreSQL indique une ressource pour laquelle une session est en attente, telle qu’une entrée/sortie (E/S) et des verrous. Les événements d’attente permettent de déterminer les raisons pour lesquelles les sessions attendent des ressources et d’identifier les goulots d’étranglement. Pour plus d’informations, consultez Événements d'attente Aurora PostgreSQL.

Aurora PostgreSQL Limitless Database possède ses propres événements d’attente liés aux routeurs et aux partitions. La plupart d’entre eux concernent les routeurs qui attendent que des partitions soient disponibles pour exécuter leurs tâches. Les événements d’attente de partition contiennent des détails sur les tâches en cours d’exécution.

Rubriques

Interrogation des événements d’attente

Vous pouvez utiliser la vue limitless_stat_activity pour interroger des événements d’attente, comme illustré dans l’exemple suivant.

```
SELECT wait_event FROM rds_aurora.limitless_stat_activity WHERE
 wait_event_type='AuroraLimitless';

      wait_event
----------------------
 RemoteStatementSetup
 RemoteStatementSetup
(2 rows)
```

Vous pouvez également utiliser la fonction `aurora_stat_system_waits` pour répertorier le nombre d’attentes et le temps total consacré à chaque événement d’attente, comme indiqué dans l’exemple suivant.

```
postgres_limitless=> SELECT type_name,event_name,waits,wait_time
    FROM aurora_stat_system_waits()
    NATURAL JOIN aurora_stat_wait_event()
    NATURAL JOIN aurora_stat_wait_type()
```

```
    WHERE type_name='AuroraLimitless'
    ORDER BY wait_time DESC;

    type_name    |       event_name          |  waits  |  wait_time
-----------------+---------------------------+---------+-------------
 AuroraLimitless | RemoteStatementSetup      |    7518 | 75236507897
 AuroraLimitless | RemoteStatementExecution  |      40 |      132986
 AuroraLimitless | Connect                   |       5 |        1453
(3 rows)
```

Événements d’attente Limitless Database

Les événements d’attente suivants sont propres à Aurora PostgreSQL Limitless Database. Vous pouvez surveiller ces événements d’attente pour identifier les goulots d’étranglement dans le traitement au sein d’Aurora PostgreSQL Limitless Database.

Rubriques

- Événement d’attente IO:TwophaseFilePoolWrite
- Événement d’attente IO:TwophaseFilePoolRead
- Événement d’attente AuroraLimitless:Connect
- Événement d’attente AuroraLimitless:AsyncConnect
- Événement d’attente AuroraLimitless:RemoteStatementSetup
- Événement d’attente AuroraLimitless:RemoteDDLExecution
- Événement d’attente AuroraLimitless:RemoteStatementExecution
- Événement d’attente AuroraLimitless:FetchRemoteResults
- Événement d’attente AuroraLimitless:AsyncGetInitialResponse
- Événement d’attente AuroraLimitless:AsyncGetNextResponse
- Événement d’attente AuroraLimitless:AbortedCommandCleanup
- Événement d’attente AuroraLimitless:DistributedCommitPrepare
- Événement d’attente AuroraLimitless:DistributedCommit
- Événement d’attente AuroraLimitless:DistributedCommitPrepareThrottle
- Événement d’attente AuroraLimitless:PreparedTransactionResolution
- Événement d’attente AuroraLimitless:SendPreparedTransactionOutcome
- Événement d’attente AuroraLimitless:CommitClockBarrier
- Événement d’attente AuroraLimitless:SnapshotClockBarrier

- Événement d’attente AuroraLimitless:ReaderSnapshotClockBarrier
- Événement d’attente AuroraLimitless:GatherDistributedDeadlockGraph
- Événement d’attente AuroraLimitless:DistributedDeadlockDetection
- Événement d’attente AuroraLimitless:DistributedDeadlockAbort
- Événement d’attente AuroraLimitless:GatherRemoteStats
- Événement d’attente AuroraLimitless:GlobalSequenceRefresh
- Événement d’attente AuroraLimitless:GlobalVacuumTimeExchange
- Événement d’attente AuroraLimitless:DistributedTransactionMonitorGather
- Événement d’attente AuroraLimitlessActivity:AdminTaskSchedulerMain
- Événement d’attente AuroraLimitlessActivity:AdminTaskExecutorMain
- Événement d’attente AuroraLimitlessActivity:AdminTaskMonitorMain
- Événement d’attente AuroraLimitlessActivity:DatabaseCleanupMonitorMain
- Événement d’attente AuroraLimitlessActivity:TopologyCleanupMonitorMain
- Événement d’attente AuroraLimitlessActivity:ToplogyChangeMonitorMain
- Événement d’attente AuroraLimitlessActivity:DistributedTransactionMonitorMain
- Événement d’attente AuroraLimitlessActivity:GlobalVacuumMonitorMain

Événement d’attente IO:TwophaseFilePoolWrite

En attente de l’écriture d’un fichier d’état à deux phases dans le pool de fichiers d’état à deux phases. Il s’agit d’un événement spécifique à Aurora.

Causes

Les processus exécutant une commande `PREPARED TRANSACTION`, y compris les participants à une transaction distribuée Limitless Database, doivent conserver l’état de la transaction dans un fichier en deux phases. Aurora utilise un pool de fichiers pour améliorer les performances de cette opération.

Action

Cette opération correspond à une écriture d’E/S synchrone ; ainsi, une latence importante sur cet événement résulte des mêmes causes que pour `IO:XactSync` et se traite de façon identique. Si vous utilisez Limitless Database, vous devrez peut-être réduire le nombre de transactions distribuées exécutées.

Événement d’attente IO:TwophaseFilePoolRead

En attente de la lecture d’un fichier d’état à deux phases dans le pool de fichiers d’état à deux phases.

Causes

Les processus qui exécutent une commande `COMMIT PREPARED` sur une transaction déjà préparée, y compris les participants d’une transaction distribuée Limitless Database, peuvent être amenés à lire l’état de transaction précédemment enregistré dans un fichier d’état à deux phases. Aurora utilise un pool de fichiers pour améliorer les performances de cette opération.

Action

Il s’agit d’une opération d’E/S. Par conséquent, une latence importante sur cet événement résulte des mêmes causes que pour `IO:DataFileRead` et se traite de façon identique. Si vous utilisez Limitless Database, vous devrez peut-être réduire le nombre de transactions distribuées exécutées.

Événement d’attente AuroraLimitless:Connect

Le processus attend qu’une connexion soit établie avec un autre nœud du cluster.

Causes

Des connexions sont établies entre les processus et les nœuds distants pour exécuter des requêtes, des transactions distribuées et des DDL.

Action

Réduisez le nombre de connexions simultanées au cluster ou optimisez l’utilisation des requêtes inter-partitions.

Événement d’attente AuroraLimitless:AsyncConnect

Cet événement, semblable à `Connect`, représente l’attente d’un processus pendant l’établissement de connexions parallèles avec plusieurs nœuds.

Causes

L’établissement de connexions parallèles est le plus souvent effectué lors de l’exécution d’instructions DDL.

Action

Réduisez le nombre d'instructions DDL ou regroupez-en plusieurs au sein d'une même session afin d'améliorer la réutilisation des connexions.

Événement d'attente AuroraLimitless:RemoteStatementSetup

Le processus est en attente de la configuration de l'exécution d'une requête distante, telle que l'ouverture ou la fermeture d'un curseur, ou la création d'une instruction préparée.

Causes

Cet événement d'attente augmente avec le nombre d'analyses effectuées sur des tables partitionnées lorsque l'instruction ne peut pas être optimisée pour une seule partition.

Action

Optimisez les requêtes pour réduire le nombre d'opérations d'analyse ou augmenter l'éligibilité à l'optimisation à partition unique.

Événement d'attente AuroraLimitless:RemoteDDLExecution

Le processus attend la fin d'une commande DDL à distance.

Causes

Lorsque vous exécutez une commande DDL sur un groupe de partitions de base de données, celle-ci doit être distribuée aux autres routeurs et nœuds de partition avant la validation de l'opération. Certaines opérations DDL peuvent s'exécuter pendant une longue période, car les données doivent être adaptées aux modifications du schéma.

Action

Identifiez les commandes DDL de longue durée afin de les optimiser.

Événement d'attente AuroraLimitless:RemoteStatementExecution

Un processus attend la fin d'une commande à distance.

Causes

Une commande SQL est en cours d'exécution sur un nœud distant. Cet événement apparaîtra fréquemment dans le cadre des communications internes, telles que `auto_analyze` et les vérifications de pulsations.

Action

Identifiez les commandes de longue durée à l'aide de la vue limitless_stat_statements. Dans de nombreux cas, cet événement est attendu, en particulier pour les processus en arrière-plan ou internes, et aucune action n'est requise.

Événement d'attente AuroraLimitless:FetchRemoteResults

Un processus attend de récupérer les lignes d'un nœud distant.

Causes

Cet événement d'attente tend à se produire davantage lorsqu'un grand volume de lignes est extrait d'une table distante, comme une table partitionnée ou une table de référence.

Action

Identifiez les requêtes `SELECT` non optimisées à l'aide de la vue `limitless_stat_statements`. Optimisez les requêtes pour récupérer uniquement les données nécessaires. Vous pouvez également ajuster le paramètre `rds_aurora.limitless_maximum_adaptive_fetch_size`.

Événement d'attente AuroraLimitless:AsyncGetInitialResponse

Le processus attend une réponse initiale lorsque le mode pipeline est utilisé pour l'exécution des requêtes.

Causes

Cet événement est généralement observé lors de l'exécution d'une requête entre le routeur et la partition lorsque les données sont situées sur une seule partition ; il s'agit d'un fonctionnement attendu.

Action

Aucune action supplémentaire n'est requise.

Événement d'attente AuroraLimitless:AsyncGetNextResponse

Le processus attend des réponses supplémentaires lorsque le mode pipeline est utilisé pour l'exécution des requêtes.

Causes

Cet événement est généralement observé lors de l’exécution d’une requête entre le routeur et la partition lorsque les données sont situées sur une seule partition ; il s’agit d’un fonctionnement attendu.

Action

Aucune action supplémentaire n’est requise.

Événement d’attente AuroraLimitless:AbortedCommandCleanup

Le processus attend le résultat d’une requête de nettoyage à distance. Des requêtes de nettoyage sont envoyées aux nœuds de partition afin de les ramener à un état approprié lorsque qu’une transaction distribuée se termine.

Causes

Le nettoyage d’une transaction est effectué lorsqu’une transaction est annulée soit parce qu’une erreur a été détectée, soit parce qu’un utilisateur a émis une commande ABORT explicite ou annulé la requête en cours d’exécution.

Action

Recherchez la cause de l’annulation de la transaction.

Événement d’attente AuroraLimitless:DistributedCommitPrepare

Le processus valide une transaction distribuée et attend que chaque participant confirme la commande de préparation.

Causes

Les transactions qui modifient plusieurs nœuds doivent effectuer une validation distribuée. Une durée d’attente prolongée pour `DistributedCommitPrepare` peut résulter d’attentes importantes sur l’événement `IO:TwophaseFilePoolWrite` des nœuds participants.

Action

Réduisez le nombre de transactions qui modifient les données sur plusieurs nœuds. Étudiez les événements `IO:TwophaseFilePoolWrite` survenus sur d’autres nœuds du cluster.

Événement d’attente AuroraLimitless:DistributedCommit

Le processus valide une transaction distribuée et attend que le participant principal confirme la commande de préparation.

Causes

Les transactions qui modifient plusieurs nœuds doivent effectuer une validation distribuée. Une durée d’attente prolongée pour `DistributedCommit` peut résulter d’attentes importantes sur l’événement `IO:XactSync` le participant principal.

Action

Réduisez le nombre de transactions qui modifient les données sur plusieurs nœuds. Étudiez les événements `IO:XactSync` survenus sur d’autres nœuds du cluster.

Événement d’attente AuroraLimitless:DistributedCommitPrepareThrottle

Le processus tente de préparer une transaction distribuée et est limité en raison de transactions préparées existantes.

Causes

Les transactions qui modifient plusieurs nœuds doivent effectuer une validation distribuée. Les participants à ces transactions doivent effectuer une opération de préparation dans le cadre du protocole de validation. Aurora limite le nombre de préparations simultanées ; si cette limite est dépassée, le processus attendra lors de l’événement `DistributedCommitPrepareThrottle`.

Action

Réduisez le nombre de transactions qui modifient les données sur plusieurs nœuds. Examinez les événements `IO:TwophaseFilePoolWrite`, car une augmentation de leur durée pourrait entraîner une accumulation de transactions préparées existantes, ce qui ralentirait les nouvelles tentatives de préparation.

Événement d’attente AuroraLimitless:PreparedTransactionResolution

Le processus a rencontré un tuple modifié par une transaction distribuée qui est à l’état préparé. Le processus doit déterminer si la transaction distribuée sera visible dans son instantané.

Causes

Les transactions qui modifient plusieurs nœuds doivent effectuer une validation distribuée comprenant une phase de préparation. Un nombre élevé de transactions distribuées ou une latence accrue lors des validations distribuées peuvent provoquer la survenue de l'événement d'attente `PreparedTransactionResolution` dans d'autres processus.

Action

Réduisez le nombre de transactions qui modifient les données sur plusieurs nœuds. Examinez les événements liés aux validations distribuées, car un allongement de leur durée peut augmenter la latence du processus de validation des transactions distribuées. Il peut aussi être utile d'analyser les charges du réseau et du CPU.

Événement d'attente AuroraLimitless:SendPreparedTransactionOutcome

Le processus s'exécute sur un nœud qui coordonne une transaction distribuée et un autre processus s'est renseigné sur l'état de cette transaction, ou le processus a validé une transaction distribuée et envoie le résultat aux participants.

Causes

Les processus confrontés à l'événement d'attente `PreparedTransactionResolution` interrogeront le coordinateur des transactions. La réponse sur le coordinateur de transactions rencontrera l'événement SendPreparedTransactionOutcome.

Action

Réduisez le nombre de transactions qui modifient les données sur plusieurs nœuds. Examinez les événements liés aux validations distribuées, ainsi que les événements `IO:TwophaseFilePoolWrite` et `IO:TwophaseFilePoolRead`, car ils peuvent augmenter la latence du processus de validation des transactions distribuées. Il peut aussi être utile d'analyser les charges du réseau et du CPU.

Événement d'attente AuroraLimitless:CommitClockBarrier

Le processus procède à la validation d'une transaction et doit attendre que le temps de validation attribué soit assurément passé pour tous les nœuds du cluster.

Causes

Une saturation du CPU ou du réseau peut accentuer la dérive de l'horloge, entraînant ainsi une perte de temps au cours de cet événement d'attente.

Action

Examinez la saturation de l’UC ou du réseau dans votre cluster.

Événement d’attente AuroraLimitless:SnapshotClockBarrier

Le processus a reçu d’un autre nœud une heure d’instantané dont l’horloge est en avance, et il attend que sa propre horloge atteigne cette heure.

Causes

Cela se produit généralement lorsque le processus a reçu les résultats d’une fonction ayant été déléguée à une partition, et qu’il existe une dérive d’horloge entre les nœuds. Une saturation du CPU ou du réseau peut accentuer la dérive de l’horloge, entraînant ainsi une perte de temps au cours de cet événement d’attente.

Action

Examinez la saturation de l’UC ou du réseau dans votre cluster.

Événement d’attente AuroraLimitless:ReaderSnapshotClockBarrier

Cet événement se produit sur les nœuds de lecture. Le processus attend que le nœud de lecture rejoue le flux d’écriture afin que toutes les écritures effectuées avant l’heure d’instantané du processus aient été appliquées.

Causes

Une augmentation du retard de réplica Aurora peut entraîner une hausse du temps d’attente pour cet événement.

Action

Examinez le retard de réplica Aurora.

Événement d’attente AuroraLimitless:GatherDistributedDeadlockGraph

Le processus communique avec les autres nœuds afin de collecter les graphes de verrous dans le cadre de la détection de blocages distribués.

Causes

Lorsqu’un processus attend un verrou, il effectue une vérification des blocages distribués après une attente supérieure à `rds_aurora.limitless_distributed_deadlock_timeout`.

Action

Examinez les causes de la contention sur les verrous dans votre application et envisagez d'ajuster le paramètre `rds_aurora.limitless_distributed_deadlock_timeout`.

Événement d'attente AuroraLimitless:DistributedDeadlockDetection

Le processus communique avec les autres nœuds afin de détecter un blocage distribué.

Causes

Lorsqu'un processus attend un verrou, il effectue une vérification des blocages distribués après une attente supérieure à `rds_aurora.limitless_distributed_deadlock_timeout`.

Action

Examinez les causes de la contention sur les verrous dans votre application et envisagez d'ajuster le paramètre `rds_aurora.limitless_distributed_deadlock_timeout`.

Événement d'attente AuroraLimitless:DistributedDeadlockAbort

Le processus communique avec un autre nœud afin d'interrompre une session choisie comme victime d'un interblocage distribué.

Causes

Les modèles d'application entraînent des blocages distribués.

Action

Examinez les modèles d'application entraînant des blocages distribués.

Événement d'attente AuroraLimitless:GatherRemoteStats

Le processus collecte des statistiques à partir des autres nœuds du cluster.

Causes

Les requêtes et les vues de surveillance ou d'activité, par exemple`limitless_stat_activity`, permettront de récupérer les statistiques provenant d'autres nœuds.

Action

Aucune action supplémentaire n'est requise.

Événement d’attente AuroraLimitless:GlobalSequenceRefresh

Le processus génère une nouvelle valeur de séquence et doit demander un nouveau fragment à partir de la séquence globale.

Causes

Un taux élevé de génération de valeurs de séquence peut entraîner des blocages lors de cet événement si le paramètre `rds_aurora.limitless_sequence_chunk_size` est insuffisant.

Action

Il s’agit d’un phénomène normal. Si vous constatez que cet événement dure trop longtemps, ajustez `rds_aurora.limitless_sequence_chunk_size`. Consultez la documentation sur les séquences dans Limitless Database.

Événement d’attente AuroraLimitless:GlobalVacuumTimeExchange

Le processus échange des données d’instantané afin de prendre en charge l’opération de vacuum.

Causes

Les nœuds de Limitless Database échangent les données relatives à l’heure du plus ancien instantané actif avec d’autres nœuds, afin de calculer le moment de coupure correct pour l’exécution du vacuum.

Action

Aucune action supplémentaire n’est requise.

Événement d’attente AuroraLimitless:DistributedTransactionMonitorGather

Le processus collecte les métadonnées des transactions à partir d’autres nœuds pour faciliter le nettoyage des transactions distribuées.

Causes

Les nœuds de Limitless Database échangent les métadonnées de transaction avec d’autres nœuds afin de déterminer à quel moment l’état des transactions distribuées peut être purgé.

Action

Aucune action supplémentaire n’est requise.

Événement d’attente AuroraLimitlessActivity:AdminTaskSchedulerMain

En attente dans la boucle principale du processus du planificateur de tâches.

Événement d’attente AuroraLimitlessActivity:AdminTaskExecutorMain

En attente dans la boucle principale du processus d’exécuteur de tâches.

Événement d’attente AuroraLimitlessActivity:AdminTaskMonitorMain

En attente dans la boucle principale du processus de surveillance de tâches.

Événement d’attente AuroraLimitlessActivity:DatabaseCleanupMonitorMain

En attente dans la boucle principale du processus de surveillance du nettoyage de la base de données.

Événement d’attente AuroraLimitlessActivity:TopologyCleanupMonitorMain

En attente dans la boucle principale du processus de surveillance du nettoyage de la topologie.

Événement d’attente AuroraLimitlessActivity:ToplogyChangeMonitorMain

En attente dans la boucle principale du processus de surveillance des changements de topologie.

Événement d’attente AuroraLimitlessActivity:DistributedTransactionMonitorMain

En attente dans la boucle principale du processus de surveillance des transactions distribuées.

Événement d’attente AuroraLimitlessActivity:GlobalVacuumMonitorMain

En attente dans la boucle principale du processus de surveillance du vacuum global.

Sauvegarde et restauration d’Aurora PostgreSQL Limitless Database

Vous pouvez sauvegarder et restaurer des clusters de bases de données qui utilisent Aurora PostgreSQL Limitless Database.

Table des matières

Sauvegarde d’un cluster de bases de données utilisant Aurora PostgreSQL Limitless Database

La sauvegarde d’un cluster de bases de données avec Aurora PostgreSQL Limitless Database présente à la fois des similitudes et des différences fonctionnelles par rapport à la sauvegarde d’un cluster de bases de données Aurora standard.

- Si vous créez un instantané manuel d’un cluster de bases de données Aurora utilisant Limitless Database, celui-ci comprend les données du groupe de partitions de la base de données.
- Les sauvegardes continues incluent les données du groupe de partitions de base de données.
- Les instantanés quotidiens automatisés incluent les données du groupe de partitions de base de données.
- La copie des instantanés de cluster de bases de données est prise en charge. Pour plus d’informations, consultez Copie d’un instantané de cluster de bases de données.
- Le partage des instantanés de cluster de bases de données est pris en charge. Pour plus d’informations, consultez Partage d’un instantané de cluster de bases de données.
- Vous ne pouvez pas utiliser l’utilitaire `pg_dump` ou `pg_dumpall` pour sauvegarder les bases de données du groupe de partitions de base de données.
- La fonctionnalité de prise d’instantané final lors de la suppression de clusters de bases de données est disponible dans Aurora PostgreSQL Limitless Database.

- La conservation des sauvegardes automatiques lors de la suppression de clusters de bases de données n’est pas disponible dans Aurora PostgreSQL Limitless Database.

Création d’un instantané de cluster de bases de données

La procédure de création d’un instantané de cluster de bases de données Aurora PostgreSQL Limitless Database est identique à celle d’un cluster de bases de données Aurora standard, comme illustré dans l’exemple AWS CLI suivant :

```
aws rds create-db-cluster-snapshot \
    --db-cluster-identifier my-db-cluster \
    --db-cluster-snapshot-identifier my-db-cluster-snapshot
```

Pour plus d’informations sur la sauvegarde des clusters de bases de données, consultez Présentation de la sauvegarde et de la restauration d’un cluster de bases de données Aurora.

Restauration d’un cluster de bases de données utilisant Aurora PostgreSQL Limitless Database

La restauration d’un cluster de bases de données avec Aurora PostgreSQL Limitless Database présente à la fois des similitudes et des différences fonctionnelles par rapport à la sauvegarde d’un cluster de bases de données Aurora standard.

- Vous pouvez restaurer un cluster de bases de données Limitless Database uniquement à partir d’un cluster de bases de données source qui utilise une version de moteur de base de données compatible avec Limitless Database, telle que `16.4-limitless`.
- Lorsque vous restaurez un cluster de bases de données à partir d’un instantané manuel d’un cluster de bases de données qui utilise Limitless Database, l’intégralité du stockage de cluster de bases de données est restaurée. Cela comprend également le stockage du groupe de partitions de base de données.

  Vous devez créer un groupe de partitions de base de données pour accéder au stockage Limitless Database.
- Vous pouvez restaurer un cluster de bases de données à l’aide d’une reprise ponctuelle (PITR) à n’importe quel moment de la période de rétention. Le cluster de bases de données restauré comprend le stockage du groupe de partitions de base de données.

  Vous devez créer un groupe de partitions de base de données pour accéder au stockage Limitless Database.
- La reprise ponctuelle (PITR) n’est pas disponible pour les clusters de bases de données dans Aurora PostgreSQL Limitless Database.
- Lorsque vous restaurez un cluster de bases de données à partir d’un instantané quotidien automatisé, le stockage du groupe de partitions de base de données est également restauré.
- Lorsque vous restaurez un cluster de bases de données Aurora PostgreSQL Limitless Database, vous devez activer la Surveillance améliorée et Performance Insights. Veillez à inclure l’ID de clé KMS Performance Insights.

Après avoir restauré un cluster de bases de données Aurora PostgreSQL Limitless Database, vérifiez son bon fonctionnement en y exécutant vos requêtes.

Restauration d’un cluster de bases de données à partir d’un instantané de base de données.

Les exemples AWS CLI suivants expliquent comment restaurer un cluster de bases de données Aurora PostgreSQL Limitless Database à partir d’un instantané de cluster de bases de données.

Vous devez utiliser la version du moteur de base de données `16.4-limitless`.

Pour restaurer un cluster de bases de données Limitless Database à partir d’un instantané de cluster de bases de données

1. Restaurez le cluster de bases de données :

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier my-new-db-cluster \
    --snapshot-identifier my-db-cluster-snapshot \
    --engine aurora-postgresql \
    --engine-version 16.4-limitless \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --performance-insights-kms-key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole
```

2. Créez le groupe de partitions de base de données :

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-new-db-cluster \
    --db-shard-group-identifier my-new-DB-shard-group \
    --max-acu 1000
```

   Pour plus d’informations, consultez Ajout d’un groupe de partitions de base de données à un cluster de bases de données Aurora PostgreSQL Limitless Database existant.

Pour plus d’informations sur la restauration d’un cluster de bases de données Aurora à partir d’un instantané de cluster de bases de données, consultez Restauration à partir d’un instantané de cluster de bases de données.

Restauration d’un cluster de bases de données à l’aide de la reprise ponctuelle

Les exemples AWS CLI suivants expliquent comment restaurer un cluster de bases de données Aurora PostgreSQL Limitless Database à l’aide de la reprise ponctuelle (PITR).

Pour restaurer un cluster de bases de données Limitless Database à l’aide de la PITR

1. Restaurez le cluster de bases de données :

```
aws rds restore-db-cluster-to-point-in-time \
    --source-db-cluster-identifier my-db-cluster \
    --db-cluster-identifier my-new-db-cluster \
    --use-latest-restorable-time \
    --enable-performance-insights \
    --performance-insights-retention-period 31 \
    --performance-insights-kms-key-id arn:aws:kms:us-
east-1:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab \
    --monitoring-interval 5 \
    --monitoring-role-arn arn:aws:iam::123456789012:role/EMrole
```

2. Créez le groupe de partitions de base de données :

```
aws rds create-db-shard-group \
    --db-cluster-identifier my-new-db-cluster \
    --db-shard-group-identifier my-new-DB-shard-group \
    --max-acu 1000
```

   Pour plus d’informations, consultez Ajout d’un groupe de partitions de base de données à un cluster de bases de données Aurora PostgreSQL Limitless Database existant.

Pour plus d’informations sur la PITR, consultez Restauration d’un cluster de bases de données à une date définie.

Les utilitaires de sauvegarde et de restauration PostgreSQL ne sont pas pris en charge

Les utilitaires PostgreSQL suivants ne sont pris en charge ni pour le cluster de bases de données principal ni pour le groupe de partitions de base de données :

- `pg_dump`
- `pg_dumpall`
- `pg_restore`

Bien que vous puissiez les utiliser via des binaires open source ou d’autres méthodes, cela pourrait produire des résultats incohérents.

Mise à niveau d’Amazon Aurora PostgreSQL Limitless Database

Les considérations suivantes concernent la mise à niveau d’Aurora PostgreSQL Limitless Database :

- Les mises à niveau mineures sont prises en charge.
- L’application de correctifs à Aurora PostgreSQL Limitless Database est prise en charge. Les correctifs figurent parmi les tâches de maintenance en attente, qui seront appliquées pendant la fenêtre de maintenance.

Mise à niveau des clusters de bases de données utilisant Amazon Aurora PostgreSQL Limitless Database

Pour effectuer la mise à niveau d’un cluster de bases de données, vous devez le modifier et sélectionner une nouvelle version du moteur de base de données. Pour ce faire, vous pouvez utiliser la AWS Management Console ou la AWS CLI.

Console

Pour mettre à niveau votre cluster de bases de données Limitless Database

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Accédez à la page Bases de données.
3. Sélectionnez le cluster de bases de données Limitless Database que vous souhaitez mettre à niveau.
4. Sélectionnez Modify.

   La page Modifier le cluster de bases de données s’affiche.
5. Pour la version du moteur de base de données, choisissez la nouvelle version du moteur de base de données, par exemple Aurora PostgreSQL avec Limitless Database (compatible avec PostgreSQL 16.6).
6. Choisissez Continuer.
7. Sur la page de résumé, indiquez si les modifications doivent être appliquées immédiatement ou au cours du prochain créneau de maintenance planifié, puis choisissez Modifier le cluster.

Interface de ligne de commande (CLI)

Utilisez la commande modify-db-cluster de l'AWS CLI, comme illustré dans l'exemple suivant.

```
aws rds modify-db-cluster \
    --db-cluster-identifier my-sv2-cluster \
    --engine-version 16.6-limitless \
    --apply-immediately
```

Référence Aurora PostgreSQL Limitless Database

Nous proposons les rubriques de référence suivantes pour Aurora PostgreSQL Limitless Database.

Rubriques

Commandes SQL du langage de définition de données (DDL) prises en charge et non prises en charge

Le tableau suivant répertorie les commandes DDL prises en charge et non prises en charge par Aurora PostgreSQL Limitless Database, ainsi que les éventuelles limitations et informations supplémentaires.

Command	Pris en charge ?	Limitations ou informations supplémentaires
ALTER AGREGAT	Non	Ne s’applique pas
ALTER COLLATION	Oui	Aucun
ALTER CONVERSION	Oui	Aucun
ALTER DATABASE	Non	Ne s’applique pas
ALTER DEFAULT PRIVILEGES	Non	Ne s’applique pas

Command	Pris en charge ?	Limitations ou informations supplémentaires
ALTER DOMAIN	Non	Ne s’applique pas
ALTER EVENT TRIGGER	Non	Ne s’applique pas
ALTER EXTENSION	Oui	Extensions
ALTER FOREIGN DATA WRAPPER	Non	Ne s’applique pas
ALTER FOREIGN TABLE	Non	Ne s’applique pas
ALTER FUNCTION	Oui	Fonctions
ALTER GROUP	Oui	Aucun
ALTER INDEX	Oui	Aucun
ALTER LANGUAGE	Non	Ne s’applique pas
ALTER LARGE OBJECT	Non	Ne s’applique pas
ALTER MATERIALIZED VIEW	Non	Ne s’applique pas
ALTER OPERATOR	Oui	Aucun
ALTER OPERATOR CLASS	Oui	Aucun
ALTER OPERATOR FAMILY	Oui	Aucun
ALTER POLICY	Non	Ne s’applique pas
ALTER PROCEDURE	Oui	Aucun
ALTER PUBLICATION	Non	Ne s’applique pas
ALTER ROLE	Oui	Aucun
ALTER ROUTINE	Non	Ne s’applique pas

Command	Pris en charge ?	Limitations ou informations supplémentaires
ALTER RULE	Non	Ne s’applique pas
ALTER SCHEMA	Oui	Aucun
ALTER SEQUENCE	Oui	Séquences
ALTER SERVER	Non	Ne s’applique pas
ALTER STATISTICS	Non	Ne s’applique pas
ALTER SUBSCRIPTION	Non	Ne s’applique pas
ALTER SYSTEM	Non	Ne s’applique pas
ALTER TABLE	Oui	ALTER TABLE
ALTER TABLESPACE	Non	Ne s’applique pas
ALTER TEXT SEARCH CONFIGURATION	Non	Ne s’applique pas
ALTER TEXT SEARCH DICTIONARY	Non	Ne s’applique pas
ALTER TEXT SEARCH PARSER	Non	Ne s’applique pas
ALTER TEXT SEARCH TEMPLATE	Non	Ne s’applique pas
ALTER TRIGGER	Non	Ne s’applique pas
ALTER TYPE	Oui	Aucun
ALTER USER	Oui	Aucun
ALTER USER MAPPING	Non	Ne s’applique pas

Command	Pris en charge ?	Limitations ou informations supplémentaires
ALTER VIEW	Oui	Aucun
COMMENT	Non	Ne s’applique pas
CREATE ACCESS METHOD	Non	Ne s’applique pas
CREATE AGGREGATE	Non	Ne s’applique pas
CREATE CAST	Oui	Aucun
CREATE COLLATION	Oui	Aucun
CREATE CONVERSION	Oui	Aucun
CREATE DATABASE	Oui	CREATE DATABASE
CREATE DOMAIN	Non	Ne s’applique pas
CREATE EVENT TRIGGER	Non	Ne s’applique pas
CREATE EXTENSION	Oui	Extensions
CREATE FOREIGN DATA WRAPPER	Non	Ne s’applique pas
CREATE FOREIGN TABLE	Non	Ne s’applique pas
CREATE FUNCTION	Oui	Fonctions
CREATE GROUP	Oui	Aucun
CREATE INDEX	Oui	CREATE INDEX
CREATE LANGUAGE	Non	Ne s’applique pas
CREATE MATERIALIZED VIEW	Non	Ne s’applique pas
CREATE OPERATOR	Oui	Aucun

Command	Pris en charge ?	Limitations ou informations supplémentaires
CREATE OPERATOR CLASS	Oui	Aucun
CREATE OPERATOR FAMILY	Oui	Aucun
CREATE POLICY	Oui	Aucun
CREATE PROCEDURE	Oui	Aucun
CREATE PUBLICATION	Non	Ne s’applique pas
CREATE ROLE	Oui	Aucun
CREATE RULE	Non	Ne s’applique pas
CREATE SCHEMA	Oui	CREATE SCHEMA
CREATE SEQUENCE	Oui	Séquences
CREATE SERVER	Non	Ne s’applique pas
CREATE STATISTICS	Non	Ne s’applique pas
CREATE SUBSCRIPTION	Non	Ne s’applique pas
CREATE TABLE	Oui	CREATE TABLE
CREATE TABLE AS	Oui	CREATE TABLE AS
CREATE TABLESPACE	Non	Ne s’applique pas
CREATE TEMPORARY TABLE	Non	Ne s’applique pas
CREATE TEMPORARY TABLE AS	Non	Ne s’applique pas

Command	Pris en charge ?	Limitations ou informations supplémentaires
CREATE TEXT SEARCH CONFIGURATION	Non	Ne s’applique pas
CREATE TEXT SEARCH DICTIONARY	Non	Ne s’applique pas
CREATE TEXT SEARCH PARSER	Non	Ne s’applique pas
CREATE TEXT SEARCH TEMPLATE	Non	Ne s’applique pas
CREATE TRANSFORM	Non	Ne s’applique pas
CREATE TRIGGER	Non	Ne s’applique pas
CREATE TYPE	Oui	Aucun
CREATE USER	Oui	Aucun
CREATE USER MAPPING	Non	Ne s’applique pas
CREATE VIEW	Oui	Aucun
DROP ACCESS METHOD	Non	Ne s’applique pas
DROP AGGREGATE	Oui	Aucun
DROP CAST	Oui	Aucun
DROP COLLATION	Oui	Aucun
DROP CONVERSION	Oui	Aucun
DROP DATABASE	Oui	DROP DATABASE
DROP DOMAIN	Non	Ne s’applique pas

Command	Pris en charge ?	Limitations ou informations supplémentaires
DROP EVENT TRIGGER	Non	Ne s’applique pas
DROP EXTENSION	Oui	Extensions
DROP FOREIGN DATA WRAPPER	Non	Ne s’applique pas
DROP FOREIGN TABLE	Non	Ne s’applique pas
DROP FUNCTION	Oui	Fonctions
DROP GROUP	Oui	Aucun
DROP INDEX	Oui	Aucun
DROP LANGUAGE	Non	Ne s’applique pas
DROP MATERIALIZED VIEW	Non	Ne s’applique pas
DROP OPERATOR	Oui	Aucun
DROP OPERATOR CLASS	Oui	Aucun
DROP OPERATOR FAMILY	Oui	Aucun
DROP OWNED	Non	Ne s’applique pas
DROP POLICY	Non	Ne s’applique pas
DROP PROCEDURE	Oui	Aucun
DROP PUBLICATION	Non	Ne s’applique pas
DROP ROLE	Oui	Aucun
DROP ROUTINE	Non	Ne s’applique pas
DROP RULE	Non	Ne s’applique pas

Command	Pris en charge ?	Limitations ou informations supplémentaires
DROP SCHEMA	Oui	Aucun
DROP SEQUENCE	Oui	Aucun
DROP SERVER	Non	Ne s’applique pas
DROP STATISTICS	Non	Ne s’applique pas
DROP SUBSCRIPTION	Non	Aucun
DROP TABLE	Oui	Aucun
DROP TABLESPACE	Non	Ne s’applique pas
DROP TEXT SEARCH CONFIGURATION	Non	Ne s’applique pas
DROP TEXT SEARCH DICTIONARY	Non	Ne s’applique pas
DROP TEXT SEARCH PARSER	Non	Ne s’applique pas
DROP TEXT SEARCH TEMPLATE	Non	Ne s’applique pas
DROP TRANSFORM	Non	Ne s’applique pas
DROP TRIGGER	Non	Ne s’applique pas
DROP TYPE	Oui	Aucun
DROP USER	Oui	Aucun
DROP USER MAPPING	Non	Ne s’applique pas
DROP VIEW	Oui	Aucun

Command	Pris en charge ?	Limitations ou informations supplémentaires
GRANT	Oui	Aucun
REASSIGN OWNED	Non	Ne s’applique pas
REVOKE	Oui	Aucun
SECURITY LABEL	Non	Ne s’applique pas
SELECT INTO	Oui	SELECT INTO
SET	Oui	Aucun
SET CONSTRAINTS	Non	Ne s’applique pas
SET ROLE	Oui	Aucun
SET SESSION AUTHORIZATION	Oui	Aucun
SET TRANSACTION	Oui	Aucun
TRUNCATE	Oui	Aucun

Limitations du langage DDL et autres informations relatives à Aurora PostgreSQL Limitless Database

Les sections suivantes décrivent les limitations ou apportent des informations supplémentaires concernant les commandes SQL DDL dans Aurora PostgreSQL Limitless Database.

Rubriques

ALTER TABLE

La commande `ALTER TABLE` est généralement prise en charge dans Aurora PostgreSQL Limitless Database. Pour plus d'informations, consultez ALTER TABLE dans la documentation PostgreSQL.

Limitations

`ALTER TABLE` présente les limitations suivantes pour les options prises en charge.

Suppression d'une colonne

- Vous ne pouvez pas supprimer les colonnes qui font partie de la clé de partition dans les tables partitionnées.
- Vous ne pouvez pas supprimer les colonnes de clé primaire dans les tables de référence.

Modification du type de données d'une colonne

- L'expression `USING` n'est pas prise en charge.
- Vous ne pouvez pas modifier le type des colonnes qui font partie de la clé de partition dans les tables partitionnées.

Ajout ou suppression d'une contrainte

Pour en savoir plus sur les fonctionnalités non prises en charge, consultez Constaintes.

Modification de la valeur par défaut d'une colonne

Les valeurs par défaut sont prises en charge. Pour plus d'informations, consultez Valeurs par défaut.

Options non prises en charge

Certaines options ne sont pas prises en charge, car elles dépendent de fonctionnalités non prises en charge, telles que les déclencheurs.

Les options au niveau de la table suivantes pour `ALTER TABLE` ne sont pas prises en charge :

- `ALL IN TABLESPACE`
- `ATTACH PARTITION`
- `DETACH PARTITION`
- Indicateur `ONLY`

- `RENAME CONSTRAINT`

Les options au niveau de la colonne suivantes pour `ALTER TABLE` ne sont pas prises en charge :

- ADD GENERATED
- DROP EXPRESSION [IF EXISTS]
- DROP IDENTITY [IF EXISTS]
- RESET
- RESTART
- SET
- SET COMPRESSION
- SET STATISTICS

CREATE DATABASE

Dans Aurora PostgreSQL Limitless Database, seules les bases de données sans limite sont prises en charge.

Lors de l'exécution de `CREATE DATABASE`, les bases de données créées sur un ou plusieurs nœuds peuvent échouer sur d'autres nœuds, car la création d'une base de données est une opération non transactionnelle. Dans ce cas, les objets de base de données créés sont automatiquement supprimés de tous les nœuds dans un délai prédéterminé afin de maintenir la cohérence du groupe de partitions de base de données. Pendant ce temps, la recréation d'une base de données portant le même nom peut entraîner une erreur indiquant que la base de données existe déjà.

Les options suivantes sont prises en charge :

- Classement :

```
CREATE DATABASE name WITH
    [LOCALE = locale]
    [LC_COLLATE = lc_collate]
    [LC_CTYPE = lc_ctype]
    [ICU_LOCALE = icu_locale]
    [ICU_RULES = icu_rules]
    [LOCALE_PROVIDER = locale_provider]
    [COLLATION_VERSION = collation_version];
```

- `CREATE DATABASE WITH OWNER`:

```
CREATE DATABASE name WITH OWNER = user_name;
```

Les options suivantes ne sont pas prises en charge :

- `CREATE DATABASE WITH TABLESPACE`:

```
CREATE DATABASE name WITH TABLESPACE = tablespace_name;
```

- `CREATE DATABASE WITH TEMPLATE`:

```
CREATE DATABASE name WITH TEMPLATE = template;
```

CREATE INDEX

`CREATE INDEX CONCURRENTLY` est pris en charge pour les tables partitionnées :

```
CREATE INDEX CONCURRENTLY index_name ON table_name(column_name);
```

`CREATE UNIQUE INDEX` est pris en charge pour tous les types de tables :

```
CREATE UNIQUE INDEX index_name ON table_name(column_name);
```

`CREATE UNIQUE INDEX CONCURRENTLY` n'est pas pris en charge :

```
CREATE UNIQUE INDEX CONCURRENTLY index_name ON table_name(column_name);
```

Pour plus d'informations, consultez UNIQUE. Pour obtenir des informations générales sur la création d'index, consultez CREATE INDEX dans la documentation PostgreSQL.

Affichage des index

> Les index ne sont pas tous visibles sur les routeurs lorsque vous utilisez `\d table_name` ou des commandes similaires. Utilisez plutôt la vue `pg_catalog.pg_indexes` pour obtenir des index, comme illustré dans l'exemple suivant.

```
SET rds_aurora.limitless_create_table_mode='sharded';
```

```
SET rds_aurora.limitless_create_table_shard_key='{"id"}';
CREATE TABLE items (id int PRIMARY KEY, val int);
CREATE INDEX items_my_index on items (id, val);

postgres_limitless=> SELECT * FROM pg_catalog.pg_indexes WHERE tablename='items';

 schemaname | tablename |   indexname    | tablespace |
   indexdef
------------+-----------+---------------+------------
+-------------------------------------------------------------------------
 public     | items     | items_my_index |            | CREATE INDEX items_my_index
 ON ONLY public.items USING btree (id, val)
 public     | items     | items_pkey     |            | CREATE UNIQUE INDEX
 items_pkey ON ONLY public.items USING btree (id)
(2 rows)
```

CREATE SCHEMA

`CREATE SCHEMA` avec un élément de schéma n'est pas pris en charge :

```
CREATE SCHEMA my_schema CREATE TABLE (column_name INT);
```

Un erreur semblable à la suivante est générée :

```
ERROR: CREATE SCHEMA with schema elements is not supported
```

CREATE TABLE

Les relations dans les instructions `CREATE TABLE` ne sont pas prises en charge, par exemple :

```
CREATE TABLE orders (orderid int, customerId int, orderDate date) WITH
 (autovacuum_enabled = false);
```

Les colonnes `IDENTITY` ne sont pas prises en charge, par exemple :

```
CREATE TABLE orders (orderid INT GENERATED ALWAYS AS IDENTITY);
```

Aurora PostgreSQL Limitless Database prend en charge jusqu'à 54 caractères pour les noms de table partitionnées.

CREATE TABLE AS

Pour créer une table à l'aide de `CREATE TABLE AS`, vous devez utiliser la variable `rds_aurora.limitless_create_table_mode`. Pour les tables partitionnées, vous devez également utiliser la variable `rds_aurora.limitless_create_table_shard_key`. Pour plus d'informations, consultez Création de tables sans limite à l'aide de variables.

```
-- Set the variables.
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"a"}';

CREATE TABLE ctas_table AS SELECT 1 a;

-- "source" is the source table whose columns and data types are used to create the new
 "ctas_table2" table.
CREATE TABLE ctas_table2 AS SELECT a,b FROM source;
```

Vous ne pouvez pas les utiliser `CREATE TABLE AS` pour créer des tables de référence, car elles nécessitent des contraintes de clé primaire. `CREATE TABLE AS`ne propage pas les clés primaires vers les nouvelles tables.

Pour des informations générales, consultez CREATE TABLE AS dans la documentation PostgreSQL.

DROP DATABASE

Vous pouvez supprimer les bases de données que vous avez créées.

La commande `DROP DATABASE` s'exécute de manière asynchrone en arrière-plan. Durant l'exécution de la commande, une erreur s'affichera si vous essayez de créer une base de données portant le même nom.

SELECT INTO

`SELECT INTO` est fonctionnellement similaire à CREATE TABLE AS. Vous devez utiliser la variable `rds_aurora.limitless_create_table_mode`. Pour les tables partitionnées, vous devez également utiliser la variable `rds_aurora.limitless_create_table_shard_key`. Pour plus d'informations, consultez Création de tables sans limite à l'aide de variables.

```
-- Set the variables.
SET rds_aurora.limitless_create_table_mode='sharded';
```

```
SET rds_aurora.limitless_create_table_shard_key='{"a"}';

-- "source" is the source table whose columns and data types are used to create the new
 "destination" table.
SELECT * INTO destination FROM source;
```

Actuellement, l'opération `SELECT INTO` est effectuée via le routeur, et non directement via les partitions. Les performances peuvent donc être réduites.

Pour des informations générales, consultez SELECT INTO dans la documentation PostgreSQL.

Constaintes

Les limitations suivantes s'appliquent aux contraintes d'Aurora PostgreSQL Limitless Database.

CHECK

Les contraintes simples qui utilisent des opérateurs de comparaison avec des littéraux sont prises en charge. Les expressions et contraintes plus complexes qui nécessitent des validations de fonctions ne sont pas prises en charge, comme le montrent les exemples suivants.

```
CREATE TABLE my_table (
    id  INT CHECK (id > 0)                                        -- supported
  , val INT CHECK (val > 0 AND val < 1000)                        -- supported
  , tag TEXT CHECK (length(tag) > 0)                              -- not supported:
 throws "Expression inside CHECK constraint is not supported"
  , op_date TIMESTAMP WITH TIME ZONE CHECK (op_date <= now())  -- not supported:
 throws "Expression inside CHECK constraint is not supported"
);
```

Vous pouvez attribuer des noms explicites aux contraintes, comme illustré dans l'exemple suivant.

```
CREATE TABLE my_table (
    id  INT CONSTRAINT positive_id  CHECK (id > 0)
  , val INT CONSTRAINT val_in_range CHECK (val > 0 AND val < 1000)
);
```

Vous pouvez utiliser une syntaxe de contrainte au niveau de la table avec la contrainte `CHECK`, comme illustré dans l'exemple suivant.

```
CREATE TABLE my_table (
    id INT CONSTRAINT positive_id  CHECK (id > 0)
  , min_val INT CONSTRAINT min_val_in_range CHECK (min_val > 0 AND min_val < 1000)
  , max_val INT
  , CONSTRAINT max_val_in_range CHECK (max_val > 0 AND max_val < 1000 AND max_val >
 min_val)
);
```

EXCLUDE

Les contraintes d'exclusion ne sont pas prises en charge dans Aurora PostgreSQL Limitless Database.

FOREIGN KEY

Pour plus d'informations, consultez Clés étrangères.

NOT NULL

Les contraintes `NOT NULL` sont prises en charge sans aucune restriction.

PRIMARY KEY

Étant donné que la clé primaire impose des contraintes uniques, elle est soumise aux mêmes restrictions que celles des contraintes uniques. Autrement dit :

- Si une table est convertie en table partitionnée, la clé de partition doit être un sous-ensemble de la clé primaire. C'est-à-dire que la clé primaire contient toutes les colonnes de la clé de partition.
- Si une table est convertie en table de référence, elle doit comporter une clé primaire.

Les exemples suivants illustrent l'utilisation des clés primaires.

```
-- Create a standard table.
CREATE TABLE public.my_table (
    item_id INT
  , location_code INT
  , val INT
  , comment text
);

-- Change the table to a sharded table using the 'item_id' and 'location_code'
 columns as shard keys.
CALL rds_aurora.limitless_alter_table_type_sharded('public.my_table',
 ARRAY['item_id', 'location_code']);
```

Tentative d'ajout d'une clé primaire qui ne contient pas de clé de partition :

```
-- Add column 'item_id' as the primary key.
-- Invalid because the primary key doesnt include all columns from the shard key:
-- 'location_code' is part of the shard key but not part of the primary key
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id); -- ERROR

-- add column "val" as primary key
-- Invalid because primary key does not include all columns from shard key:
--  item_id and location_code iare part of shard key but not part of the primary key
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id); -- ERROR
```

Tentative d'ajout d'une clé primaire qui contient une clé de partition :

```
-- Add the 'item_id' and 'location_code' columns as the primary key.
-- Valid because the primary key contains the shard key.
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id, location_code); -- OK

-- Add the 'item_id', 'location_code', and 'val' columns as the primary key.
-- Valid because the primary key contains the shard key.
ALTER TABLE public.my_table ADD PRIMARY KEY (item_id, location_code, val); -- OK
```

Remplacez une table standard par une table de référence.

```
-- Create a standard table.
CREATE TABLE zipcodes (zipcode INT PRIMARY KEY, details VARCHAR);

-- Convert the table to a reference table.
CALL rds_aurora.limitless_alter_table_type_reference('public.zipcode');
```

Pour plus d'informations sur la création de tables partitionnées et de tables de référence, consultez Création de tables Aurora PostgreSQL Limitless Database.

UNIQUE

Dans les tables partitionnées, la clé unique doit contenir la clé de partition, c'est-à-dire que la clé de partition doit être un sous-ensemble de la clé unique. Cette vérification est effectuée lors du changement du type de table en table partitionnée. Aucune restriction n'est appliquée aux tables de référence.

```
CREATE TABLE customer (
    customer_id INT NOT NULL
  , zipcode INT
  , email TEXT UNIQUE
);
```

Les contraintes UNIQUE au niveau de la table sont prises en charge, comme illustré dans l'exemple suivant.

```
CREATE TABLE customer (
    customer_id INT NOT NULL
  , zipcode INT
  , email TEXT
```

```
  , CONSTRAINT zipcode_and_email UNIQUE (zipcode, email)
);
```

L'exemple suivant montre l'utilisation conjointe d'une clé primaire et d'une clé unique. Les deux clés doivent inclure la clé de partition.

```
SET rds_aurora.limitless_create_table_mode='sharded';
SET rds_aurora.limitless_create_table_shard_key='{"p_id"}';
CREATE TABLE t1 (
p_id BIGINT NOT NULL,
c_id BIGINT NOT NULL,
PRIMARY KEY (p_id),
UNIQUE (p_id, c_id)
);
```

Pour plus d'informations, consultez Contraintes dans la documentation PostgreSQL.

Valeurs par défaut

Aurora PostgreSQL Limitless Database prend en charge les expressions dans les valeurs par défaut.

L'exemple suivant illustre l'utilisation des valeurs par défaut.

```
CREATE TABLE t (
    a INT DEFAULT 5,
    b TEXT DEFAULT 'NAN',
    c NUMERIC
);

CALL rds_aurora.limitless_alter_table_type_sharded('t', ARRAY['a']);
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c
---+-----+---
 5 | NAN |
(1 row)
```

Les expressions sont prises en charge, comme illustré dans l'exemple suivant.

```
CREATE TABLE t1 (a NUMERIC DEFAULT random());
```

L'exemple suivant illustre l'ajout d'une nouvelle colonne qui présente la valeur `NOT NULL` et une valeur par défaut.

```
ALTER TABLE t ADD COLUMN d BOOLEAN NOT NULL DEFAULT FALSE;
SELECT * FROM t;

 a |  b  | c | d
---+-----+---+---
 5 | NAN |   | f
(1 row)
```

L'exemple suivant illustre la modification d'une colonne existante avec une valeur par défaut.

```
ALTER TABLE t ALTER COLUMN c SET DEFAULT 0.0;
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c   |  d
---+-----+-----+-----
 5 | NAN |     | f
 5 | NAN | 0.0 | f
(2 rows)
```

L'exemple suivant illustre la suppression d'une valeur par défaut.

```
ALTER TABLE t ALTER COLUMN a DROP DEFAULT;
INSERT INTO t DEFAULT VALUES;
SELECT * FROM t;

 a |  b  | c   |  d
---+-----+-----+-----
 5 | NAN |     | f
 5 | NAN | 0.0 | f
   | NAN | 0.0 | f
(3 rows)
```

Pour plus d'informations, consultez Valeurs par défaut dans la documentation PostgreSQL.

Extensions

Les fonctionnalités PostgreSQL suivantes sont prises en charge dans Aurora PostgreSQL Limitless Database :

- `aurora_limitless_fdw` : cette extension est préinstallée. Vous ne pouvez pas la supprimer.
- `aws_s3` : le fonctionnement de cette extension est similaire dans Aurora PostgreSQL Limitless Database et dans Aurora PostgreSQL.

  Vous pouvez importer des données à partir d'un compartiment Amazon S3 vers un cluster de bases de données Aurora PostgreSQL Limitless Database ou exporter des données à partir d'un cluster de bases de données Aurora PostgreSQL Limitless Database vers un compartiment Amazon S3. Pour plus d'informations, consultez Importation de données Amazon S3 dans une d'un cluster de base de données Aurora PostgreSQL et Exportation de données à partir d'un cluster de bases de données Aurora PostgreSQL vers Amazon S3.
- `btree_gin`
- `citext`
- `ip4r`
- `pg_buffercache` : cette extension se comporte différemment dans Aurora PostgreSQL Limitless Database par rapport à PostgreSQL communautaire. Pour plus d'informations, consultez Différences pg_buffercache dans Aurora PostgreSQL Limitless Database.
- `pg_stat_statements`
- `pg_trgm`
- `pgcrypto`
- `pgstattuple` : cette extension se comporte différemment dans Aurora PostgreSQL Limitless Database par rapport à PostgreSQL communautaire. Pour plus d'informations, consultez Différences pgstattuple dans Aurora PostgreSQL Limitless Database.
- `pgvector`
- `plpgsql` : cette extension est préinstallée, mais vous pouvez la supprimer.
- `PostGIS` : les transactions longues et les fonctions de gestion de tables ne sont pas prises en charge. La modification de la table de référence spatiale n'est pas prise en charge.
- `unaccent`
- `uuid`

La plupart des extensions PostgreSQL ne sont pas prises en charge dans Aurora PostgreSQL Limitless Database. Toutefois, vous pouvez toujours utiliser le paramètre de configuration shared_preload_libraries (SPL) pour charger des extensions dans le cluster de bases de données principal Aurora PostgreSQL. Même si elles sont également chargées dans Aurora PostgreSQL Limitless Database, elles risquent de ne pas fonctionner correctement.

Vous pouvez, par exemple, charger l'extension `pg_hint_plan`, mais son chargement ne garantit pas que les indications transmises dans les commentaires de requête soient prises en compte.

> **Note**
>
> Vous ne pouvez pas modifier les objets associés à l'extension pg_stat_statements. Pour en savoir plus sur l'installation `pg_stat_statements`, consultez limitless_stat_statements.

Vous pouvez utiliser les fonctions `pg_available_extensions` et `pg_available_extension_versions` pour rechercher les extensions prises en charge dans Aurora PostgreSQL Limitless Database.

Les DDL suivants sont compatibles avec les extensions :

CREATE EXTENSION

Vous pouvez créer des extensions, comme dans PostgreSQL.

```
CREATE EXTENSION [ IF NOT EXISTS ] extension_name
    [ WITH ] [ SCHEMA schema_name ]
             [ VERSION version ]
             [ CASCADE ]
```

Pour plus d'informations, consultez CREATE EXTENSION dans la documentation PostgreSQL.

ALTER EXTENSION

Les DDL suivants sont pris en charge :

```
ALTER EXTENSION name UPDATE [ TO new_version ]

ALTER EXTENSION name SET SCHEMA new_schema
```

Pour plus d'informations, consultez ALTER EXTENSION dans la documentation PostgreSQL.

DROP EXTENSION

Vous pouvez supprimer des extensions, comme dans PostgreSQL.

```
DROP EXTENSION [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

Pour plus d'informations, consultez DROP EXTENSION dans la documentation PostgreSQL.

Les DDL suivants ne sont pas compatibles avec les extensions :

ALTER EXTENSION

Vous ne pouvez pas ajouter ou supprimer des objets membres des extensions.

```
ALTER EXTENSION name ADD member_object

ALTER EXTENSION name DROP member_object
```

Différences pg_buffercache dans Aurora PostgreSQL Limitless Database

Dans Aurora PostgreSQL Limitless Database, lorsque vous installez l'extension pg_buffercache et que vous utilisez la vue `pg_buffercache`, vous ne recevez des informations relatives aux tampons que depuis le nœud auquel vous êtes actuellement connecté, à savoir le routeur. De même, l'utilisation de la fonction `pg_buffercache_summary` ou `pg_buffercache_usage_counts` ne fournit des informations que depuis le nœud connecté.

Il est possible que vous ayez plusieurs nœuds et que vous deviez consulter les informations sur les tampons depuis n'importe lequel d'entre eux afin de diagnostiquer efficacement les problèmes. Limitless Database fournit donc les fonctions suivantes :

- `rds_aurora.limitless_pg_buffercache(subcluster_id)`
- `rds_aurora.limitless_pg_buffercache_summary(subcluster_id)`
- `rds_aurora.limitless_pg_buffercache_usage_counts(subcluster_id)`

En saisissant l'ID de sous-cluster de n'importe quel nœud, qu'il s'agisse d'un routeur ou d'une partition, vous pouvez facilement accéder aux informations sur les tampons spécifiques à ce nœud. Ces fonctions sont directement disponibles lorsque vous installez l'extension `pg_buffercache` dans la base de données sans limite.

> **Note**
>
> Aurora PostgreSQL Limitless Database prend en charge ces fonctions pour les versions 1.4 et ultérieures de l'extension pg_buffercache.

Les colonnes affichées dans la vue limitless_pg_buffercache sont légèrement différentes de celles de la vue pg_buffercache :

- bufferid : similaire à pg_buffercache.
- relname : au lieu d'afficher le numéro de nœud de fichier tel qu'il est indiqué dans pg_buffercache, limitless_pg_buffercache présente le relname associé s'il est disponible dans la base de données actuelle ou dans les catalogues de systèmes partagés ; à défaut, il présente la valeur NULL.
- parent_relname : cette nouvelle colonne, absente dans pg_buffercache, affiche le relname parent si la valeur de la colonne relname représente une table partitionnée (dans le cas de tables partitionnées). Dans le cas contraire, il présente la valeur NULL.
- spcname : au lieu d'afficher l'identifiant d'objet de l'espace de table (OID) comme dans pg_buffercache, limitless_pg_buffercache affiche le nom de l'espace de table.
- datname : au lieu d'afficher l'OID de la base de données comme dans pg_buffercache, limitless_pg_buffercache affiche le nom de la base de données.
- relforknumber : similaire à pg_buffercache.
- relblocknumber : similaire à pg_buffercache.
- isdirty : similaire à pg_buffercache.
- usagecount : similaire à pg_buffercache.
- pinning_backends : similaire à pg_buffercache.

Les colonnes des vues limitless_pg_buffercache_summary et limitless_pg_buffercache_usage_counts sont similaires à celles des vues standard pg_buffercache_summary et pg_buffercache_usage_counts, respectivement.

L'utilisation de ces fonction vous permet d'obtenir des informations détaillées sur les tampons de l'ensemble des nœuds de votre environnement Limitless Database, facilitant ainsi le diagnostic et la gestion de vos systèmes de base de données.

Différences pgstattuple dans Aurora PostgreSQL Limitless Database

Dans Aurora PostgreSQL, l'extension pgstattuple ne prend actuellement pas en charge les tables étrangères, les tables partitionnées ou les index partitionnés. Toutefois, dans Aurora PostgreSQL Limitless Database, les objets créés par les utilisateurs appartiennent souvent à ces types non pris en charge. Bien qu'il existe des tables et des index classiques (par exemple, les tables de catalogue et leurs index), la plupart des objets résident sur des nœuds étrangers, et sont donc considérés comme des objets étrangers par le routeur.

Nous reconnaissons l'importance de cette extension pour l'obtention de statistiques au niveau des tuples, essentielles pour des tâches telles que la suppression du gonflement et la collecte d'informations de diagnostic. Par conséquent, Aurora PostgreSQL Limitless Database prend en charge l'extension `pgstattuple` dans des bases de données sans limite.

Aurora PostgreSQL Limitless Database inclut les fonctions suivantes dans le schéma `rds_aurora` :

Fonctions de statistiques au niveau des tuples

`rds_aurora.limitless_pgstattuple(`*`relation_name`*`)`

- Objectif : extraire des statistiques au niveau des tuples pour les tables standard et leurs index
- Entrée : `relation_name` (texte) : le nom de la relation
- Sortie : colonnes cohérentes avec celles renvoyées par la fonction `pgstattuple` dans Aurora PostgreSQL

`rds_aurora.limitless_pgstattuple(`*`relation_name`*`, `*`subcluster_id`*`)`

- Objectif : extraire des statistiques au niveau des tuples pour les tables de référence, les tables partitionnées, les tables de catalogue et leurs index
- Entrée :
  - `relation_name` (texte) : le nom de la relation
  - `subcluster_id` (texte) : l'ID du sous-cluster du nœud où les statistiques doivent être extraites
- Sortie :
  - Pour les tables de référence et de catalogue (y compris leurs index), les colonnes sont cohérentes avec celles d'Aurora PostgreSQL.
  - Pour les tables partitionnées, les statistiques représentent uniquement la partition de la table partitionnée résidant sur le sous-cluster spécifié.

Fonctions de statistiques d’index

`rds_aurora.limitless_pgstatindex(relation_name)`

- Objectif : extraire les statistiques des index B-tree sur des tables standard
- Entrée : `relation_name` (texte) : le nom de l’index B-tree
- Sortie : toutes les colonnes sauf `root_block_no` sont renvoyées. Les colonnes transmises sont cohérentes avec celles renvoyées par la fonction `pgstatindex` dans Aurora PostgreSQL.

`rds_aurora.limitless_pgstatindex(relation_name, subcluster_id)`

- Objectif : extraire les statistiques des index B-tree sur les tables de référence, les tables partitionnées et les tables de catalogue.
- Entrée :
  - `relation_name` (texte) : le nom de l’index B-tree
  - `subcluster_id` (texte) : l’ID du sous-cluster du nœud où les statistiques doivent être extraites
- Sortie :
  - Pour les index des tables de référence et de catalogue, toutes les colonnes (sauf `root_block_no`) sont renvoyées. Les colonnes transmises sont cohérentes avec celles d’Aurora PostgreSQL.
  - Pour les tables partitionnées, les statistiques représentent uniquement la partition de l’index de la table partitionnée résidant sur le sous-cluster spécifié. La colonne `tree_level` indique la moyenne de toutes les tranches de table du sous-cluster demandé.

`rds_aurora.limitless_pgstatginindex(relation_name)`

- Objectif : extraire les statistiques des index inversés généralisés (GIN) sur les tables standard
- Entrée : `relation_name` (texte) : le nom du GIN
- Sortie : colonnes cohérentes avec celles renvoyées par la fonction `pgstatginindex` dans Aurora PostgreSQL

`rds_aurora.limitless_pgstatginindex(relation_name, subcluster_id)`

- Objectif : extraire les statistiques des index GIN sur les tables de référence, les tables partitionnées et les tables de catalogue.
- Entrée :
  - `relation_name` (texte) : le nom de l’index

    - `subcluster_id` (texte) : l'ID du sous-cluster du nœud où les statistiques doivent être extraites
- Sortie :
    - Pour les index GIN des tables de référence et de catalogue, les colonnes sont cohérentes avec celles d'Aurora PostgreSQL.
    - Pour les tables partitionnées, les statistiques représentent uniquement la partition de l'index de la table partitionnée résidant sur le sous-cluster spécifié.

`rds_aurora.limitless_pgstathashindex(`*`relation_name`*`)`

- Objectif : extraire les statistiques des index de hachage sur des tables standard
- Entrée : `relation_name` (texte) : le nom de l'index de hachage
- Sortie : colonnes cohérentes avec celles renvoyées par la fonction `pgstathashindex` dans Aurora PostgreSQL

`rds_aurora.limitless_pgstathashindex(`*`relation_name`*`, `*`subcluster_id`*`)`

- Objectif : extraire les statistiques des index de hachage sur les tables de référence, des tables partitionnées et des tables de catalogue.
- Entrée :
    - `relation_name` (texte) : le nom de l'index
    - `subcluster_id` (texte) : l'ID du sous-cluster du nœud où les statistiques doivent être extraites
- Sortie :
    - Pour les index de hachage des tables de référence et de catalogue, les colonnes sont cohérentes avec celles d'Aurora PostgreSQL.
    - Pour les tables partitionnées, les statistiques représentent uniquement la partition de l'index de la table partitionnée résidant sur le sous-cluster spécifié.

Fonctions de comptage de pages

`rds_aurora.limitless_pg_relpages(`*`relation_name`*`)`

- Objectif : extraire le nombre de pages pour les tables standard et leurs index
- Entrée : `relation_name` (texte) : le nom de la relation
- Sortie : nombre de pages de la relation spécifiée

`rds_aurora.limitless_pg_relpages(`*`relation_name`*`, `*`subcluster_id`*`)`

- Objectif : extraire le nombre de pages pour les tables de référence, les tables partitionnées et les tables de catalogue (y compris leurs index)
- Entrée :
  - `relation_name` (texte) : le nom de la relation
  - `subcluster_id` (texte) : l'ID du sous-cluster du nœud où le nombre de page doit être extrait
- Sortie : pour les tables partitionnées, le nombre de pages correspond à la somme des pages de toutes les tranches de tableau du sous-cluster spécifié.

Fonctions de statistiques approximatives au niveau des tuples

`rds_aurora.limitless_pgstattuple_approx(`*`relation_name`*`)`

- Objectif : extraire des statistiques approximatives au niveau des tuples pour les tables standard et leurs index
- Entrée : `relation_name` (texte) : le nom de la relation
- Sortie : colonnes cohérentes avec celles renvoyées par la fonction pgstattuple_approx dans Aurora PostgreSQL

`rds_aurora.limitless_pgstattuple_approx(`*`relation_name`*`, `*`subcluster_id`*`)`

- Objectif : extraire des statistiques approximatives au niveau des tuples pour les tables de référence, les tables partitionnées et les tables de catalogue (y compris leurs index)
- Entrée :
  - `relation_name` (texte) : le nom de la relation
  - `subcluster_id` (texte) : l'ID du sous-cluster du nœud où les statistiques doivent être extraites
- Sortie :
  - Pour les tables de référence et de catalogue (y compris leurs index), les colonnes sont cohérentes avec celles d'Aurora PostgreSQL.
  - Pour les tables partitionnées, les statistiques représentent uniquement la partition de la table partitionnée résidant sur le sous-cluster spécifié.

> **Note**
>
> Actuellement, Aurora PostgreSQL Limitless Database ne prend pas en charge l'extension `pgstattuple` sur les vues matérialisées, les tables TOAST ou les tables temporaires.

Dans Aurora PostgreSQL Limitless Database, vous devez fournir l’entrée sous forme de texte, bien qu’Aurora PostgreSQL prenne en charge d’autres formats.

Clés étrangères

Les contraintes de clé étrangère (`FOREIGN KEY`) sont prises en charge avec certaines restrictions :

- `CREATE TABLE` avec `FOREIGN KEY` est pris en charge uniquement pour les tables standard. Pour créer une table partitionnée ou de référence avec `FOREIGN KEY`, créez d'abord la table sans contrainte de clé étrangère. Modifiez-la ensuite à l'aide de l'instruction suivante :

  ```
  ALTER TABLE ADD CONSTRAINT;
  ```

- La conversion d'une table standard en table partitionnée ou en table de référence n'est pas prise en charge lorsque la table est soumise à une contrainte de clé étrangère. Supprimez la contrainte, puis ajoutez-la après la conversion.
- Les limitations suivantes concernent les types de tables comportant des contraintes de clé étrangère :
  - Une table standard peut être soumise à une contrainte de clé étrangère vers une autre table standard.
  - Une table partitionnée peut être soumise à une contrainte de clé étrangère lorsque les tables parente et enfant sont colocalisées et que la clé étrangère constitue un sur-ensemble de la clé de partition.
  - Une table partitionnée peut être soumise à une contrainte de clé étrangère vers une table de référence.
  - Une table de référence peut être soumise à une contrainte de clé étrangère vers une autre table de référence.

Rubriques

Options à clé étrangère

Les clés étrangères sont prises en charge dans Aurora PostgreSQL Limitless Database pour certaines options DDL. Le tableau suivant répertorie les options prises en charge et non prises en charge entre les tables Aurora PostgreSQL Limitless Database.

Option DDL	Référence à référence	Partitionnée à partitionnée (colocalisée)	Partitionnée à référence	Standard à standard
`DEFERRABLE`	Oui	Oui	Oui	Oui
`INITIALLY DEFERRED`	Oui	Oui	Oui	Oui
`INITIALLY IMMEDIATE`	Oui	Oui	Oui	Oui
`MATCH FULL`	Oui	Oui	Oui	Oui
`MATCH PARTIAL`	Non	Non	Non	Non
`MATCH SIMPLE`	Oui	Oui	Oui	Oui
`NOT DEFERRABLE`	Oui	Oui	Oui	Oui
`NOT VALID`	Oui	Non	Non	Oui
`ON DELETE CASCADE`	Oui	Oui	Oui	Oui
`ON DELETE NO ACTION`	Oui	Oui	Oui	Oui
`ON DELETE RESTRICT`	Oui	Oui	Oui	Oui
`ON DELETE SET DEFAULT`	Non	Non	Non	Non
`ON DELETE SET NULL`	Oui	Non	Non	Oui

Option DDL	Référence à référence	Partitionnée à partitionnée (colocalisée)	Partitionnée à référence	Standard à standard
`ON UPDATE CASCADE`	Non	Non	Non	Oui
`ON UPDATE NO ACTION`	Oui	Oui	Oui	Oui
`ON UPDATE RESTRICT`	Oui	Oui	Oui	Oui
`ON UPDATE SET DEFAULT`	Non	Non	Non	Non
`ON UPDATE SET NULL`	Oui	Non	Non	Oui

Exemples

- Standard à standard :

```
set rds_aurora.limitless_create_table_mode='standard';

CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer REFERENCES products (product_no),
    quantity integer
);

SELECT constraint_name, table_name, constraint_type
FROM information_schema.table_constraints WHERE constraint_type='FOREIGN KEY';
```

```
 constraint_name         | table_name  | constraint_type
-------------------------+-------------+-----------------
 orders_product_no_fkey  | orders      | FOREIGN KEY
(1 row)
```

- Partitionnée à partitionnée (colocalisée) :

```
set rds_aurora.limitless_create_table_mode='sharded';
set rds_aurora.limitless_create_table_shard_key='{"product_no"}';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

set rds_aurora.limitless_create_table_shard_key='{"order_id"}';
set rds_aurora.limitless_create_table_collocate_with='products';
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

- Partitionnée à référence :

```
set rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);

set rds_aurora.limitless_create_table_mode='sharded';
set rds_aurora.limitless_create_table_shard_key='{"order_id"}';
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);
```

```
ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

- Référence à référence :

```
set rds_aurora.limitless_create_table_mode='reference';
CREATE TABLE products(
    product_no integer PRIMARY KEY,
    name text,
    price numeric
);
CREATE TABLE orders (
    order_id integer PRIMARY KEY,
    product_no integer,
    quantity integer
);

ALTER TABLE orders ADD CONSTRAINT order_product_fk FOREIGN KEY (product_no)
 REFERENCES products (product_no);
```

Fonctions

Les fonctions sont prises en charge dans Aurora PostgreSQL Limitless Database.

Les DDL suivants sont pris en charge pour les fonctions :

CREATE FUNCTION

Il est possible de créer des fonctions comme dans Aurora PostgreSQL, toutefois, la volatilité ne peut pas être changée lors d'un remplacement.

Pour plus d'informations, consultez CREATE FUNCTION dans la documentation PostgreSQL.

ALTER FUNCTION

Vous pouvez modifier des fonctions, comme dans Aurora PostgreSQL, à l'exception du changement de leur volatilité.

Pour plus d'informations, consultez ALTER FUNCTION dans la documentation PostgreSQL.

DROP FUNCTION

Vous pouvez supprimer des fonctions, comme dans Aurora PostgreSQL.

```
DROP FUNCTION [ IF EXISTS ] name [ ( [ [ argmode ] [ argname ] argtype [, ...] ] ) ]
 [, ...]
    [ CASCADE | RESTRICT ]
```

Pour plus d'informations, consultez DROP FUNCTION dans la documentation PostgreSQL.

Rubriques

- Répartition des fonctions
- Volatilité des fonctions

Répartition des fonctions

Lorsque toutes les instructions d'une fonction sont ciblées sur une seule partition, il est avantageux de transférer l'ensemble de la fonction vers la partition cible. Le résultat est simplement renvoyé au routeur, sans que la fonction y soit déroulée. La fonctionnalité de poussée vers le bas des fonctions

et procédures stockées est utile pour les clients qui souhaitent exécuter leur fonction ou procédure stockée au plus près de la source de données, c'est-à-dire de la partition.

Pour distribuer une fonction, commencez par la créer avant d'appeler la procédure `rds_aurora.limitless_distribute_function` en vue de la distribuer. Cette fonction utilise la syntaxe suivante :

```
SELECT rds_aurora.limitless_distribute_function('function_prototype',
 ARRAY['shard_key'], 'collocating_table');
```

La fonction présente les paramètres suivants :

- *`function_prototype`* : la fonction à distribuer. Mentionnez uniquement les arguments d'entrée, en excluant les arguments de sortie.

  Si l'un des arguments est défini en tant que paramètres `OUT`, n'incluez pas son type dans les arguments `function_prototype`.
- `ARRAY['`*`shard_key`*`']` : la liste des arguments de fonction identifiés comme étant la clé de partition de la fonction.
- *`collocating_table`* : la table partitionnée qui contient la plage de données sur la partition cible.

Pour identifier la partition sur laquelle pousser cette fonction vers la bas, le système utilise l'argument `ARRAY['`*`shard_key`*`']`, calcule sa valeur de hachage, puis recherche la partition dans *`collocating_table`* qui héberge la plage contenant cette valeur de hachage.

Restrictions

Lorsque vous distribuez une fonction ou une procédure, elle ne traite que les données délimitées par la plage de clés de cette partition. Dans les cas où la fonction ou la procédure essaie d'accéder aux données d'une autre partition, les résultats renvoyés par la fonction ou procédure distribuée seront différents de ceux renvoyés par une fonction ou procédure non distribuée.

Par exemple, vous créez une fonction contenant des requêtes qui toucheront plusieurs partitions, puis vous appelez la procédure `rds_aurora.limitless_distribute_function` pour la distribuer. Lorsque vous invoquez cette fonction en fournissant des arguments pour une clé de partition, il est probable que les résultats de son exécution seront limités par les valeurs présentes dans cette partition. Ces résultats diffèrent de ceux obtenus lorsque la fonction n'est pas distribuée.

Exemples

Prenons l'exemple de la fonction `func` suivante, avec la table partitionnée `customers` et la clé de partition `customer_id`.

```
postgres_limitless=> CREATE OR REPLACE FUNCTION func(c_id integer, sc integer)
 RETURNS int
    language SQL
    volatile
    AS $$
    UPDATE customers SET score = sc WHERE customer_id = c_id RETURNING score;
    $$;
```

Nous distribuons cette fonction :

```
SELECT rds_aurora.limitless_distribute_function('func(integer, integer)',
 ARRAY['c_id'], 'customers');
```

Voici quelques exemples de plans de requête.

```
EXPLAIN(costs false, verbose true) SELECT func(27+1,10);

                    QUERY PLAN
 ---------------------------------------------------
  Foreign Scan
    Output: (func((27 + 1), 10))
    Remote SQL:  SELECT func((27 + 1), 10) AS func
  Single Shard Optimized
 (4 rows)
```

```
EXPLAIN(costs false, verbose true)
 SELECT * FROM customers,func(customer_id, score) WHERE customer_id=10 AND score=27;

                          QUERY PLAN
 ---------------------------------------------------------------------
  Foreign Scan
    Output: customer_id, name, score, func
    Remote SQL:  SELECT customers.customer_id,
      customers.name,
      customers.score,
      func.func
```

```
     FROM public.customers,
      LATERAL func(customers.customer_id, customers.score) func(func)
    WHERE ((customers.customer_id = 10) AND (customers.score = 27))
  Single Shard Optimized
 (10 rows)
```

L'exemple suivant illustre une procédure utilisant les paramètres `IN` et `OUT` en tant qu'arguments.

```
CREATE OR REPLACE FUNCTION get_data(OUT id INTEGER, IN arg_id INT)
    AS $$
    BEGIN
        SELECT customer_id,
        INTO id
        FROM customer
        WHERE customer_id = arg_id;
    END;
    $$ LANGUAGE plpgsql;
```

L'exemple suivant illustre la distribution de la procédure utilisant uniquement les paramètres `IN`.

```
EXPLAIN(costs false, verbose true) SELECT * FROM get_data(1);

             QUERY PLAN
 -----------------------------------
  Foreign Scan
    Output: id
    Remote SQL:  SELECT customer_id
     FROM get_data(1) get_data(id)
  Single Shard Optimized
 (6 rows)
```

Volatilité des fonctions

Vous pouvez déterminer si une fonction est immuable, stable ou volatile en vérifiant la valeur `provolatile` dans la vue pg_proc. La valeur `provolatile` indique si le résultat de la fonction dépend uniquement de ses arguments d'entrée ou s'il est affecté par des facteurs extérieurs.

La valeur est l'une des suivantes :

- `i` : fonctions immuables, qui fournissent toujours le même résultat pour les mêmes entrées
- `s` : fonctions stables, dont les résultats (pour les entrées fixes) ne changent pas au cours d'un scan

- v : fonctions volatiles, dont les résultats peuvent changer à tout moment. Utilisez également v pour les fonctions présentant des effets secondaires, de sorte que leurs appels ne puissent pas être optimisés.

Les exemples suivants présentent des fonctions volatiles.

```
SELECT proname, provolatile FROM pg_proc WHERE proname='pg_sleep';

 proname  | provolatile
----------+-------------
 pg_sleep | v
(1 row)

SELECT proname, provolatile FROM pg_proc WHERE proname='uuid_generate_v4';

     proname      | provolatile
------------------+-------------
 uuid_generate_v4 | v
(1 row)

SELECT proname, provolatile FROM pg_proc WHERE proname='nextval';

 proname | provolatile
---------+-------------
 nextval | v
(1 row)
```

La modification de la volatilité d'une fonction existante n'est pas prise en charge dans Aurora PostgreSQL Limitless Database. Cela s'applique à la fois aux commandes `ALTER FUNCTION` et `CREATE OR REPLACE FUNCTION`, comme indiqué dans les exemples suivants.

```
-- Create an immutable function
CREATE FUNCTION immutable_func1(name text) RETURNS text language plpgsql
AS $$
BEGIN
    RETURN name;
END;
$$IMMUTABLE;

-- Altering the volatility throws an error
ALTER FUNCTION immutable_func1 STABLE;
```

```
-- Replacing the function with altered volatility throws an error
CREATE OR REPLACE FUNCTION immutable_func1(name text) RETURNS text language plpgsql
AS $$
BEGIN
    RETURN name;
END;
$$VOLATILE;
```

Nous vous recommandons vivement d'attribuer les volatilités appropriées aux fonctions. Par exemple, si votre fonction utilise `SELECT` à partir de plusieurs tables ou fait référence à des objets de la base de données, ne la définissez pas sur `IMMUTABLE`. Si le contenu de la table change, l'immuabilité est rompue.

Aurora PostgreSQL permet l'utilisation de `SELECT` dans les fonctions immuables, toutefois les résultats obtenus risquent d'être inexacts. Aurora PostgreSQL Limitless Database peut renvoyer à la fois des erreurs et des résultats incorrects. Pour plus d'informations sur la volatilité des fonctions, consultez Catégories de volatilité des fonctions dans la documentation PostgreSQL.

Séquences

Les séquences nommées sont des objets de base de données qui génèrent des nombres uniques dans un ordre croissant ou décroissant. `CREATE SEQUENCE` crée un nouveau générateur de numéros de séquence. L'unicité des valeurs de séquence est garantie.

Lorsque vous créez une séquence nommée dans Aurora PostgreSQL Limitless Database, un objet de séquence distribuée est créé. Aurora PostgreSQL Limitless Database distribue ensuite des fragments de valeurs de séquence non superposés sur tous les routeurs de transactions distribués (routeurs). Les fragments sont représentés sous forme d'objets de séquence locaux sur les routeurs ; par conséquent, les opérations de séquence telles que `nextval` et `currval` sont exécutées localement. Les routeurs fonctionnent indépendamment et demandent de nouveaux fragments à partir de la séquence distribuée en cas de besoin.

Pour plus d'informations sur les séquences, consultez CREATE SEQUENCE dans la documentation PostgreSQL.

Rubriques

Demande d'un nouveau fragment

La taille des fragments alloués sur les routeurs peut être configurée à l'aide du paramètre `rds_aurora.limitless_sequence_chunk_size`. La valeur par défaut est `250000`. Chaque routeur présente initialement deux fragments : actif et réservé. Les fragments actifs sont utilisés pour configurer les objets de séquence locaux (paramètre `minvalue` et `maxvalue`), et les fragments réservés sont stockés dans une table de catalogue interne. Lorsqu'un fragment actif atteint la valeur minimale ou maximale, il est remplacé par le segment réservé. Pour ce faire, `ALTER SEQUENCE` est utilisé en interne, ce qui signifie que `AccessExclusiveLock` est acquis.

Des processus en arrière-plan s'exécutent toutes les 10 secondes sur les nœuds des routeurs afin de rechercher les séquences dont les fragments réservés ont été utilisés. Si un fragment utilisé est

trouvé, le processus demande un nouveau fragment à partir de la séquence distribuée. Assurez-vous de définir une taille de fragment suffisamment grande pour que les processus en arrière-plan aient suffisamment de temps pour en demander de nouveaux. Les demandes à distance ne s'exécutent jamais dans le contexte des sessions utilisateur ; vous ne pouvez donc pas demander directement une nouvelle séquence.

Limitations

Les limitations suivantes s'appliquent aux séquences dans Aurora PostgreSQL Limitless Database :

- Le catalogue `pg_sequence`, la fonction `pg_sequences` et l'instruction `SELECT * FROM sequence_name` affichent tous uniquement l'état local de la séquence, et non son état distribué.
- Les valeurs d'une séquence sont garanties uniques et monotones au sein d'une même session. Toutefois, l'ordre des valeurs peut différer de celui des instructions `nextval` exécutées dans d'autres sessions connectées à d'autres routeurs.
- Assurez-vous que la taille de séquence (nombre de valeurs disponibles) est suffisamment grande pour être répartie sur tous les routeurs. Utilisez le paramètre `rds_aurora.limitless_sequence_chunk_size` pour configurer `chunk_size`. (Chaque routeur comporte deux fragments.)
- L'option `CACHE` est prise en charge, mais le cache doit être inférieur à`chunk_size`.

Options non prises en charge

Les options suivantes ne sont pas prises en charge dans Aurora PostgreSQL Limitless Database.

Fonctions de manipulation de séquence

La fonction `setval` n'est pas prise en charge. Pour plus d'informations, consultez Fonctions de manipulation de séquence dans la documentation PostgreSQL.

CREATE SEQUENCE

Les options suivantes ne sont pas prises en charge.

```
CREATE [{ TEMPORARY | TEMP} | UNLOGGED] SEQUENCE
    [[ NO ] CYCLE]
```

Pour plus d'informations, consultez CREATE SEQUENCE dans la documentation PostgreSQL.

ALTER SEQUENCE

Les options suivantes ne sont pas prises en charge.

```
ALTER SEQUENCE
    [[ NO ] CYCLE]
```

Pour plus d'informations, consultez ALTER SEQUENCE dans la documentation PostgreSQL.

ALTER TABLE

La commande `ALTER TABLE` n'est pas compatible avec les séquences.

Exemples

CREATE/DROP SEQUENCE

```
postgres_limitless=> CREATE SEQUENCE s;
CREATE SEQUENCE

postgres_limitless=> SELECT nextval('s');

 nextval
---------
       1
(1 row)

postgres_limitless=> SELECT * FROM pg_sequence WHERE seqrelid='s'::regclass;

 seqrelid | seqtypid | seqstart | seqincrement | seqmax | seqmin | seqcache |
 seqcycle
----------+----------+----------+--------------+--------+--------+----------
+----------
    16960 |       20 |        1 |            1 |  10000 |      1 |        1 | f
(1 row)

% connect to another router
postgres_limitless=> SELECT nextval('s');

 nextval
---------
   10001
(1 row)
```

```
postgres_limitless=> SELECT * FROM pg_sequence WHERE seqrelid='s'::regclass;

 seqrelid | seqtypid | seqstart | seqincrement | seqmax | seqmin | seqcache |
 seqcycle
----------+----------+----------+--------------+--------+--------+----------
+----------
    16959 |       20 |    10001 |            1 |  20000 |  10001 |        1 | f
(1 row)

postgres_limitless=> DROP SEQUENCE s;
DROP SEQUENCE
```

ALTER SEQUENCE

```
postgres_limitless=> CREATE SEQUENCE s;
CREATE SEQUENCE

postgres_limitless=> ALTER SEQUENCE s RESTART 500;
ALTER SEQUENCE

postgres_limitless=> SELECT nextval('s');

 nextval
---------
     500
(1 row)

postgres_limitless=> SELECT currval('s');

 currval
---------
     500
(1 row)
```

Fonctions de manipulation de séquence

```
postgres=# CREATE TABLE t(a bigint primary key, b bigint);
CREATE TABLE

postgres=# CREATE SEQUENCE s minvalue 0 START 0;
CREATE SEQUENCE
```

```
postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));

INSERT 0 1

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));
INSERT 0 1

postgres=# SELECT * FROM t;

 a | b
---+---
 0 | 0
 1 | 1
(2 rows)

postgres=# ALTER SEQUENCE s RESTART 10000;
ALTER SEQUENCE

postgres=# INSERT INTO t VALUES (nextval('s'), currval('s'));

INSERT 0 1

postgres=# SELECT * FROM t;

   a   |   b
-------+-------
     0 |     0
     1 |     1
 10000 | 10000
(3 rows)
```

Vues de séquence

Aurora PostgreSQL Limitless Database propose les vues suivantes pour les séquences.

rds_aurora.limitless_distributed_sequence

Cette vue montre l'état et la configuration d'une séquence distribuée. Les colonnes `minvalue`, `maxvalue`, `start`, `inc` et `cache` ont la même signification que dans la vue pg_sequences et indiquent les options avec lesquelles la séquence a été créée. La colonne `lastval` indique la dernière valeur allouée ou réservée dans un objet de séquence distribuée. Cela ne signifie pas que la valeur a déjà été utilisée, puisque les routeurs conservent les fragments de séquence localement.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_distributed_sequence WHERE
 sequence_name='test_serial_b_seq';

 schema_name |   sequence_name   | lastval | minvalue |  maxvalue  | start | inc |
 cache
-------------+-------------------+---------+----------+------------+-------+-----
+-------
 public      | test_serial_b_seq | 1250000 |        1 | 2147483647 |     1 |   1 |
   1
(1 row)
```

rds_aurora.limitless_sequence_metadata

Cette vue montre les métadonnées de séquence distribuée et agrège les métadonnées de séquence provenant des nœuds du cluster. Elle utilise les colonnes suivantes :

- `subcluster_id` : l'ID du nœud du cluster qui possède un fragment.
- Fragment actif : fragment d'une séquence en cours d'utilisation (`active_minvalue`, `active_maxvalue`).
- Fragment réservé : le fragment local qui sera utilisé ensuite (`reserved_minvalue`, `reserved_maxvalue`).
- `local_last_value` : la dernière valeur observée à partir d'une séquence locale.
- `chunk_size` : la taille d'un fragment, telle qu'elle a été configurée lors de la création.

```
postgres_limitless=> SELECT * FROM rds_aurora.limitless_sequence_metadata WHERE
 sequence_name='test_serial_b_seq' order by subcluster_id;
```

```
 subcluster_id |   sequence_name   | schema_name | active_minvalue | active_maxvalue
 | reserved_minvalue | reserved_maxvalue | chunk_size | chunk_state |
 local_last_value
---------------+-------------------+-------------+-----------------
+-----------------+-------------------+-------------------+------------
+-------------+------------------
 1             | test_serial_b_seq | public      |          500001 |          750000
 |           1000001 |           1250000 |     250000 |           1 |
 550010
 2             | test_serial_b_seq | public      |          250001 |          500000
 |            750001 |           1000000 |     250000 |           1 |

(2 rows)
```

Résolution des problèmes liés aux séquences

Les problèmes suivants peuvent survenir avec les séquences.

La taille des fragments est insuffisante

Si la taille des fragments n'est pas suffisamment grande et que le taux de transactions est élevé, les processus en arrière-plan risquent de ne pas avoir le temps de demander de nouveaux fragments avant que les fragments actifs ne soient épuisés. Cela peut entraîner une contention et des événements d'attente tels que `LIMITLESS:AuroraLimitlessSequenceReplace`, `LWLock:LockManager` , `Lockrelation` et `LWlock:bufferscontent`.

Augmentez la valeur du paramètre `rds_aurora.limitless_sequence_chunk_size`.

Le cache de séquence défini est trop élevé

Dans PostgreSQL, la mise en cache des séquences s'effectue au niveau de la session. Chaque session alloue des valeurs de séquence successives lors d'un accès à l'objet de séquence et augmente la `last_value` de l'objet de la séquence en conséquence. Par la suite, les utilisations de `nextval` au sein de la même session retournent directement les valeurs préallouées, sans interaction avec l'objet de séquence.

Tous les numéros alloués mais non utilisés au cours d'une session sont perdus à la fin de cette dernière, ce qui entraîne des « trous » dans la séquence. Cette situation peut provoquer une consommation rapide du sequence_chunk et générer des contentions et événements d'attente comme `LIMITLESS:AuroraLimitlessSequenceReplace`, `LWLock:LockManager` , `Lockrelation` et `LWlock:bufferscontent`.

Réduisez le paramètre de mise en cache des séquences.

La figure suivante montre les événements d'attente provoqués par des problèmes de séquence.

Commandes SQL de traitement de requêtes et de manipulation de données (DML) prises en charge ou non prises en charge

Le tableau suivant répertorie les commandes DML prises en charge et non prises en charge par Aurora PostgreSQL Limitless Database, ainsi que les éventuelles limitations et informations supplémentaires.

Command	Pris en charge ?	Limitations ou informations supplémentaires
ABORT	Oui	Aucun
ANALYSE	Oui	ANALYSE
BEGIN	Oui	Aucun
CALL	Oui	Aucun
CHECKPOINT	Oui	Aucun
CLOSE	Oui	Aucun
CLUSTER	Oui	CLUSTER
COMMIT	Oui	Aucun
COMMIT PREPARE	Non	Ne s’applique pas
COPY	Oui	Aucun
DEALLOCATE	Oui	Aucun
DECLARE	Oui	Aucun
DELETE	Oui	Aucun
DISCARD	Oui	Aucun
DO	Oui	Aucun
FIN	Oui	Aucun

Command	Pris en charge ?	Limitations ou informations supplémentaires
EXECUTE	Oui	Aucun
EXPLAIN	Oui	EXPLAIN
FETCH	Oui	Aucun
IMPORT FOREIGN SCHEMA	Non	Ne s’applique pas
INSERT	Oui	INSERT
LISTEN	Non	Ne s’applique pas
LOCK	Oui	Aucun
MERGE	Non	Ne s’applique pas
MOVE	Oui	Aucun
NOTIFY	Non	Ne s’applique pas
OPEN	Oui	Aucun
PREPARE	Oui	Aucun
PREPARE TRANSACTION	Non	Ne s’applique pas
REFRESH MATERIALIZED VIEW	Non	Ne s’applique pas
REINDEX	Non	Ne s’applique pas
RELEASE SAVEPOINT	Oui	Aucun
ROLLBACK	Oui	Aucun
ROLLBACK PREPARED	Non	Ne s’applique pas
ROLLBACK TO SAVEPOINT	Oui	Aucun

Command	Pris en charge ?	Limitations ou informations supplémentaires
SAVEPOINT	Oui	Aucun
SELECT	Oui	Aucun
SELECT INTO	Oui	Aucun
MONTRER	Oui	Aucun
START TRANSACTION	Oui	Aucun
UNLISTEN	Non	Aucun
UPDATE	Oui	UPDATE
UPDATE … WHERE CURRENT OF	Non	Ne s’applique pas
VACUUM	Oui	VACUUM
VALUES	Oui	Aucun

Limitations du langage DML et autres informations relatives à Aurora PostgreSQL Limitless Database

Les sections suivantes décrivent les limitations ou apportent des informations supplémentaires concernant les commandes SQL de traitement des données (DML) et de traitement des requêtes dans Aurora PostgreSQL Limitless Database.

Rubriques

ANALYSE

La commande `ANALYZE` collecte des statistiques sur le contenu des tables de la base de données. Le planificateur de requêtes utilise ensuite ces statistiques pour déterminer les plans d'exécution les plus efficaces pour les requêtes. Pour en savoir plus, consultez ANALYZE dans la documentation PostgreSQL.

Dans Aurora PostgreSQL Limitless Database, la commande `ANALYZE` collecte les statistiques des tables sur tous les routeurs et partitions lors de son exécution.

Pour empêcher le calcul de statistiques sur chaque routeur pendant l'exécution de la commande `ANALYZE`, les statistiques des tables sont calculées sur l'un des routeurs, puis copiées sur des routeurs homologues.

CLUSTER

La commande `CLUSTER` réorganise physiquement une table en fonction d'un index. L'index doit déjà avoir été défini dans la table. Dans Aurora PostgreSQL Limitless Database, le clustering s'applique uniquement à la portion de l'index stockée sur chaque partition.

Pour en savoir plus, consultez CLUSTER dans la documentation PostgreSQL.

EXPLAIN

Le paramètre suivant est utilisé pour configurer le résultat produit par la commande `EXPLAIN` :

- `rds_aurora.limitless_explain_options` : ce qu'il faut inclure dans la sortie `EXPLAIN`. La valeur par défaut est `single_shard_optimization` : elle indique si les plans sont optimisés pour une partition unique, sans toutefois inclure les plans de partition.

Dans cet exemple, la sortie `EXPLAIN` n'affiche pas les plans issus des partitions.

```
postgres_limitless=> EXPLAIN SELECT * FROM employees where id =25;

                      QUERY PLAN
------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
 Single Shard Optimized
(2 rows)
```

Maintenant, nous définissons `rds_aurora.limitless_explain_options` pour inclure `shard_plans` et `single_shard_optimization`. Nous pouvons consulter les plans d'exécution des instructions à la fois sur les routeurs et sur les partitions. En outre, nous désactivons le paramètre `enable_seqscan` pour garantir que le scan d'index soit utilisé sur la couche de partition.

```
postgres_limitless=> SET rds_aurora.limitless_explain_options = shard_plans,
 single_shard_optimization;
SET

postgres_limitless=> SET enable_seqscan = OFF;
SET

postgres_limitless=> EXPLAIN SELECT * FROM employees WHERE id = 25;

                                                    QUERY PLAN
-------------------------------------------------------------------------------------------------------
 Foreign Scan  (cost=100.00..101.00 rows=100 width=0)
   Remote Plans from Shard postgres_s4:
         Index Scan using employees_ts00287_id_idx on employees_ts00287
 employees_fs00003  (cost=0.14..8.16 rows=1 width=15)
           Index Cond: (id = 25)
 Single Shard Optimized
(5 rows)
```

Pour plus d'informations sur la commande `EXPLAIN`, consultez EXPLAIN dans la documentation PostgreSQL.

INSERT

La plupart des commandes `INSERT` sont prises en charge dans Aurora PostgreSQL Limitless Database.

PostgreSQL ne propose pas de commande explicite `UPSERT`, mais permet d'exécuter des instructions `INSERT ... ON CONFLICT`.

La commande `INSERT ... ON CONFLICT` n'est pas prise en charge si l'action de conflit comporte une sous-requête ou une fonction mutable :

```
-- RANDOM is a mutable function.
INSERT INTO sharded_table VALUES (1, 100) ON CONFLICT (id) DO UPDATE SET other_id =
 RANDOM();

ERROR: Aurora Limitless Tables doesn't support pushdown-unsafe functions with DO UPDATE
 clauses.
```

Pour plus d'informations sur la commande `INSERT`, consultez INSERT dans la documentation PostgreSQL.

UPDATE

La mise à jour de la clé de partition n'est pas prise en charge. Supposons que vous disposiez d'une table partitionnée nommée `customers`, avec une clé de partition `customer_id`. Les instructions DML suivantes sont à l'origine d'erreurs :

```
postgres_limitless=> UPDATE customers SET customer_id = 11 WHERE customer_id =1;
ERROR:  Shard key column update is not supported

postgres_limitless=> UPDATE customers SET customer_id = 11 WHERE customer_name='abc';
ERROR:  Shard key column update is not supported
```

Pour mettre à jour une clé de partition, vous devez d'abord exécuter la commande `DELETE` sur la ligne contenant la clé de partition, puis `INSERT` sur une nouvelle ligne avec la valeur de clé de partition mise à jour.

Pour plus d'informations sur la commande UPDATE, consultez Mise à jour de données dans la documentation PostgreSQL.

VACUUM

Vous pouvez effectuer un vacuum sur les tables partitionnées et sur les tables de référence. Les fonctions VACUUM suivantes sont entièrement prises en charge dans Aurora PostgreSQL Limitless Database :

- VACUUM
- ANALYSE
- DISABLE_PAGE_SKIPPING
- FREEZE
- FULL
- INDEX_CLEANUP
- PARALLEL
- PROCESS_TOAST
- TRUNCATE
- VERBOSE

Sur Aurora PostgreSQL Limitless Database, la fonction VACUUM présente les limitations suivantes :

- L'extension pg_visibility_map n'est pas prise en charge.
- La vérification des index inutilisés avec la vue pg_stat_all_indexes n'est pas prise en charge.
- Les vues unifiées de pg_stat_user_indexes, pg_class et pg_stats ne sont pas disponibles.

Pour plus d'informations sur la commande VACUUM, consultez VACUUM dans la documentation PostgreSQL. Pour plus d'informations sur le fonctionnement du vacuum dans Aurora PostgreSQL Limitless Database, consultez Récupération de l'espace de table via une opération de vacuum.

Variables dans Aurora PostgreSQL Limitless Database

Vous pouvez utiliser les variables suivantes pour configurer Aurora PostgreSQL Limitless Database.

rds_aurora.limitless_active_shard_key

Définit une clé de partition unique lors de l'interrogation de la base de données, ce qui permet d'ajouter la clé de partition à toutes les requêtes `SELECT` et DML en tant que prédicat constant. Pour plus d'informations, consultez Configuration d'une clé de partition active.

rds_aurora.limitless_create_table_collocate_with

Attribuez à cette variable le nom d'une table spécifique pour colocaliser les tables nouvellement créées avec celle-ci. Pour plus d'informations, consultez Création de tables sans limite à l'aide de variables.

rds_aurora.limitless_create_table_mode

Définit le mode de création de table. Pour plus d'informations, consultez Création de tables sans limite à l'aide de variables.

rds_aurora.limitless_create_table_shard_key

Définissez cette variable en lui attribuant un tableau de noms de colonnes qui serviront de clés de partition. Pour plus d'informations, consultez Création de tables sans limite à l'aide de variables.

rds_aurora.limitless_explain_options

Ce qu'il faut inclure dans la sortie `EXPLAIN`. Pour plus d'informations, consultez EXPLAIN.

Paramètres de cluster de bases de données dans Aurora PostgreSQL Limitless Database

Vous pouvez utiliser les paramètres de cluster de bases de données suivants pour configurer Aurora PostgreSQL Limitless Database.

rds_aurora.limitless_adaptive_fetch_size

Améliore l'extraction préalable par lots. Lorsqu'il est défini sur `true`, ce paramètre permet un ajustement automatique (adaptatif) de la taille d'extraction lors de l'extraction préalable des données. Lorsqu'elle est définie sur `false`, la taille d'extraction est constante.

rds_aurora.limitless_auto_scale_options

Définit les options disponibles pour ajouter des routeurs ou diviser des partitions dans un groupe de partitions de base de données. Cette valeur peut être `add_router`, `split_shard` ou les deux.

Pour plus d'informations, consultez Ajout d'un routeur à un groupe de partitions de base de données et Fractionnement d'une partition dans un groupe de partitions de base de données.

rds_aurora.limitless_distributed_deadlock_timeout

Durée d'attente sur un verrou avant de vérifier la présence d'un blocage distribué, exprimée en millisecondes. La valeur par défaut est `1000` (1 seconde).

Pour plus d'informations, consultez Blocages distribués dans Aurora PostgreSQL Limitless Database.

rds_aurora.limitless_enable_auto_scale

Permet d'ajouter des routeurs et de diviser les partitions dans un groupe de partitions de base de données.

Pour plus d'informations, consultez Ajout d'un routeur à un groupe de partitions de base de données et Fractionnement d'une partition dans un groupe de partitions de base de données.

rds_aurora.limitless_finalize_split_shard_mode

Détermine la manière dont les divisions de partitions initiées par le système sont finalisées. Pour plus d'informations, consultez Fractionnement d'une partition dans un groupe de partitions de base de données.

rds_aurora.limitless_maximum_adaptive_fetch_size

Définit la limite supérieure de la taille d’extraction adaptative. L’intervalle de valeurs est de 1 – `INT_MAX`. La valeur par défaut est `1000`.

Utilisation d’Amazon Aurora Global Database

Avec la fonctionnalité Amazon Aurora Global Database, vous pouvez configurer plusieurs clusters de bases de données Aurora répartis sur plusieurs Régions AWS. Aurora synchronise automatiquement toutes les modifications apportées dans le cluster de bases de données principal avec un ou plusieurs clusters secondaires. Une base de données globale Aurora a un cluster de bases de données principal dans une région et jusqu’à 10 clusters de bases de données secondaires dans différentes régions. Cette configuration multi-régions assure une reprise rapide dans les rares cas où une panne pourrait affecter une Région AWS entière. Le fait de disposer d’une copie complète de toutes vos données dans plusieurs zones géographiques permet également d’effectuer des opérations de lecture à faible latence pour les applications qui se connectent à partir de lieux très éloignés dans le monde entier.

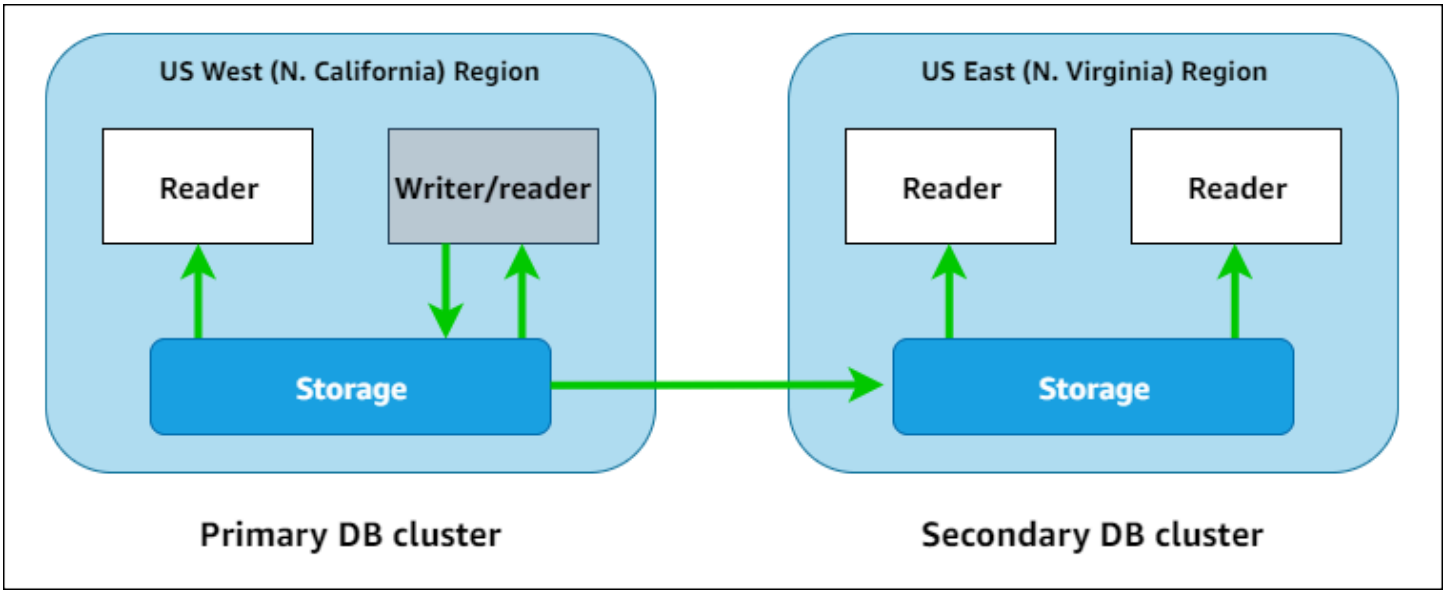
Rubriques

Présentation d’Amazon Aurora Global Database

Avec la fonctionnalité Amazon Aurora Global Database, vous pouvez exécuter vos applications distribuées à l’échelle mondiale à l’aide d’une base de données Aurora unique couvrant plusieurs Régions AWS.

Une base de données globale Aurora se compose d'une base de données principale Région AWS dans laquelle vos données sont écrites et d'un maximum de 10 bases secondaires en lecture seule. Régions AWS Vous devez émettre les opérations d’écriture directement vers le cluster de bases de données principal de l’ Région AWS principale. La méthode la plus pratique consiste à se connecter au point de terminaison d’enregistreur Aurora Global Database, qui pointe toujours vers le cluster de bases de données principal, même après une opération de bascule ou de basculement vers une autre Région AWS. Après chaque opération d'écriture, Aurora réplique les données vers le secondaire à l' Régions AWS aide d'une infrastructure dédiée, avec une latence généralement inférieure à une seconde.

Dans le schéma suivant, vous pouvez trouver un exemple de base de données globale Aurora qui s'étend sur deux Régions AWS.



Vous pouvez augmenter verticalement chaque cluster secondaire de manière indépendante. Pour ce faire, ajoutez-y une ou plusieurs instances de lecteur Aurora afin de répondre aux besoins des charges de travail en lecture seule. Pour une mise à l’échelle encore plus granulaire et flexible, vous pouvez utiliser Aurora Serverless v2 pour les instances de lecteur.

Seul le cluster principal exécute les opérations d’écriture. Les clients qui effectuent des opérations d’écriture se connectent au point de terminaison d’enregistreur Aurora Global Database, qui pointe

toujours vers l'instance de base de données d'enregistreur du cluster principal. Comme indiqué dans le diagramme, Aurora utilise le volume de stockage du cluster et non le moteur de base de données pour assurer une réplication rapide à moindre coût. Pour en savoir plus, consultez Présentation du stockage Amazon Aurora.

Aurora Global Database est conçu pour les applications ayant une empreinte mondiale. Les clusters de base de données secondaires en lecture seule répartis en plusieurs Régions AWS permettent d'optimiser les opérations de lecture au plus près des utilisateurs de l'application. Grâce à la fonctionnalité de transfert d'écriture, vous pouvez aussi configurer une base de données globale afin que les clusters secondaires envoient les demandes au cluster principal. Pour plus d'informations, consultez Utilisation du transfert d'écriture dans une base de données globale Amazon Aurora.

Aurora Global Database prend en charge deux opérations différentes pour modifier la région de votre cluster de bases de données principal, selon le scénario qui s'applique : bascule Aurora Global Database et basculement Aurora Global Database.

- Pour les procédures opérationnelles planifiées telles que la rotation régionale, utilisez le mécanisme de bascule (qui s'appelait auparavant « basculement planifié géré »). Avec cette fonctionnalité, vous pouvez relocaliser le cluster principal d'une base de données Aurora Global Database saine vers l'une de ses régions secondaires sans perte de données. Pour en savoir plus, consultez Réalisation de bascules pour les bases de données Amazon Aurora Global Database.
- Pour récupérer votre base de données Aurora Global Database après une panne dans la région principale, utilisez le mécanisme de basculement. Cette fonctionnalité vous permet de basculer votre cluster de bases de données principal vers une autre région (basculement entre régions). Pour en savoir plus, consultez Réalisation de basculements gérés pour les bases de données globales Aurora.

Avantages d'Amazon Aurora Global Database

Amazon Aurora Global Database offre les avantages suivants :

- Lecture globale avec latence locale : si vous avez des bureaux dans le monde entier, vous pouvez utiliser Aurora Global Database pour mettre à jour vos principales sources d'information dans la Région AWS principale. Les bureaux de vos autres régions peuvent accéder aux informations dans leur propre région, avec une latence locale.
- Clusters de bases de données Aurora secondaires évolutifs : vous pouvez mettre à l'échelle vos clusters secondaires en ajoutant d'autres instances en lecture seule à une Région AWS

secondaire. Le cluster secondaire est en lecture seule, de sorte qu'il peut prendre en charge jusqu'à 16 instances de base de données en lecture seule, au lieu de 15 habituellement par cluster Aurora.

- Réplication rapide du cluster de bases de données Aurora principal vers les clusters secondaires : la réplication effectuée par Aurora Global Database a peu d'impact sur les performances du cluster de bases de données principal. Les ressources des instances de base de données sont entièrement dévouées aux charges de travail d'application en lecture et en écriture.
- Récupération suite aux pannes à l'échelle de la région : les clusters secondaires vous permettent de rendre une base de données Aurora Global Database disponible dans une nouvelle Région AWS principale plus rapidement (RTO moins élevé) et avec moins de perte de données (RPO moins élevé) que les solutions de réplication traditionnelles.

Disponibilité des régions et des versions

La disponibilité et la prise en charge des fonctionnalités varient selon les versions spécifiques de chaque moteur de base de données Aurora, et selon les Régions AWS. Pour en savoir plus sur les versions et les régions disponibles pour Aurora Global Database, consultez Régions et moteurs de base de données pris en charge pour les bases de données globales Aurora.

Limites d'Amazon Aurora Global Database

Les limites suivantes s'appliquent actuellement à Aurora Global Database :

- La base de données globale Aurora est disponible dans certaines versions Régions AWS et pour des versions spécifiques d'Aurora MySQL et d'Aurora PostgreSQL. Pour de plus amples informations, veuillez consulter Régions et moteurs de base de données pris en charge pour les bases de données globales Aurora.
- Aurora Global Database présente des exigences de configuration spécifiques pour les classes d'instance de base de données Aurora prises en charge, le nombre maximal de Régions AWS, etc. Pour plus d'informations, consultez Configuration requise pour une base de données Amazon Aurora globale.
- Pour Aurora MySQL avec la compatibilité MySQL 5.7, les bascules de bases de données globales Aurora nécessitent la version 2.09.1 ou une version mineure supérieure.
- Vous ne pouvez effectuer une bascule ou un basculement géré entre régions avec Aurora Global Database que si les clusters de bases de données principal et secondaires possèdent les mêmes versions de moteur majeures et mineures. Selon le moteur et les versions du moteur,

les niveaux de correctifs doivent parfois être identiques ou peuvent être différents. Pour obtenir la liste des moteurs et des versions de moteur qui autorisent ces opérations entre les clusters principaux et secondaires avec différents niveaux de correctif, consultez Compatibilité des niveaux de correctif pour les bascules ou basculements gérés entre régions. Si les versions de votre moteur nécessitent des niveaux de correctifs identiques, vous pouvez effectuer le basculement manuellement en suivant les étapes décrites dans Réalisation de basculements manuels pour les bases de données globales Aurora.

- Aurora Global Database ne prend actuellement pas en charge les fonctionnalités Aurora suivantes :
  - Aurora Serverless v1
  - Retour sur trace dans Aurora
- Pour connaître les limites de l'utilisation de la fonctionnalité de proxy RDS avec Aurora Global Database, consultez Limites pour le proxy RDS avec les bases de données globales.
- La mise à niveau automatique des versions mineures ne s'applique pas aux clusters Aurora MySQL et Aurora PostgreSQL qui font partie d'une base de données globale. Notez que vous pouvez spécifier ce paramètre pour une instance de base de données faisant partie d'un cluster de bases de données globale, mais ce paramètre est sans effet.
- Aurora Global Database ne prend actuellement pas en charge Aurora Auto Scaling pour les clusters de bases de données secondaires.
- Pour utiliser Database Activity Streams (DAS) sur Aurora Global Database avec Aurora MySQL 5.7, utilisez un moteur version 2.08 ou supérieure. Pour en savoir plus sur DAS, consultez Surveillance d'Amazon Aurora à l'aide des flux d'activité de base de données.
- Les limites suivantes s'appliquent actuellement à la mise à niveau d'Aurora Global Database :
  - Vous ne pouvez pas appliquer un groupe de paramètres personnalisés au cluster de bases de données globale pendant que vous effectuez une mise à niveau majeure de la version de cette base de données globale Aurora. Vous créez vos groupes de paramètres personnalisés dans chaque région du cluster global et vous les appliquez manuellement aux clusters régionaux après la mise à niveau.
  - Avec une base de données globale Aurora basée sur Aurora MySQL, vous ne pouvez pas effectuer une mise à niveau sur place d'Aurora MySQL version 2 vers la version 3 si le paramètre `lower_case_table_names` est activé. Pour plus d'informations sur les méthodes que vous pouvez utiliser, consultez Mises à niveau de version majeure..
  - Avec Aurora Global Database, vous ne pouvez pas effectuer de mise à niveau de version majeure du moteur de base de données Aurora PostgreSQL si la fonctionnalité d'objectif de point

de reprise (RPO) est activée. Pour en savoir plus sur la fonction RPO, consultez Gestion des RPO pour les bases de données globales basées sur Aurora PostgreSQL–.

- Avec Aurora Global Database, vous ne pouvez pas effectuer une mise à niveau de version mineure de la version 3.01 ou 3.02 d'Aurora MySQL vers la version 3.03 ou supérieure en utilisant le processus standard. Pour plus de détails sur le processus à utiliser, consultez Mise à niveau d'Aurora MySQL par modification de la version du moteur.

Pour plus d'informations sur la mise à niveau d'Aurora Global Database, consultez Création d'une Amazon Aurora Global Database.

- Vous ne pouvez pas arrêter ou démarrer les clusters de bases de données Aurora dans votre base de données globale individuellement. Pour en savoir plus, consultez Arrêt et démarrage d'un cluster de bases de données Amazon Aurora.
- Les instances de base de données de lecteur Aurora attachées au cluster de bases de données Aurora secondaire peuvent redémarrer dans certaines circonstances. Si l'instance de base Région AWS de données d'écriture du serveur principal est redémarrée ou basculée, les instances de base de données de lecture des régions secondaires redémarrent également. Le cluster secondaire est alors indisponible tant que toutes les instances de base de données de lecteur ne sont pas resynchronisées avec l'instance d'enregistreur du cluster de bases de données principal. Le comportement du cluster principal lors du redémarrage ou d'un basculement est le même que celui d'un cluster de bases de données unique non global. Pour plus d'informations, consultez Réplication avec Amazon Aurora.

  Assurez-vous de bien comprendre les impacts qu'elles auront sur votre base de données globale avant d'apporter des modifications à votre cluster de bases de données principal. Pour en savoir plus, veuillez consulter la section Reprise d'une base de données Amazon Aurora globale à partir d'une panne non planifiée.
- La base de données globale Aurora ne prend actuellement pas en charge le `inaccessible-encryption-credentials-recoverable` statut lorsqu'Amazon Aurora perd l'accès à la AWS KMS clé du cluster de bases de données. Dans ce cas, le cluster de bases de données chiffré passe directement à l'état `inaccessible-encryption-credentials` terminal. Pour plus d'informations sur les états, consultez Affichage du statut du cluster de bases de données.
- Secrets Manager ne prend pas en charge Aurora Global Database. Lorsque vous ajoutez une région à une base de données globale, vous devez d'abord désactiver l'intégration Secrets Manager pour l'instance de base de données.
- Les clusters de bases de données basés sur Aurora PostgreSQL qui utilisent Aurora Global Database présentent les limitations suivantes :

- La gestion des caches de clusters n'est pas prise en charge pour les clusters de bases de données Aurora PostgreSQL qui font partie des bases de données globales Aurora.
- Si le cluster de bases de données principal de votre base de données globale est basé sur un réplica d'une instance Amazon RDS PostgreSQL, vous ne pouvez pas créer de cluster secondaire. N'essayez pas de créer un secondaire à partir de ce cluster à l' AWS Management Console aide de l'opération AWS CLI, de ou de l'`CreateDBCluster`API. Vos tentatives expireront et le cluster secondaire ne sera pas créé.

Nous vous recommandons de créer les clusters de bases de données secondaires pour vos bases de données globales à l'aide de la version du moteur de base de données Aurora utilisée pour le cluster principal. Pour plus d'informations, consultez Création d'une base de données Amazon Aurora globale.

Mise en route avec Amazon Aurora Global Database

Pour commencer avec Aurora Global Database, vous devez d'abord choisir quel moteur de base de données Aurora sera utilisé et dans quelles Régions AWS. Seules des versions spécifiques des moteurs de base de données Aurora MySQL et Aurora PostgreSQL dans certaines Régions AWS prennent en charge Aurora Global Database. Pour en obtenir la liste complète, consultez Régions et moteurs de base de données pris en charge pour les bases de données globales Aurora.

Vous pouvez créer une base de données Aurora globale de l'une des manières suivantes :

- Créer une base de données Aurora Global Database avec de nouveaux clusters de bases de données Aurora et instances de base de données Aurora – Pour ce faire, suivez les étapes indiquées dans Création d'une base de données Amazon Aurora globale. Après avoir créé le cluster de bases de données Aurora principal, vous ajoutez ensuite la Région AWS secondaire en suivant les étapes de la section Ajout d'une Région AWS à une base de données Amazon Aurora globale.
- Utiliser un cluster de bases de données Aurora existant, qui prend en charge la fonction de base de données Aurora Global Database et y ajouter une Région AWS – Cela n'est possible que si votre cluster de bases de données Aurora existant utilise une version du moteur de base de données compatible avec le mode global.

  Vérifiez si vous pouvez choisir Add region (Ajouter une région) pour Action dans AWS Management Console quand votre cluster de bases de données Aurora est sélectionné. Si cela est possible, vous pouvez utiliser ce cluster de bases de données Aurora pour votre cluster Aurora

global. Pour plus d'informations, consultez Ajout d'une Région AWS à une base de données Amazon Aurora globale.

Avant de créer une base de données Aurora Global Database, nous vous recommandons de comprendre toutes les exigences de configuration.

Rubriques

- Configuration requise pour une base de données Amazon Aurora globale
- Création d'une base de données Amazon Aurora globale
- Ajout d'une Région AWS à une base de données Amazon Aurora globale
- Création d'un cluster de bases de données Aurora sans tête dans une région secondaire
- Création d'une base de données Amazon Aurora Global Database à partir d'un instantané Aurora ou Amazon RDS

Configuration requise pour une base de données Amazon Aurora globale

Avant de commencer à utiliser votre base de données globale, planifiez les noms des clusters et des instances, les régions AWS, le nombre d'instances et les classes d'instance que vous avez l'intention d'utiliser. Assurez-vous que vos choix sont conformes aux exigences de configuration suivantes.

Une base de données Aurora globale couvre au moins deux Régions AWS. La Région AWS principale prend en charge un cluster de bases de données Aurora qui dispose d'une instance de base de données Aurora de rédacteur. Une Région AWS secondaire exécute un cluster de bases de données Aurora en lecture seule entièrement composé de réplicas Aurora. Au moins une Région AWS secondaire est requise, mais une base de données Aurora globale peut comporter jusqu'à 10 Régions AWS secondaires. Ce tableau répertorie le nombre maximal de clusters de bases de données Aurora, d'instances de base de données Aurora et de réplicas Aurora autorisés dans une base de données Aurora globale.

Description	Région AWS principale	Régions AWS secondaire
Clusters DB Aurora	1	10 (maximum)
Instances de scripteur	1	0

Description	Région AWS principale	Régions AWS secondaire
Instances en lecture seule (réplicas Aurora), par cluster de bases de données Aurora	15 (max)	16 (total)
Instances en lecture seule (maximum autorisé, compte tenu du nombre réel de régions secondaires)	15 - s	s = nombre total d'Régions AWS secondaires

Les exigences spécifiques suivantes s'appliquent aux clusters de bases de données Aurora qui composent une base de données Aurora globale :

- Exigences relatives aux classes d'instance de base de données – Une base de données Aurora globale nécessite des classes d'instance de base de données optimisées pour les applications gourmandes en mémoire. Pour en savoir plus sur les classes d'instance de base de données à mémoire optimisée, consultez Types de classes d'instance de base de données. Nous vous recommandons d'utiliser une classe d'instance db.r5 ou supérieure.
- Exigences relatives aux régions Région AWS : une base de données Aurora globale a besoin d'un cluster de bases de données Aurora principal dans une Région AWS et d'au moins un cluster de bases de données Aurora secondaire dans une Région différente. Vous pouvez créer jusqu'à 10 clusters de bases de données Aurora secondaires (en lecture seule). Chacun doit être dans une région différente. En d'autres termes, deux clusters de bases de données Aurora d'une base de données Aurora globale ne peuvent pas se trouver dans la même Région AWS.

  Pour plus d'informations sur les combinaisons de moteur de base de données Aurora, de version du moteur et de Région AWS que vous pouvez utiliser avec la base de données Aurora Global Database, consultez Régions et moteurs de base de données pris en charge pour les bases de données globales Aurora.
- Exigences relatives à l'attribution de noms : les noms que vous choisissez pour chacun de vos clusters de bases de données Aurora doivent être uniques, dans toutes les Régions AWS. Vous ne pouvez pas utiliser le même nom pour différents clusters de bases de données Aurora, même s'ils se trouvent dans des régions différentes.

- Exigences en matière de capacité pour Aurora Serverless v2 : pour une base de données globale avec Aurora Serverless v2, la capacité minimale recommandée pour le cluster de bases de données dans la Région AWS principale est de 8 ACU.

Vous devez posséder un Compte AWS pour suivre la procédure de cette section. Terminez les tâches de configuration pour travailler avec Amazon Aurora. Pour plus d'informations, consultez Configuration de votre environnement pour Amazon Aurora. Vous devez également effectuer d'autres étapes préliminaires pour créer un cluster de bases de données Aurora. Pour en savoir plus, consultez Création d'un cluster de bases de données Amazon Aurora.

Lorsque vous serez prêt à configurer votre base de données globale, consultez la procédure Création d'une base de données Amazon Aurora globale de création de toutes les ressources nécessaires. Vous pouvez également suivre la procédure Ajout d'une Région AWS à une base de données Amazon Aurora globale pour créer une base de données globale en utilisant un cluster Aurora existant comme cluster principal.

Création d'une base de données Amazon Aurora globale

Pour créer une base de données globale Aurora et ses ressources associées à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API RDS, suivez les étapes ci-dessous.

> **Note**
>
> Si un cluster de bases de données Aurora exécute un moteur de base de données Aurora compatible avec le mode global, vous pouvez opter pour une procédure plus rapide. Dans ce cas, vous pouvez ajouter une autre Région AWS au cluster de bases de données pour créer votre base de données Aurora globale. Pour ce faire, consultez Ajout d'une Région AWS à une base de données Amazon Aurora globale.

Console

La procédure de création d'une base de données Aurora globale commence par la connexion à une Région AWS prenant en charge la fonction de base de données Aurora globale. Pour obtenir la liste complète, consultez Régions et moteurs de base de données pris en charge pour les bases de données globales Aurora.

L'une des étapes suivantes consiste à choisir un cloud privé virtuel (VPC) basé sur Amazon VPC pour votre cluster de bases de données Aurora. Pour utiliser votre propre VPC, nous vous

recommandons de le créer à l'avance afin de pouvoir le sélectionner. Dans le même temps, créez les sous-réseaux associés et, si nécessaire, un groupe de sous-réseaux et un groupe de sécurité. Pour savoir comment procéder, consultez Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement).

Pour obtenir des informations générales sur la création d'un cluster de bases de données Aurora, consultez Création d'un cluster de bases de données Amazon Aurora.

Pour créer une base de données Aurora globale

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Create database (Créer une base de données). Sur la page Create database (Créer une base de données), procédez comme suit :
   - Pour la méthode de création de la base de données, sélectionnez Standard create (Création standard). (Ne sélectionnez pas Easy create [Création facile].)
   - Pour `Engine type` dans la section Options du moteur, choisissez le type de moteur applicable, Aurora (compatible MySQL) ou Aurora (compatible PostgreSQL).
3. Continuez à créer votre base de données globale Aurora en suivant les étapes des procédures suivantes.

Création d'une base de données globale à l'aide de Aurora MySQL

Les étapes suivantes s'appliquent à toutes les versions d'Aurora MySQL.

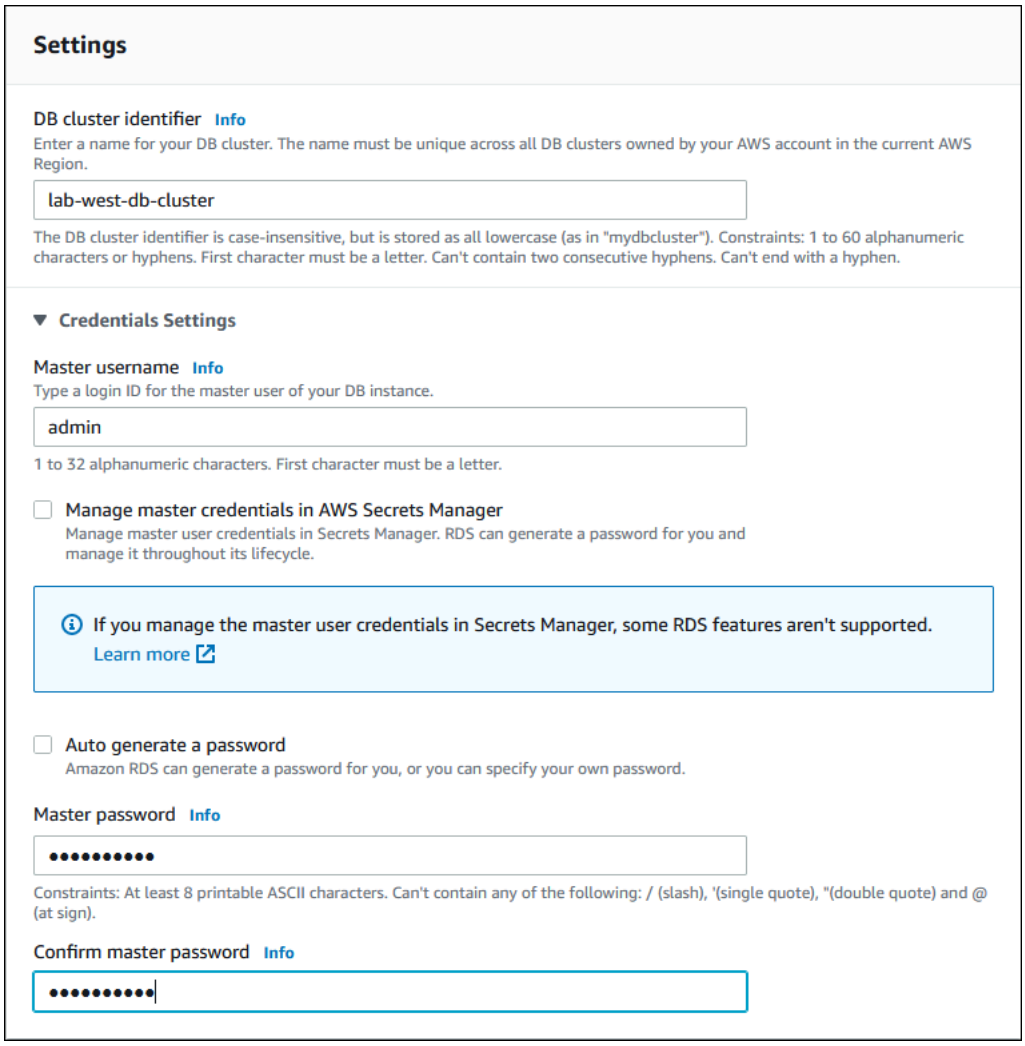
Pour créer une base de données globale Aurora avec Aurora MySQL

Remplir la page Créer une base de données.

1. Choisissez les options de moteur suivantes :
   - Pour Engine version (Version du moteur), sélectionnez la version d'Aurora MySQL que vous souhaitez utiliser pour votre base de données Aurora globale.
2. Pour Modèles, sélectionnez Production. Alternativement, vous pouvez choisir Dev/Test si cela s'applique à votre cas d'utilisation. N'utilisez pas Dev/Test dans les environnements de production.
3. Sous Paramètres, procédez comme suit :

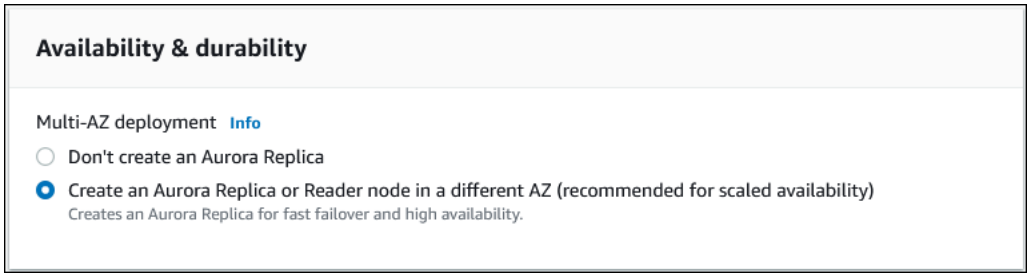
a. Spécifiez un nom significatif pour l'identifiant de cluster de bases de données. Lorsque vous aurez terminé de créer la base de données Aurora globale, ce nom identifie le cluster de bases de données principal.

b. Saisissez le mot de passe de votre choix pour le compte d'utilisateur `admin` de l'instance de base de données, ou laissez Aurora en générer un pour vous. Si vous choisissez de générer automatiquement un mot de passe, vous disposez d'une option permettant de copier le mot de passe.



4. Pour DB instance class (Classe d'instance de base de données), choisissez `db.r5.large` ou une autre classe d'instance de base de données à mémoire optimisée. Nous vous recommandons d'utiliser une classe d'instance db.r5 ou supérieure.

5. Pour Availability & durability (Disponibilité et durabilité), nous vous recommandons de laisser Aurora créer un réplica Aurora dans une zone de disponibilité différente en votre nom. Si vous ne créez pas de réplica Aurora maintenant, vous devrez le faire ultérieurement.



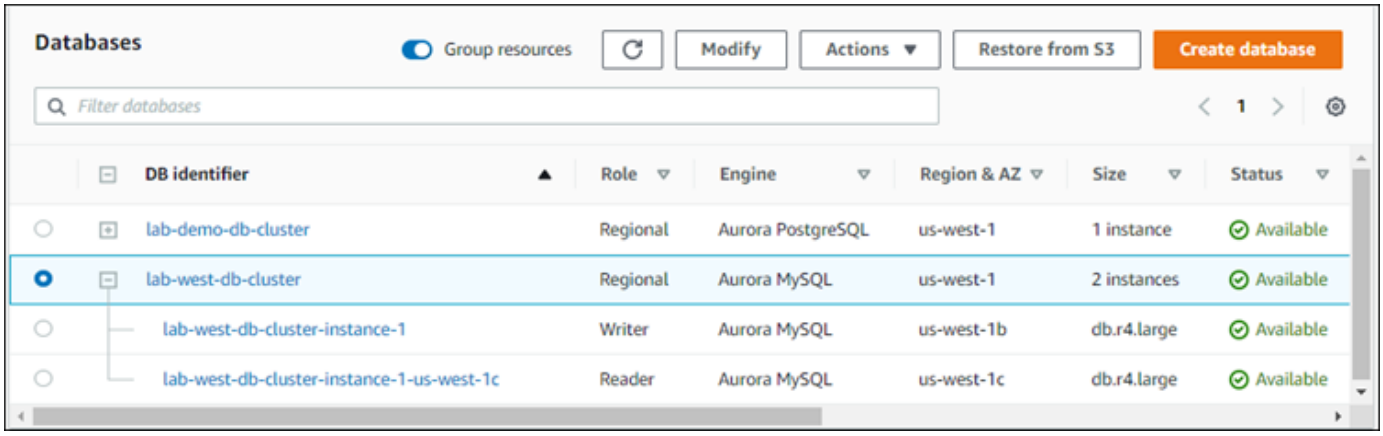
6. Pour Connectivité, choisissez le cloud privé virtuel (VPC) en fonction du Amazon VPC qui définit l'environnement de mise en réseau virtuel pour cette instance de base de données. Vous pouvez choisir les valeurs par défaut pour simplifier cette tâche.
7. Spécifiez les paramètres Database authentication (Authentification de base de données). Pour simplifier le processus, vous pouvez choisir Password authentication (Authentification par mot de passe) maintenant et configurer Gestion des identités et des accès AWS (IAM) ultérieurement.
8. Sous Configuration supplémentaire, procédez comme suit :

   a. Spécifiez un nom pour Nom de base de données initial pour créer l'instance de base de données Aurora principale pour ce cluster. Il s'agit du nœud de scripteur pour le cluster de bases de données Aurora principal.

      Laissez les valeurs par défaut sélectionnées pour le groupe de paramètres de cluster de bases de données et le groupe de paramètres de base de données, sauf si vous souhaitez utiliser vos propres groupes de paramètres personnalisés.

   b. Désactivez la case à cocher Enable backtrack (Activer le retour sur trace) si elle est activée. Les bases de données Aurora globales ne prennent pas en charge le retour sur trace. Vous pouvez accepter tous les autres paramètres par défaut pour Addition configuration (Configuration supplémentaire).
9. Choisissez Create database (Créer une base de données).

   Le processus de création de l'instance de base de données Aurora, de son réplica Aurora et du cluster de bases de données Aurora par Aurora peut prendre plusieurs minutes. Vous pouvez savoir quand le cluster de bases de données Aurora est prêt à être utilisé en tant que cluster de bases de données principal dans une base de données Aurora globale par son état. Si tel est le cas, son état et celui du scripteur et du nœud de réplica sont disponibles, comme illustré ci-dessous.



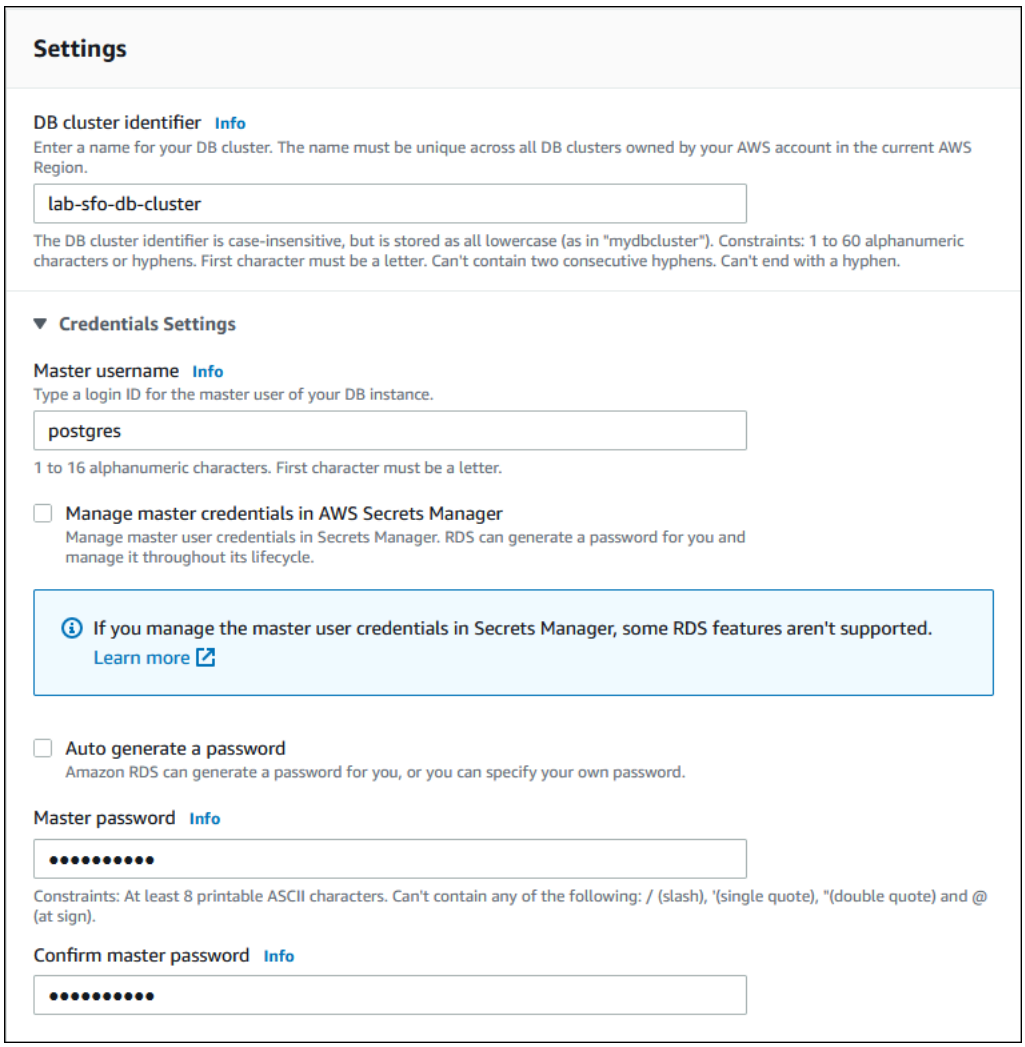
Lorsque votre cluster de bases de données principal est disponible, créez la base de données Aurora globale en y ajoutant un cluster secondaire. Pour cela, suivez les étapes de Ajout d'une Région AWS à une base de données Amazon Aurora globale.

Création d'une base de données globale à l'aide de Aurora PostgreSQL

Pour créer une base de données globale Aurora avec Aurora PostgreSQL

Remplir la page Créer une base de données.

1. Choisissez les options de moteur suivantes :
   - Pour Engine version (Version du moteur), sélectionnez la version d'Aurora PostgreSQL que vous souhaitez utiliser pour votre base de données Aurora globale.
2. Pour Modèles, sélectionnez Production. Alternativement, vous pouvez choisir Dev/Test si cela s'applique à votre cas d'utilisation. N'utilisez pas Dev/Test dans les environnements de production.
3. Sous Paramètres, procédez comme suit :
   a. Spécifiez un nom significatif pour l'identifiant de cluster de bases de données. Lorsque vous aurez terminé de créer la base de données Aurora globale, ce nom identifie le cluster de bases de données principal.
   b. Saisissez le mot de passe de votre choix pour le compte d'administrateur par défaut du cluster de bases de données, ou laissez Aurora en générer un pour vous. Si vous choisissez de générer automatiquement un mot de passe, vous disposez d'une option permettant de copier le mot de passe.



4. Pour DB instance class (Classe d'instance de base de données), choisissez `db.r5.large` ou une autre classe d'instance de base de données à mémoire optimisée. Nous vous recommandons d'utiliser une classe d'instance db.r5 ou supérieure.
5. Pour Disponibilité et durabilité, nous vous recommandons d'opter pour la création par Aurora d'un réplica Aurora dans une zone de disponibilité différente en votre nom. Si vous ne créez pas de réplica Aurora maintenant, vous devrez le faire ultérieurement.
6. Pour Connectivité, choisissez le cloud privé virtuel (VPC) en fonction du Amazon VPC qui définit l'environnement de mise en réseau virtuel pour cette instance de base de données. Vous pouvez choisir les valeurs par défaut pour simplifier cette tâche.
7. (Facultatif) Remplissez les paramètres Database authentication (Authentification de la base de données). L'authentification par mot de passe est toujours activée. Pour simplifier le processus, vous pouvez ignorer cette section et configurer l'authentification IAM ou le mot de passe et Kerberos plus tard.

8. Sous Configuration supplémentaire, procédez comme suit :

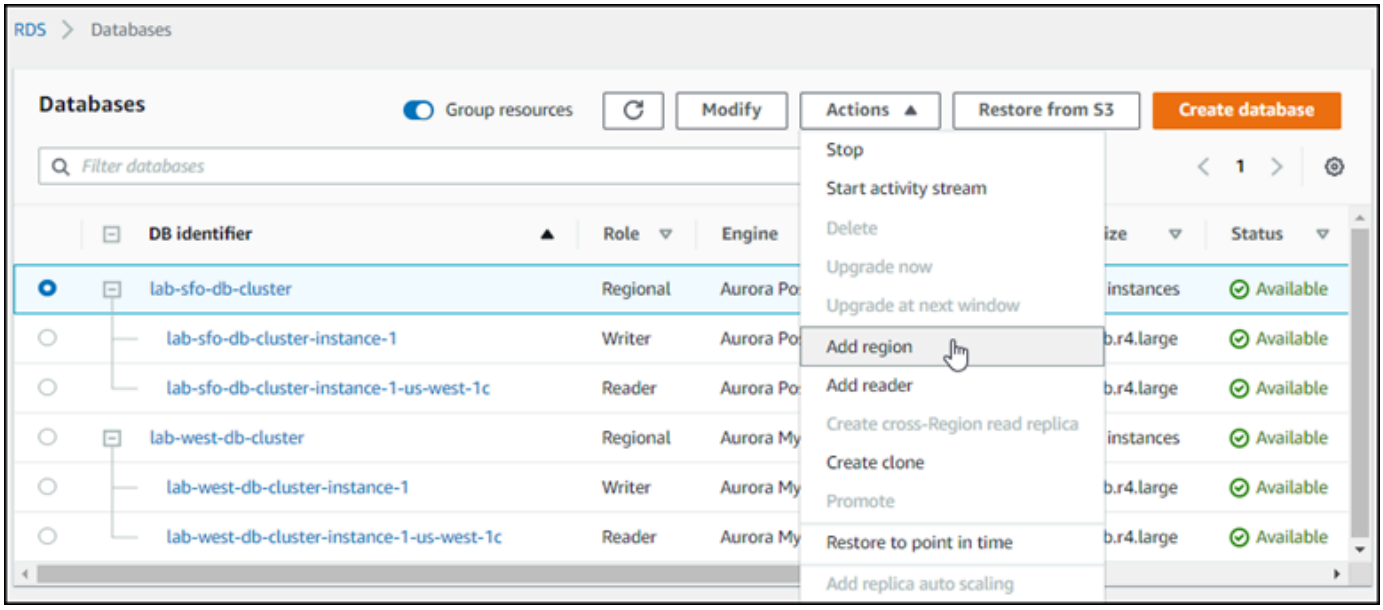
   a. Spécifiez un nom pour Nom de base de données initial pour créer l'instance de base de données Aurora principale pour ce cluster. Il s'agit du nœud de scripteur pour le cluster de bases de données Aurora principal.

      Laissez les valeurs par défaut sélectionnées pour le groupe de paramètres de cluster de bases de données et le groupe de paramètres de base de données, sauf si vous souhaitez utiliser vos propres groupes de paramètres personnalisés.

   b. Acceptez tous les autres paramètres par défaut pour Additional configuration (Configuration supplémentaire), tels que le chiffrement, les exportations de journaux, etc.

9. Choisissez Create database (Créer une base de données).

   Le processus de création de l'instance de base de données Aurora, de son réplica Aurora et du cluster de bases de données Aurora par Aurora peut prendre plusieurs minutes. Lorsque le cluster est prêt à être utilisé, le cluster de bases de données Aurora et ses nœuds d'enregistreur et de réplica affichent tous l'état Available (Disponible). Cela deviendra le cluster de bases de données principal de votre base de données Aurora globale, une fois le cluster secondaire ajouté.



Lorsque le cluster de bases de données principal est créé et disponible, suivez les étapes décrites sous pour créer un ou plusieurs clusters secondaire Ajout d'une Région AWS à une base de données Amazon Aurora globale.

AWS CLI

Les commandes AWS CLI des procédures suivantes accomplissent les tâches suivantes :

1. Créer une base de données Aurora globale, en lui donnant un nom et en spécifiant le type de moteur de base de données Aurora que vous prévoyez d'utiliser.
2. Créer un cluster de bases de données Aurora pour la base de données Aurora globale.
3. Créer l'instance de base de données Aurora pour le cluster. Il s'agit du cluster principal de la base de données Aurora pour la base de données globale.
4. Créer une deuxième instance de base de données pour le cluster de bases de données Aurora. Il s'agit d'un lecteur qui complète le cluster de bases de données Aurora.
5. Créer un second cluster de bases de données Aurora dans une autre région, puis l'ajouter à votre base de données Aurora globale, en suivant les étapes décrites dans Ajout d'une Région AWS à une base de données Amazon Aurora globale.

Suivez la procédure pour votre moteur de base de données Aurora.

Création d'une base de données globale à l'aide de Aurora MySQL

Pour créer une base de données globale Aurora avec Aurora MySQL

1. Utilisez la commande CLI `create-global-cluster`, en transmettant le nom de la Région AWS, le moteur de base de données Aurora et la version.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-global-cluster --region primary_region \
       --global-cluster-identifier global_database_id \
       --engine aurora-mysql \
       --engine-version version # optional
   ```

   Pour Windows :

   ```
   aws rds create-global-cluster ^
       --global-cluster-identifier global_database_id ^
       --engine aurora-mysql ^
       --engine-version version # optional
   ```

Cela crée une base de données Aurora globale « vide », avec juste un nom (identifiant) et un moteur de base de données Aurora. Il peut se passer quelques minutes avant que la base de données Aurora globale ne soit disponible. Utilisez donc la commande `describe-global-clusters` de l'interface de ligne de commande avant de passer à l'étape suivante, afin de vérifier qu'elle est bien disponible.

```
aws rds describe-global-clusters --region primary_region --global-cluster-
identifier global_database_id
```

Une fois la base de données Aurora globale disponible, vous pouvez créer son cluster de bases de données Aurora principal.

2. Pour créer un cluster de bases de données Aurora principal, utilisez la commande `create-db-cluster` de l'interface de ligne de commande. Incluez le nom de votre base de données globale Aurora à l'aide du paramètre `--global-cluster-identifier`.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-db-cluster \
     --region primary_region \
     --db-cluster-identifier primary_db_cluster_id \
     --master-username userid \
     --master-user-password password \
     --engine aurora-mysql \
     --engine-version version \
     --global-cluster-identifier global_database_id
   ```

   Pour Windows :

   ```
   aws rds create-db-cluster ^
     --region primary_region ^
     --db-cluster-identifier primary_db_cluster_id ^
     --master-username userid ^
     --master-user-password password ^
     --engine aurora-mysql ^
     --engine-version version ^
     --global-cluster-identifier global_database_id
   ```

Utilisez la commande de l'AWS CLI `describe-db-clusters` pour confirmer que le cluster de bases de données Aurora est prêt. Pour isoler un cluster de bases de données Aurora spécifique, utilisez le paramètre `--db-cluster-identifier`. Ou vous pouvez ne pas inclure de nom du cluster de bases de données Aurora dans la commande pour obtenir des détails sur tous vos clusters de bases de données Aurora dans la région donnée.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier primary_db_cluster_id
```

Lorsque la réponse affiche `"Status": "available"` pour le cluster, il est prêt à l'emploi.

3. Créer l'instance de base de données pour votre cluster Aurora principal. Pour ce faire, utilisez la commande `create-db-instance` de l'interface de ligne de commande. Donnez à la commande le nom de votre cluster de bases de données Aurora et spécifiez les détails de configuration de l'instance. Vous n'avez pas besoin de passer les paramètres `--master-username` et `--master-user-password` dans la commande, car elle obtient ceux du cluster de bases de données Aurora.

   Pour `--db-instance-class`, vous pouvez utiliser uniquement celles des classes à mémoire optimisée, par exemple `db.r5.large`. Nous vous recommandons d'utiliser une classe d'instance db.r5 ou supérieure. Pour en savoir plus sur ces classes, consultez Types de classes d'instance de base de données.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-db-instance \
     --db-cluster-identifier primary_db_cluster_id \
     --db-instance-class instance_class \
     --db-instance-identifier db_instance_id \
     --engine aurora-mysql \
     --engine-version version \
     --region primary_region
   ```

   Pour Windows :

   ```
   aws rds create-db-instance ^
     --db-cluster-identifier primary_db_cluster_id ^
     --db-instance-class instance_class ^
     --db-instance-identifier db_instance_id ^
   ```

```
  --engine aurora-mysql ^
  --engine-version version ^
  --region primary_region
```

L'opération `create-db-instance` peut prendre du temps. Vérifiez l'état pour voir si l'instance de base de données Aurora est disponible avant de continuer.

```
aws rds describe-db-clusters --db-cluster-identifier primary_db_cluster_id
```

Lorsque la commande renvoie l'état `available`, vous pouvez créer une autre instance de base de données Aurora pour votre cluster de bases de données principal. Il s'agit de l'instance de lecteur (le réplica Aurora) pour le cluster de bases de données Aurora.

4. Pour créer une autre instance de base de données Aurora pour le cluster, utilisez la commande `create-db-instance` de l'interface de ligne de commande.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-db-instance \
     --db-cluster-identifier primary_db_cluster_id \
     --db-instance-class instance_class \
     --db-instance-identifier replica_db_instance_id \
     --engine aurora-mysql
   ```

   Pour Windows :

   ```
   aws rds create-db-instance ^
     --db-cluster-identifier primary_db_cluster_id ^
     --db-instance-class instance_class ^
     --db-instance-identifier replica_db_instance_id ^
     --engine aurora-mysql
   ```

Lorsque l'instance de base de données est disponible, la réplication commence du nœud du scripteur vers le réplica. Avant de continuer, vérifiez que l'instance de base de données est disponible avec la commande `describe-db-instances` de l'interface de ligne de commande.

À ce stade, vous disposez d'une base de données Aurora globale avec son cluster Aurora principal contenant une instance de scripteur et un réplica Aurora. Vous pouvez désormais ajouter un cluster de bases de données Aurora en lecture seule dans une autre région pour compléter votre base de

données Aurora globale. Pour ce faire, suivez les étapes de Ajout d'une Région AWS à une base de données Amazon Aurora globale.

Création d'une base de données globale à l'aide de Aurora PostgreSQL

Lorsque vous créez des objets Aurora pour une base de données Aurora globale à l'aide des commandes suivantes, quelques minutes peuvent s'écouler avant que chacun soit disponible. Après avoir terminé l'exécution d'une commande donnée, nous vous recommandons de vérifier l'état de l'objet Aurora spécifique pour vous assurer que cet état est disponible.

Pour ce faire, utilisez la commande `describe-global-clusters` de l'interface de ligne de commande.

```
aws rds describe-global-clusters --region primary_region
    --global-cluster-identifier global_database_id
```

Pour créer une base de données globale Aurora avec Aurora PostgreSQL

1. Utilisez la commande `create-global-cluster` de l'interface de ligne de commande.

   Pour Linux, macOS ou Unix :

   ```
   aws rds create-global-cluster --region primary_region \
       --global-cluster-identifier global_database_id \
       --engine aurora-postgresql \
       --engine-version version # optional
   ```

   Pour Windows :

   ```
   aws rds create-global-cluster ^
       --global-cluster-identifier global_database_id ^
       --engine aurora-postgresql ^
       --engine-version version # optional
   ```

   Une fois la base de données Aurora globale disponible, vous pouvez créer son cluster de bases de données Aurora principal.

2. Pour créer un cluster de bases de données Aurora principal, utilisez la commande `create-db-cluster` de l'interface de ligne de commande. Incluez le nom de votre base de données globale Aurora à l'aide du paramètre `--global-cluster-identifier`.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
  --region primary_region \
  --db-cluster-identifier primary_db_cluster_id \
  --master-username userid \
  --master-user-password password \
  --engine aurora-postgresql \
  --engine-version version \
  --global-cluster-identifier global_database_id
```

Pour Windows :

```
aws rds create-db-cluster ^
  --region primary_region ^
  --db-cluster-identifier primary_db_cluster_id ^
  --master-username userid ^
  --master-user-password password ^
  --engine aurora-postgresql ^
  --engine-version version ^
  --global-cluster-identifier global_database_id
```

Vérifiez que le cluster de bases de données Aurora est prêt. Lorsque la réponse de la commande suivante affiche `"Status": "available"` pour le cluster de bases de données Aurora, vous pouvez continuer.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier primary_db_cluster_id
```

3. Créer l'instance de base de données pour votre cluster Aurora principal. Pour ce faire, utilisez la commande `create-db-instance` de l'interface de ligne de commande.

   Passez le nom de votre cluster de bases de données Aurora avec le paramètre `--db-cluster-identifier`

   Vous n'avez pas besoin de passer les paramètres `--master-username` et `--master-user-password` dans la commande, car elle obtient ceux du cluster de bases de données Aurora.

   Pour `--db-instance-class`, vous pouvez utiliser uniquement celles des classes à mémoire optimisée, par exemple `db.r5.large`. Nous vous recommandons d'utiliser une classe

d'instance db.r5 ou supérieure. Pour en savoir plus sur ces classes, consultez Types de classes d'instance de base de données.

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier db_instance_id \
  --engine aurora-postgresql \
  --engine-version version \
  --region primary_region
```

Pour Windows :

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier db_instance_id ^
  --engine aurora-postgresql ^
  --engine-version version ^
  --region primary_region
```

4. Vérifiez l'état de l'instance de base de données Aurora avant de continuer.

```
aws rds describe-db-clusters --db-cluster-identifier primary_db_cluster_id
```

Si la réponse indique que l'état de l'instance de base de données Aurora est `available`, vous pouvez créer une autre instance Aurora pour votre cluster de bases de données principal.

5. Pour créer un réplica Aurora pour le cluster de bases de données Aurora, utilisez la commande `create-db-instance` de l'interface de ligne de commande.

Pour Linux, macOS ou Unix :

```
aws rds create-db-instance \
  --db-cluster-identifier primary_db_cluster_id \
  --db-instance-class instance_class \
  --db-instance-identifier replica_db_instance_id \
  --engine aurora-postgresql
```

Pour Windows :

```
aws rds create-db-instance ^
  --db-cluster-identifier primary_db_cluster_id ^
  --db-instance-class instance_class ^
  --db-instance-identifier replica_db_instance_id ^
  --engine aurora-postgresql
```

Lorsque l'instance de base de données est disponible, la réplication commence du nœud du scripteur vers le réplica. Avant de continuer, vérifiez que l'instance de base de données est disponible avec la commande `describe-db-instances` de l'interface de ligne de commande.

Votre base de données Aurora globale existe, mais elle ne dispose que de sa région principale avec un cluster de bases de données Aurora composé d'une instance de scripteur et d'un réplica Aurora. Vous pouvez désormais ajouter un cluster de bases de données Aurora en lecture seule dans une autre région pour compléter votre base de données Aurora globale. Pour ce faire, suivez les étapes de Ajout d'une Région AWS à une base de données Amazon Aurora globale.

API RDS

Pour créer une base de données globale Aurora avec l'API RDS, exécutez l'opération CreateGlobalCluster.

Ajout d'une Région AWS à une base de données Amazon Aurora globale

Vous pouvez utiliser la procédure suivante pour ajouter un cluster secondaire supplémentaire à une base de données globale existante. Vous pouvez également créer une base de données globale à partir d'un cluster de bases de données Aurora autonome en utilisant cette procédure pour ajouter la première région AWS secondaire.

Une base de données Aurora globale a besoin d'au moins un cluster de bases de données Aurora secondaire dans une Région AWS différente de celle du cluster de bases de données Aurora principal. Vous pouvez attacher jusqu'à 10 clusters de bases de données secondaires à une base de données Aurora globale. Répétez la procédure suivante pour chaque nouveau cluster de bases de données secondaire. Pour chaque cluster de bases de données secondaire que vous ajoutez à votre base de données Aurora globale, retranchez un pour obtenir le nombre de réplicas Aurora autorisés pour le cluster de bases de données principal.

Par exemple, si votre base de données Aurora globale comporte 10 régions secondaires, votre cluster de bases de données principal ne peut avoir que cinq réplicas Aurora (au lieu de 15). Pour plus d'informations, consultez Configuration requise pour une base de données Amazon Aurora globale.

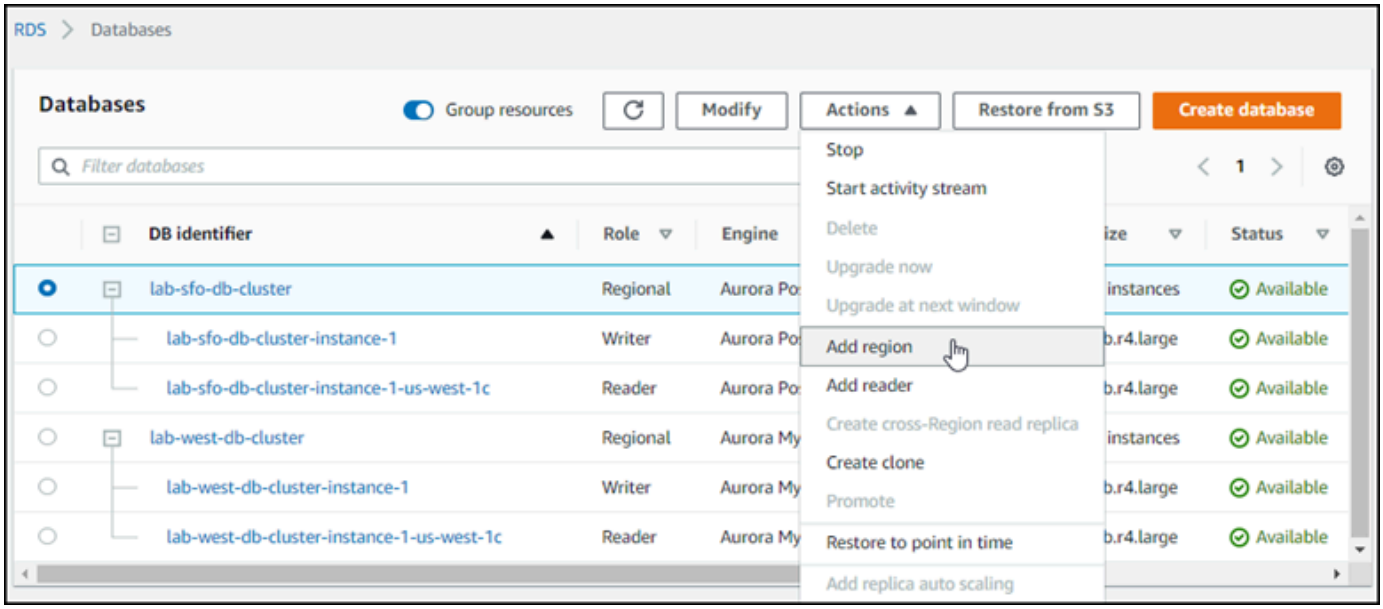
Le nombre de réplicas Aurora (instances de lecteur) dans le cluster de bases de données principal détermine le nombre de clusters de bases de données secondaires que vous pouvez ajouter. Le nombre total d'instances de lecteur dans le cluster de bases de données primaire plus le nombre de clusters secondaires ne peut pas dépasser le nombre de 15. Par exemple, si vous avez 14 instances de lecteur dans le cluster de bases de données primaire et un cluster secondaire, vous ne pouvez pas ajouter de cluster secondaire supplémentaire à la base de données globale.

> **Note**
>
> Pour Aurora MySQL version 3, lorsque vous créez un cluster secondaire, assurez-vous que la valeur de `lower_case_table_names` correspond à la valeur du cluster principal. Ce paramètre est un paramètre de base de données qui affecte la façon dont le serveur gère la sensibilité à la casse de l'identifiant. Pour plus d'informations sur les paramètres de la base de données, consultez Groupes de paramètres pour Amazon Aurora.
> Lorsque vous créez un cluster secondaire, nous vous recommandons d'utiliser la même version du moteur de base de données pour le principal et le secondaire. Si nécessaire, mettez à niveau la version principale pour qu'elle soit identique à la version secondaire. Pour plus d'informations, consultez Compatibilité des niveaux de correctif pour les bascules ou basculements gérés entre régions.

Console

Pour ajouter une Région AWS à une base de données Aurora globale

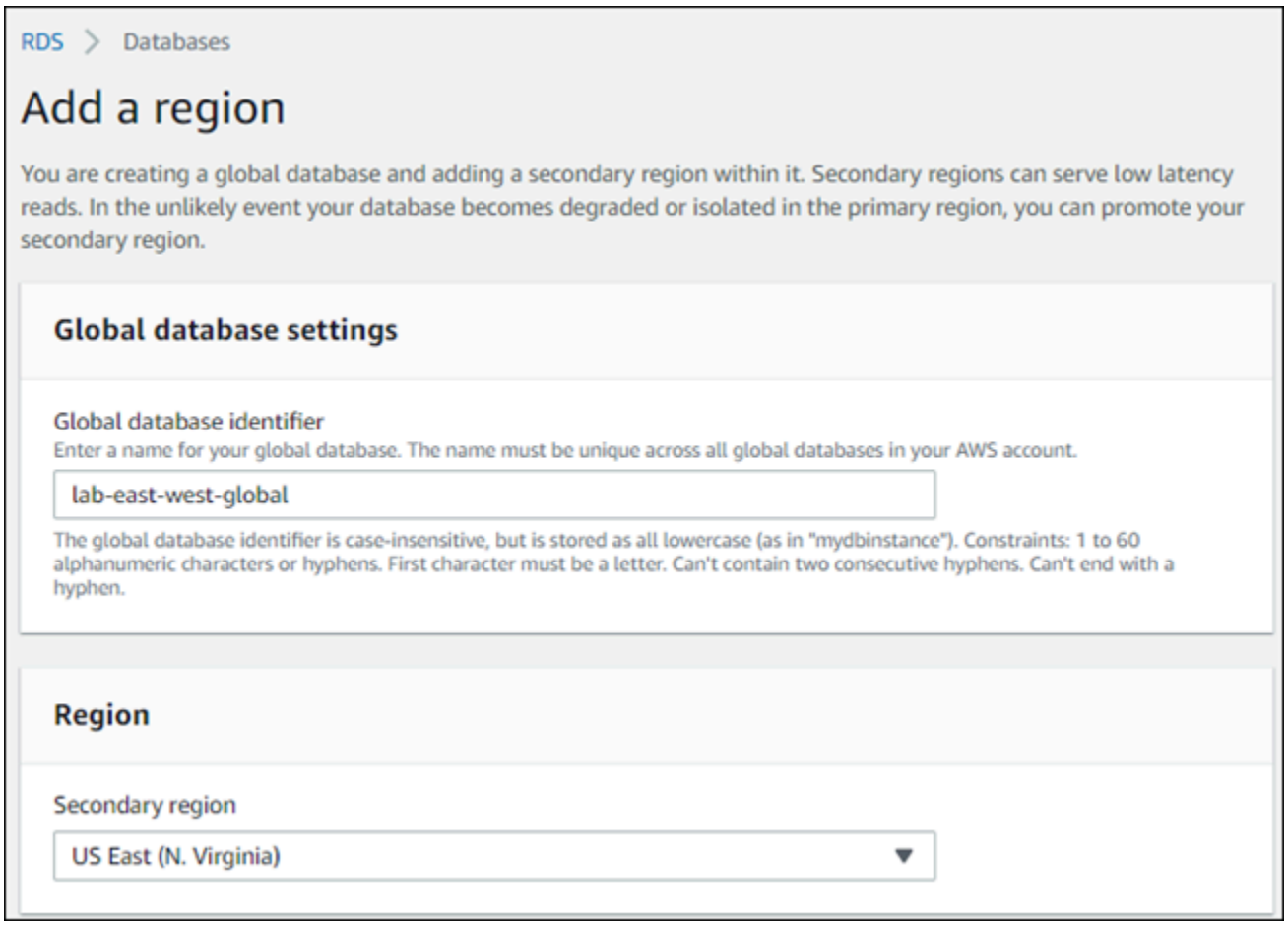
1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation de la AWS Management Console, choisissez Bases de données.
3. Sélectionnez la base de données Aurora globale qui a besoin d'un cluster de bases de données Aurora secondaire. Assurez-vous que le cluster de bases de données Aurora principal est `Available`.
4. Pour Actions, choisissez Ajouter une région AWS.



5. Sur la page Add a region (Ajouter une Région), choisissez la Région AWS secondaire.

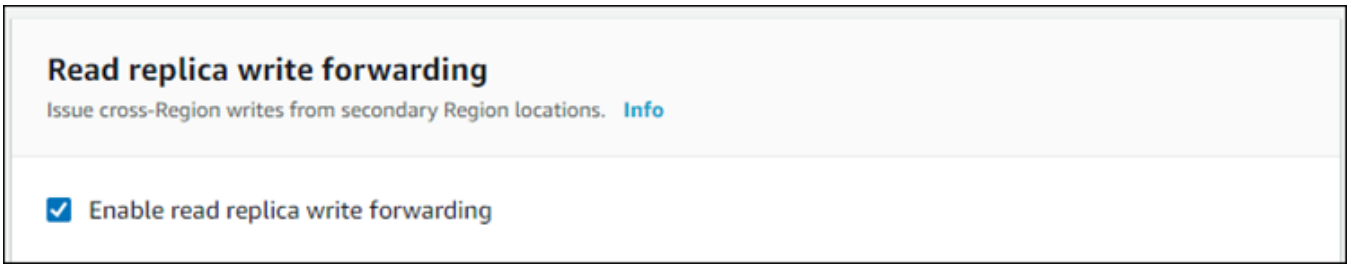
   Vous ne pouvez pas choisir une Région AWS qui possède déjà un cluster de bases de données Aurora secondaire pour la même base de données Aurora globale. De plus, il ne peut pas s'agir de la même région que le cluster de bases de données Aurora principal.

   > **Note**
   >
   > Les bases de données globales Babelfish pour Aurora PostgreSQL ne fonctionnent dans les régions secondaires que si les paramètres qui contrôlent les préférences de Babelfish sont activés dans ces régions. Pour plus d'informations, consultez Paramètres du groupe de paramètres de cluster de bases de données pour Babelfish.



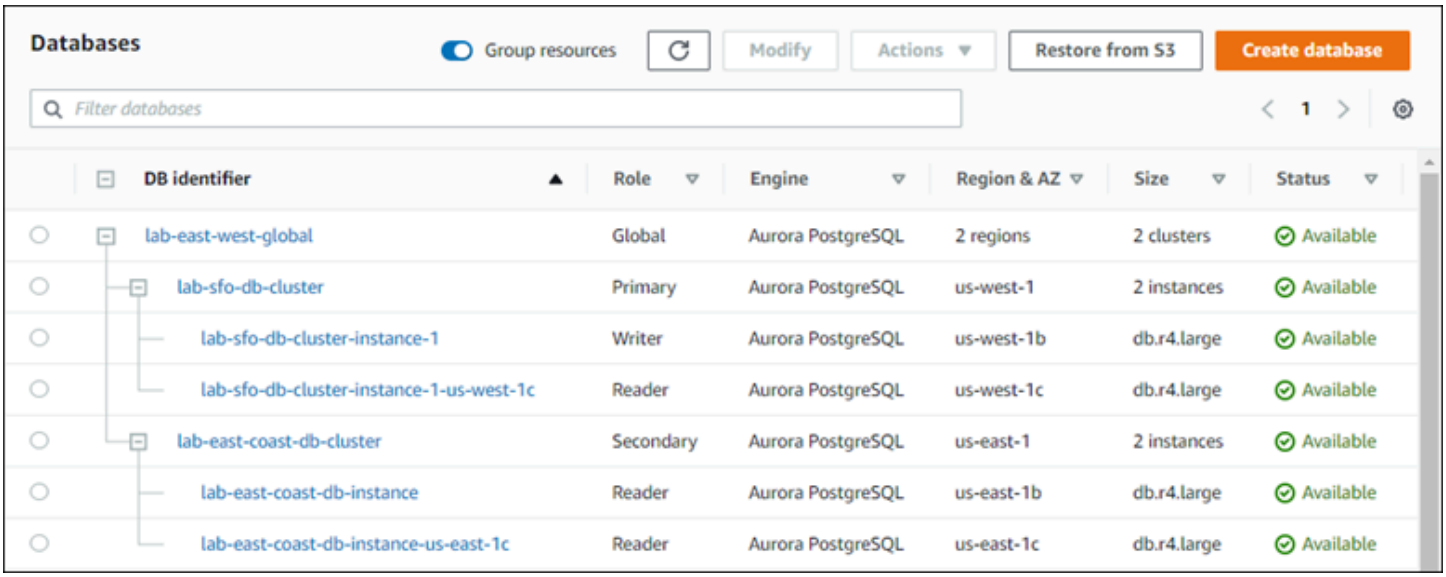
6. Remplissez les champs restants pour le cluster Aurora secondaire dans la nouvelle région AWS. Il s'agit des mêmes options de configuration que pour n'importe quelle instance de cluster de bases de données Aurora, à l'exception de l'option suivante pour les bases de données Aurora globales basées sur Aurora MySQL– :

   - Activer le transfert d'écriture du réplica en lecture – Ce paramètre facultatif permet aux clusters de bases de données secondaires de votre base de données Aurora globale de transférer les opérations d'écriture vers le cluster principal. Pour plus d'informations, consultez Utilisation du transfert d'écriture dans une base de données globale Amazon Aurora.



7. Choisissez Ajouter une région AWS.

Une fois la région ajoutée à votre base de données Aurora globale, vous pouvez la voir dans la liste des bases de données dans l'AWS Management Console comme l'illustre la capture d'écran.



AWS CLI

Pour ajouter une Région AWS secondaire à une base de données Aurora globale

Pour ajouter un cluster secondaire à votre base de données globale à l'aide de la CLI, vous devez déjà disposer de l'objet conteneur du cluster global. Si vous n'avez pas encore exécuté la commande `create-global-cluster`, consultez la procédure spécifique à la CLI dans Création d'une base de données Amazon Aurora globale.

1. Utilisez la commande `create-db-cluster` de l'interface de ligne de commande avec le nom (`--global-cluster-identifier`) de votre base de données Aurora globale. Pour les autres paramètres, procédez comme suit :
2. Pour `--region`, choisissez une Région AWS autre que votre Région Aurora principale.
3. Choisissez des valeurs spécifiques pour les paramètres `--engine` et `--engine-version`. Ces valeurs sont les mêmes que celles du cluster de bases de données Aurora principal de votre base de données Aurora globale.
4. Pour un cluster chiffré, spécifiez votre Région AWS principale comme `--source-region` pour le chiffrement.

L'exemple suivant crée un nouveau cluster de bases de données Aurora et l'attache à une base de données Aurora globale en tant que cluster Aurora secondaire en lecture seule. Dans la dernière étape, une instance de base de données Aurora est ajoutée au nouveau cluster Aurora.

Pour Linux, macOS ou Unix :

```
aws rds --region secondary_region \
  create-db-cluster \
    --db-cluster-identifier secondary_cluster_id \
    --global-cluster-identifier global_database_id \
    --engine aurora-mysql | aurora-postgresql \
    --engine-version version

aws rds --region secondary_region \
  create-db-instance \
    --db-instance-class instance_class \
    --db-cluster-identifier secondary_cluster_id \
    --db-instance-identifier db_instance_id \
    --engine aurora-mysql | aurora-postgresql
```

Pour Windows :

```
aws rds --region secondary_region ^
  create-db-cluster ^
    --db-cluster-identifier secondary_cluster_id ^
    --global-cluster-identifier global_database_id_id ^
    --engine aurora-mysql | aurora-postgresql ^
    --engine-version version

aws rds --region secondary_region ^
  create-db-instance ^
    --db-instance-class instance_class ^
    --db-cluster-identifier secondary_cluster_id ^
    --db-instance-identifier db_instance_id ^
    --engine aurora-mysql | aurora-postgresql
```

API RDS

Pour ajouter une nouvelle Région AWS à une base de données Aurora globale avec l'API RDS, exécutez l'opération CreateDBCluster. Spécifiez l'identifiant de la base de données globale existante à l'aide du paramètre `GlobalClusterIdentifier`.

Création d'un cluster de bases de données Aurora sans tête dans une région secondaire

Bien qu'une base de données Aurora globale nécessite au moins un cluster de bases de données Aurora secondaire dans une Région AWS différente de la principale, vous pouvez utiliser une configuration sans périphériques pour le cluster secondaire. Un cluster de bases de données Aurora secondaire sans tête est un cluster sans instance de base de données. Ce type de configuration peut réduire les dépenses d'une base de données Aurora globale. Dans un cluster de bases de données Aurora, le calcul et le stockage sont découplés. Sans l'instance de base de données, vous êtes facturé pour le stockage, mais pas pour le calcul. Si la configuration est correcte, le volume de stockage d'un cluster secondaire sans tête reste synchronisé avec le cluster de bases de données Aurora principal.

Ajoutez le cluster secondaire comme vous le faites normalement lors de la création d'une base de données Aurora globale. Si vous créez tous les clusters de la base de données globale, suivez la procédure décrite dans Création d'une base de données Amazon Aurora globale. Si vous avez déjà un cluster de bases de données à utiliser comme cluster principal, suivez la procédure décrite dans Ajout d'une Région AWS à une base de données Amazon Aurora globale.

Après que le cluster de bases de données Aurora principal commence la réplication vers le cluster secondaire, supprimez l'instance de base de données en lecture seule Aurora du cluster de bases de données Aurora secondaire. Ce cluster secondaire est désormais considéré comme « sans tête », car il n'a plus d'instance de base de données. Même sans instance de base de données dans le cluster secondaire, Aurora reste synchronisé avec le cluster de bases de données Aurora principal.

> **Warning**
>
> Avec Aurora PostgreSQL, pour créer un cluster sans tête dans une Région AWS secondaire, utilisez l'API AWS CLI ou RDS pour ajouter la Région AWS secondaire. Ignorez l'étape de création de l'instance de base de données de lecteur pour le cluster secondaire. Actuellement, la création d'un cluster sans tête n'est pas prise en charge dans la console RDS. Pour connaître les procédures CLI et API à utiliser, consultez Ajout d'une Région AWS à une base de données Amazon Aurora globale.
>
> Si votre base de données globale utilise une version de moteur Aurora PostgreSQL antérieure à 13.4, 12.8 ou 11.13, la création d'une instance de base de données de lecteur dans une région secondaire, puis sa suppression peuvent entraîner un problème de vacuum Aurora PostgreSQL sur l'instance de base de données d'enregistreur de la région principale. Si vous rencontrez ce problème, redémarrez l'instance de base de données d'enregistreur

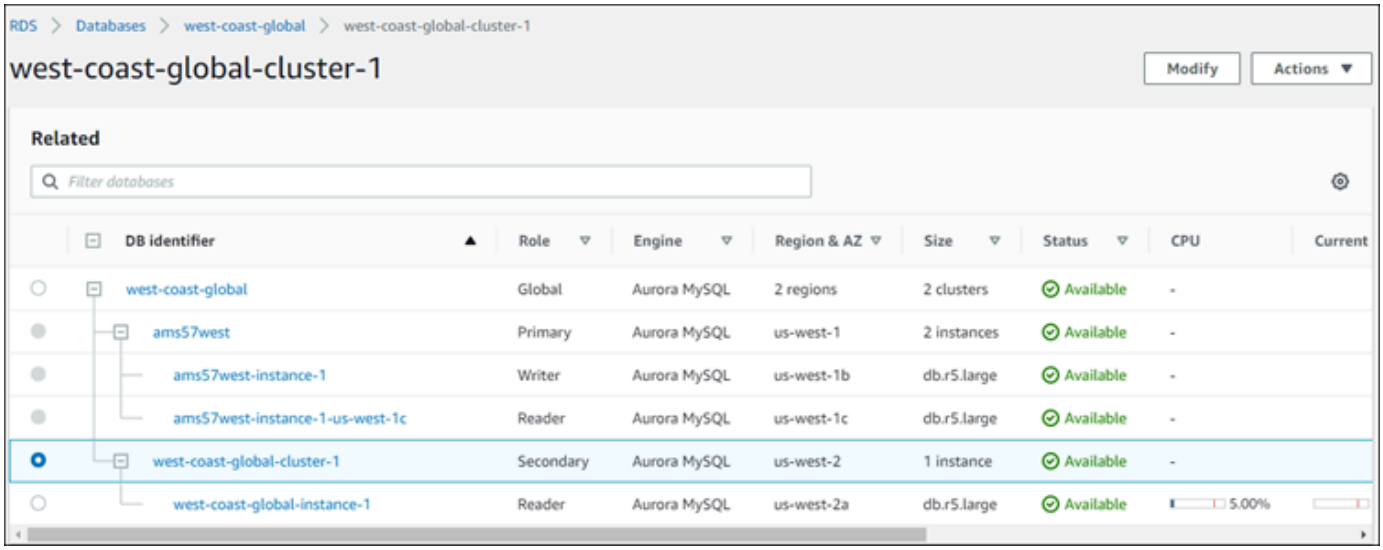
de la région principale après avoir supprimé l'instance de base de données de lecteur de la région secondaire.

Ajouter un cluster de bases de données Aurora secondaire sans tête à votre base de données Aurora globale

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation de la AWS Management Console, choisissez Bases de données.
3. Sélectionnez la base de données Aurora globale qui a besoin d'un cluster de bases de données Aurora secondaire. Assurez-vous que le cluster de bases de données Aurora principal est `Available`.
4. Pour Actions, choisissez Ajouter une région AWS.
5. Sur la page Add a region (Ajouter une Région), choisissez la Région AWS secondaire.

   Vous ne pouvez pas choisir une Région AWS qui possède déjà un cluster de bases de données Aurora secondaire pour la même base de données Aurora globale. De plus, il ne peut pas s'agir de la même région que le cluster de bases de données Aurora principal.
6. Remplissez les champs restants pour le cluster Aurora secondaire dans la nouvelle Région AWS. Il s'agit des mêmes options de configuration que pour n'importe quelle instance de cluster de bases de données Aurora.

   Pour une base de données Aurora globale basée sur Aurora MySQL–, ignorez l'option Enable read replica write forwarding (Activer le transfert d'écriture du réplica en lecture). Cette option n'a aucune fonction après la suppression de l'instance du lecteur.
7. Choisissez Ajouter une région AWS. Une fois la région ajoutée à votre base de données Aurora globale, vous pouvez la voir dans la liste des bases de données dans l'AWS Management Console comme l'illustre la capture d'écran.

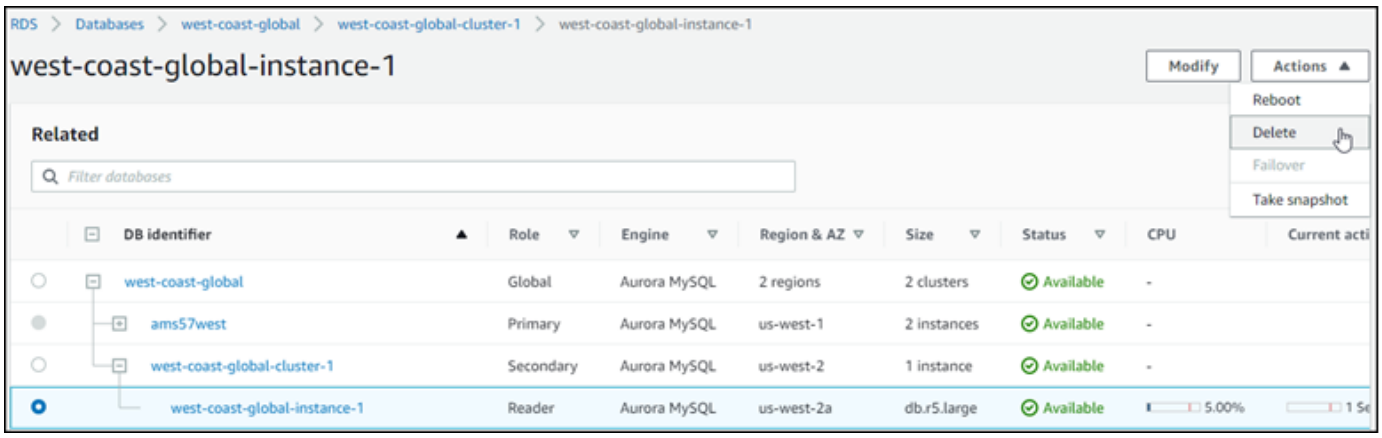


8. Avant de continuer, vérifiez l'état du cluster de bases de données Aurora secondaire et son instance de lecteur en utilisant l'AWS Management Console ou l'AWS CLI. Exemples :

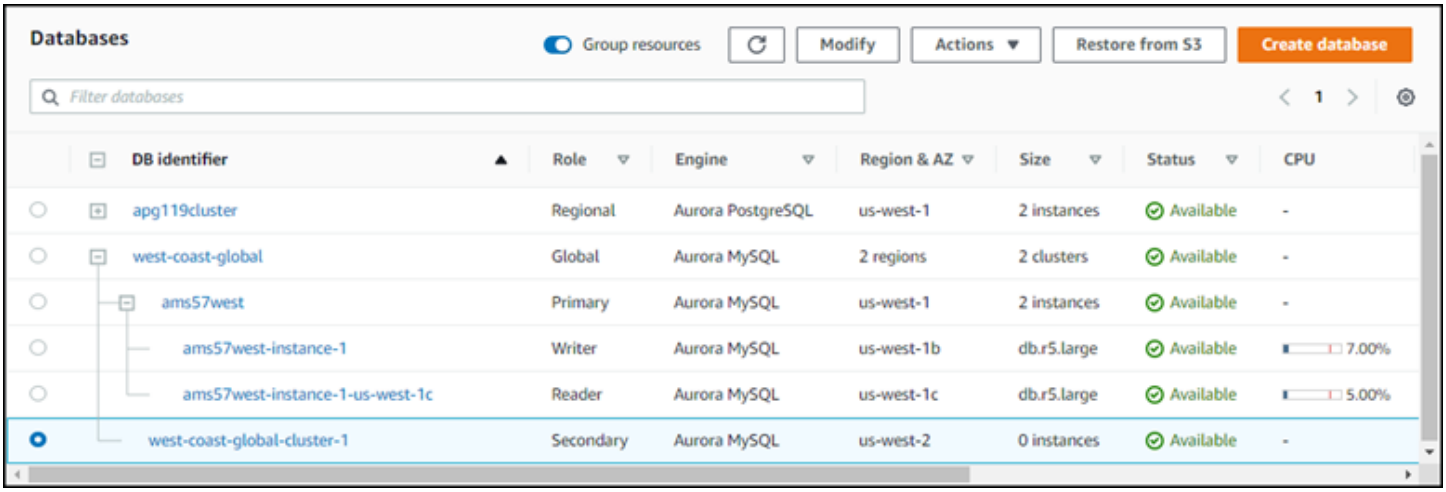
```
$ aws rds describe-db-clusters --db-cluster-identifier secondary-cluster-id --query
 '*[].[Status]' --output text
```

   Plusieurs minutes peuvent être nécessaires pour que l'état d'un cluster de bases de données Aurora secondaire nouvellement ajouté passe de `creating` à `available`. Lorsque le cluster de bases de données Aurora est disponible, vous pouvez supprimer l'instance de lecteur.

9. Choisissez l'instance de lecteur dans le cluster de bases de données Aurora secondaire, puis choisissez Delete (Supprimer).



Après la suppression de l'instance de lecteur, le cluster secondaire continue à faire partie de la base de données globale Aurora. Aucune instance ne lui est associée, comme indiqué ci-dessous.



Vous pouvez utiliser ce cluster de bases de données Aurora secondaire sans tête pour restaurer manuellement votre base de données Amazon Aurora globale en cas d'interruption non planifiée dans la Région AWS principale.

Création d'une base de données Amazon Aurora Global Database à partir d'un instantané Aurora ou Amazon RDS

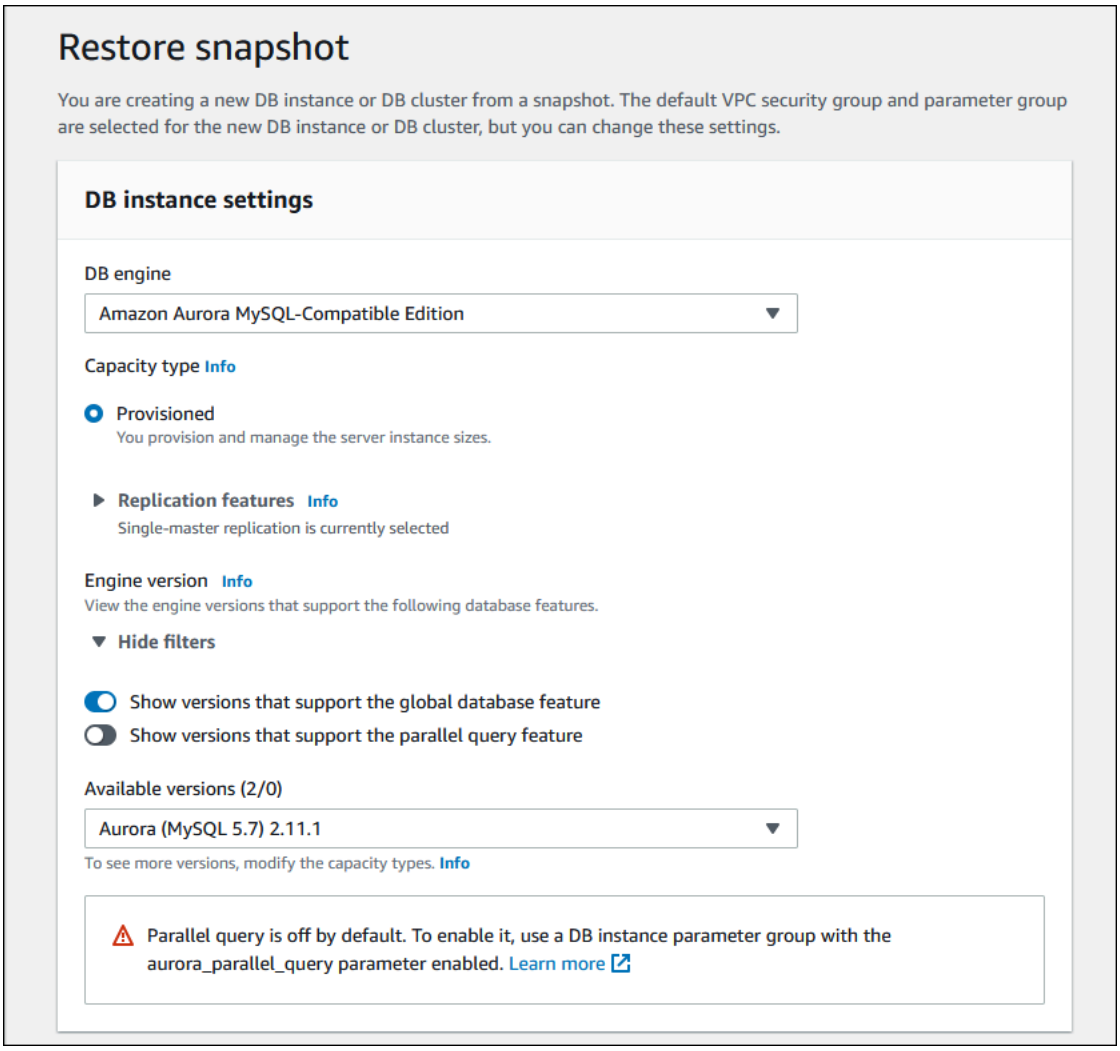
Vous pouvez restaurer un instantané de cluster de bases de données Aurora ou d'une instance de base de données Amazon RDS for l'utiliser comme point de départ pour votre base de données Aurora globale. Pour ce faire, restaurez l'instantané et créez un nouveau cluster de bases de données Aurora provisionné en même temps. Ajoutez ensuite une autre Région AWS au cluster de bases de données restauré, pour le transformer en une base de données Aurora globale. Tout cluster de bases de données Aurora créé de cette manière à l'aide d'un instantané devient le cluster principal de votre base de données Aurora globale.

Vous pouvez utiliser un instantané issu d'un cluster de bases de données `provisioned` ou `serverless` Aurora.

Au cours du processus de restauration, sélectionnez le même type de moteur de base de données que pour l'instantané. Par exemple, supposons que vous souhaitiez restaurer un instantané créé à partir d'un cluster de bases de données Aurora Serverless exécutant Aurora PostgreSQL. Dans ce cas, vous créez un cluster de bases de données Aurora PostgreSQL en utilisant le même moteur de bases de données Aurora et la même version.

Le cluster de bases de données restauré assume le rôle de cluster principal pour la base de données Aurora globale lorsque vous y ajoutez une Région AWS. Toutes les données contenues dans ce

cluster principal sont répliquées vers tous les clusters secondaires que vous ajoutez à votre base de données Aurora globale.



Gestion d’une base de données Amazon Aurora globale

Vous effectuez la plupart des opérations de gestion sur les clusters individuels qui constituent une base de données globale Aurora. Lorsque vous choisissez Group related resources (Ressources liées au groupe) sur la page Databases (Bases de données) de la console, vous pouvez voir que le cluster principal et le cluster secondaire sont regroupés sous la base de données globale associée. Pour rechercher les Régions AWS où les clusters de bases de données d’une base de données globale sont exécutées, le moteur de base de données et la version de Aurora et son identifiant, utilisez l’onglet Configuration (Configuration).

Les processus de basculement de base de données entre régions sont disponibles uniquement pour les bases de données globales Aurora, pas pour un cluster de bases de données Aurora

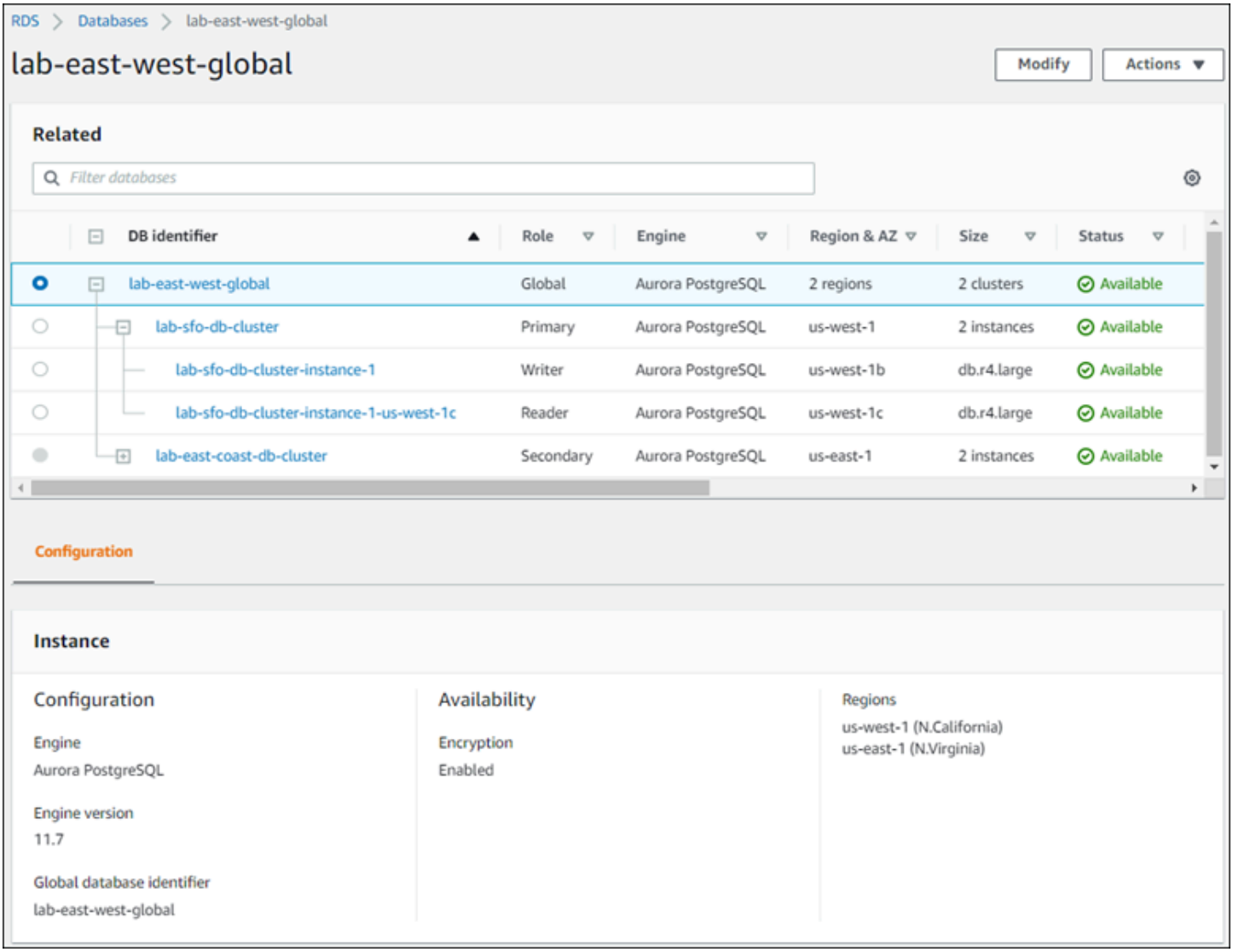
unique. Pour en savoir plus, consultez Utilisation de la bascule ou du basculement dans une base de données Amazon Aurora Global Database.

Pour restaurer une base de données Aurora globale suite à une panne non planifiée dans sa région principale, consultez Reprise d'une base de données Amazon Aurora globale à partir d'une panne non planifiée.

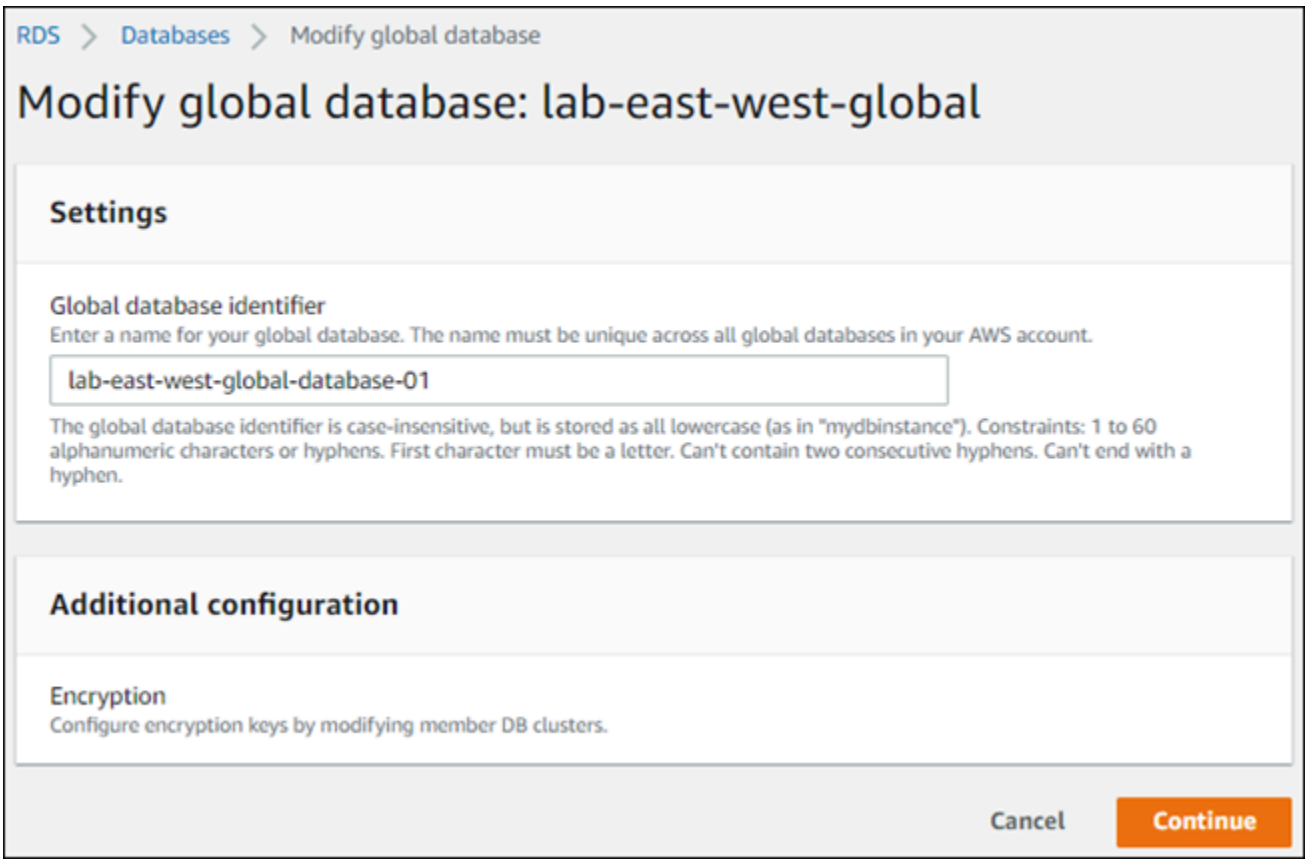
Rubriques

Suppression d'une base de données Amazon Aurora globale

La page Bases de données de l'AWS Management Console répertorie toutes vos bases de données Aurora globales et affiche le cluster principal et les clusters secondaires pour chacune d'elles. La base de données Aurora globale a ses propres paramètres de configuration. Plus précisément, elle comporte des Régions AWS associées à ses clusters principaux et secondaires, comme le montre la capture d'écran suivante.



Lorsque vous apportez des modifications à la base de données Aurora globale, vous avez la possibilité d'annuler ces modifications, comme indiqué dans la capture d'écran suivante.



Lorsque vous choisissez Continuer, vous confirmez les modifications.

Modification des paramètres d'une base de données Aurora globale

Vous pouvez configurer les groupes de paramètres Aurora indépendamment pour chaque cluster Aurora inclus dans la base de données Aurora globale. La plupart des paramètres fonctionnent de la même façon qu'avec les autres types de clusters Aurora. Nous vous recommandons de maintenir la cohérence des paramètres entre tous les clusters d'une base de données globale. Vous pourrez ainsi éviter les changements de comportement inattendus si vous choisissez un cluster secondaire en tant que cluster principal.

Par exemple, utilisez les mêmes paramètres pour les fuseaux horaires et les jeux de caractères afin d'éviter tout écart de comportement si un autre cluster devient le cluster principal.

Les paramètres de configuration `aurora_enable_repl_bin_log_filtering` et `aurora_enable_replica_log_compression` n'ont pas d'effet.

Dissociation d’un cluster d’une base de données Amazon Aurora globale

Vous pouvez supprimer des clusters Aurora de votre base de données Aurora globale pour plusieurs raisons différentes. Par exemple, vous pouvez supprimer un cluster Aurora d’une base de données Aurora globale si le cluster principal est dégradé ou isolé. Il devient alors un cluster de bases de données Aurora alloué autonome qui peut être utilisé pour créer une base de données Aurora globale. Pour en savoir plus, consultez Reprise d’une base de données Amazon Aurora globale à partir d’une panne non planifiée.

Vous pouvez également supprimer des clusters de bases de données Aurora, car vous souhaitez supprimer une base de données Aurora globale dont vous n’avez plus besoin. Vous ne pouvez pas supprimer la base de données Aurora globale tant que vous n’avez pas supprimé (détaché) tous les clusters de bases de données Aurora associés en laissant le principal en dernier. Pour plus d’informations, consultez Suppression d’une base de données Amazon Aurora globale.

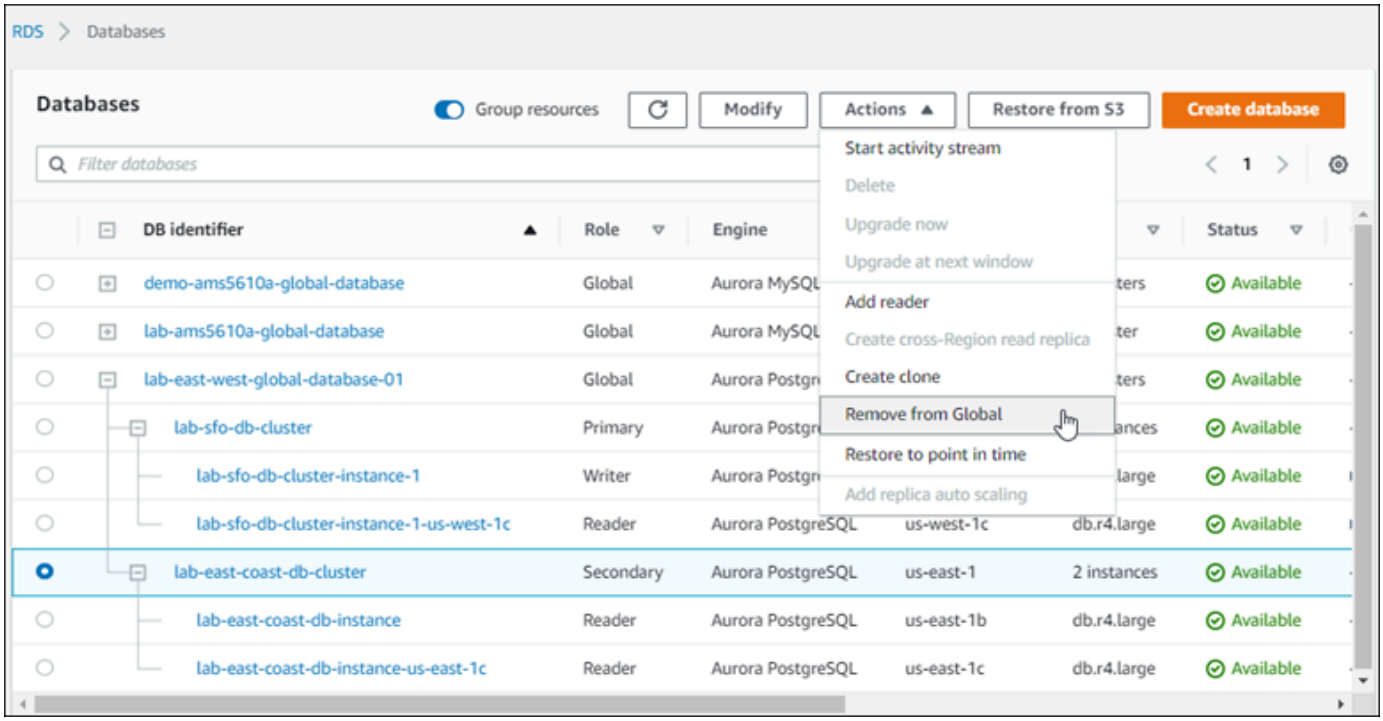
Lorsqu’un cluster de bases de données Aurora est détaché de la base de données Aurora globale, il n’est plus synchronisé avec le serveur principal. Il devient un cluster de bases de données Aurora alloué autonome avec des capacités complètes de lecture/écriture.

Vous pouvez supprimer des clusters de bases de données Aurora de votre base de données Aurora globale à l’aide de l’AWS Management Console, de l’AWS CLI ou de l’API RDS.

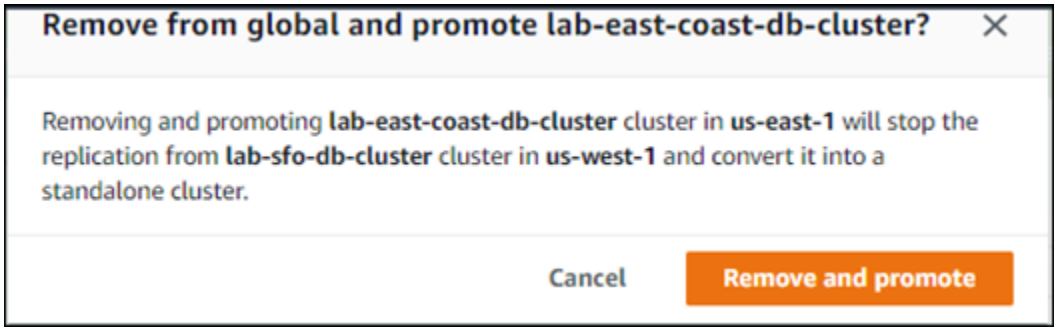
Console

Pour dissocier un cluster Aurora d’une base de données Aurora globale

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Choisissez le cluster sur la page Databases (Bases de données).
3. Pour Actions, choisissez Remove from Global (Dissocier de la base de données globale).

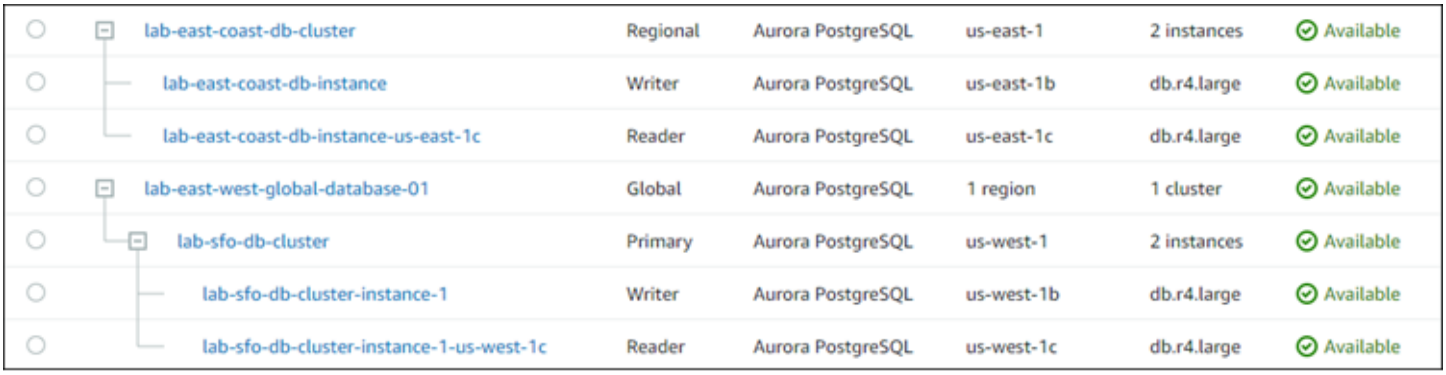


Une invite s'affiche, vous demandant de confirmer que vous souhaitez détacher le secondaire de la base de données Aurora globale.



4. Choisissez Supprimer et promouvoir pour supprimer le cluster de la base de données globale.

Le cluster Aurora ne sert plus de secondaire dans la base de données Aurora globale et n'est plus synchronisé avec le cluster de bases de données principal. Il s'agit d'un cluster de bases de données Aurora autonome avec une capacité complète de lecture/écriture.



Après avoir dissocié ou supprimé les clusters secondaires, vous pouvez procéder de la même façon pour dissocier le cluster principal. Vous ne pouvez pas détacher (supprimer) le cluster Aurora principal d'une base de données Aurora globale tant que vous n'avez pas supprimé tous les clusters secondaires.

La base de données Aurora globale peut rester dans la liste Databases (Bases de données), avec 0 région et zone de disponibilité. Vous pouvez la supprimer si vous ne souhaitez plus utiliser cette base de données Aurora globale. Pour plus d'informations, consultez Suppression d'une base de données Amazon Aurora globale.

AWS CLI

Pour dissocier un cluster Aurora d'une base de données Aurora globale, exécutez la commande remove-from-global-cluster de l'interface de ligne de commande avec les paramètres suivants :

- `--global-cluster-identifier` – Nom (identifiant) de votre base de données Aurora globale.
- `--db-cluster-identifier` – Nom de chaque cluster de bases de données Aurora à supprimer de la base de données Aurora globale. Supprimez tous les clusters de bases de données Aurora secondaires avant de supprimer le principal.

Les commandes suivantes dissocient un cluster secondaire, puis le cluster principal d'une base de données Aurora globale.

Pour Linux, macOS ou Unix :

```
aws rds --region secondary_region \
  remove-from-global-cluster \
    --db-cluster-identifier secondary_cluster_ARN \
    --global-cluster-identifier global_database_id
```

```
aws rds --region primary_region \
  remove-from-global-cluster \
    --db-cluster-identifier primary_cluster_ARN \
    --global-cluster-identifier global_database_id
```

Répétez la commande `remove-from-global-cluster --db-cluster-identifier` *`secondary_cluster_ARN`* pour chaque Région AWS secondaire de votre base de données Aurora globale.

Pour Windows :

```
aws rds --region secondary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier secondary_cluster_ARN ^
    --global-cluster-identifier global_database_id

aws rds --region primary_region ^
  remove-from-global-cluster ^
    --db-cluster-identifier primary_cluster_ARN ^
    --global-cluster-identifier global_database_id
```

Répétez la commande `remove-from-global-cluster --db-cluster-identifier` *`secondary_cluster_ARN`* pour chaque Région AWS secondaire de votre base de données Aurora globale.

API RDS

Pour dissocier un cluster Aurora d'une base de données globale Aurora avec l'API RDS, exécutez l'action RemoveFromGlobalCluster.

Suppression d'une base de données Amazon Aurora globale

Comme une base de données Aurora globale contient généralement des données critiques, vous ne pouvez pas supprimer cette base de données ainsi que les clusters associés en une seule étape. Pour supprimer une base de données Aurora globale, procédez comme suit :

- Supprimez tous les clusters de bases de données secondaires de la base de données Aurora globale. Chaque cluster devient un cluster de bases de données Aurora autonome. Pour savoir comment procéder, consultez Dissociation d'un cluster d'une base de données Amazon Aurora globale.

- À partir de chaque cluster de bases de données Aurora autonome, supprimez tous les réplicas Aurora.
- Supprimer le cluster secondaire de la base de données Aurora globale. Cela devient un cluster de bases de données Aurora autonome.
- À partir du cluster de bases de données Aurora principal, commencez par supprimer tous les réplicas Aurora, puis supprimez l'instance de scripteur de bases de données.

La suppression de l'instance de scripteur du cluster de bases de données Aurora nouvellement autonome supprime également généralement le cluster Aurora et la base de données Aurora globale.

Pour plus d'informations générales, consultez Suppression d'une instance de base de données d'un cluster de bases de données Aurora.

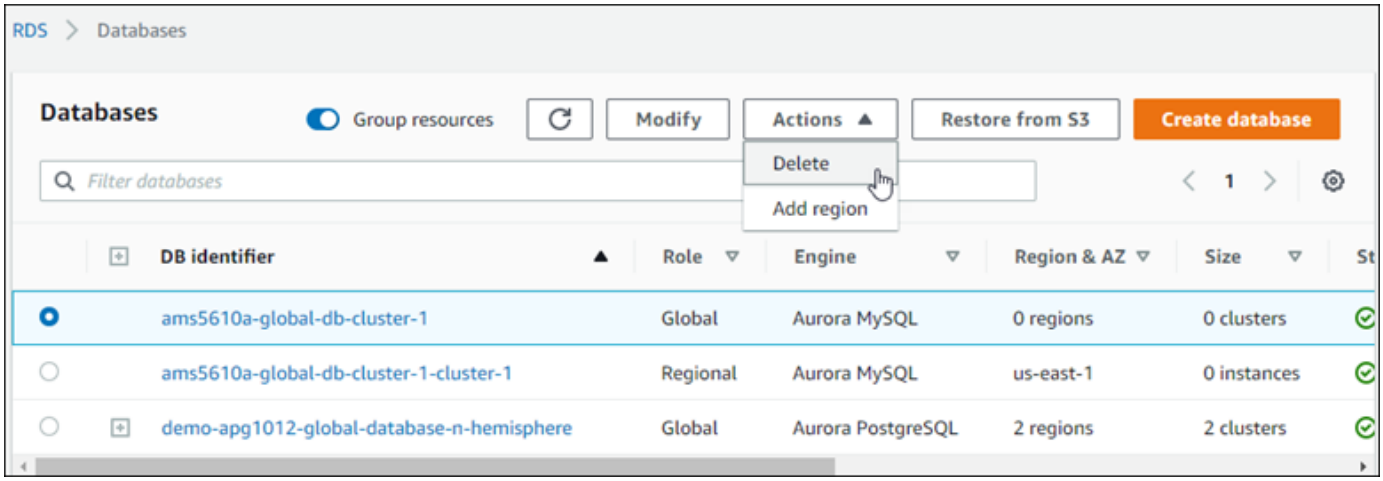
Pour supprimer une base de données Aurora globale, vous pouvez utiliser l'AWS Management Console, l'AWS CLI ou l'API RDS.

Console

Pour supprimer une base de données globale Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Sélectionnez Bases de données et recherchez la base de données Aurora globale que vous souhaitez supprimer dans la liste.
3. Confirmez que tous les autres clusters sont dissociés de la base de données Aurora globale. La base de données Aurora globale doit afficher 0 région et zone de disponibilité et une taille de 0 cluster.

   Si la base de données Aurora globale contient des clusters de bases de données Aurora, vous ne pouvez pas la supprimer. Si nécessaire, détachez les clusters de bases de données Aurora principaux et secondaires de la base de données Aurora globale. Pour plus d'informations, consultez Dissociation d'un cluster d'une base de données Amazon Aurora globale.
4. Choisissez votre base de données globale Aurora dans la liste, puis choisissez Supprimer dans le menu Actions.



AWS CLI

Pour supprimer une base de données Aurora globale, exécutez la commande de la CLI delete-global-cluster avec le nom de la Région AWS et l'identifiant de base de données Aurora global, comme illustré dans l'exemple suivant.

Pour Linux, macOS ou Unix :

```
aws rds --region primary_region delete-global-cluster \
   --global-cluster-identifier global_database_id
```

Pour Windows :

```
aws rds --region primary_region delete-global-cluster ^
   --global-cluster-identifier global_database_id
```

API RDS

Pour supprimer un cluster faisant partie d'une base de données Aurora globale, exécutez l'opération d'API DeleteGlobalCluster.

Balisage des ressources Amazon Aurora Global Database

Cette fonctionnalité vous permet d'appliquer des balises RDS aux ressources à différents niveaux au sein d'une base de données globale. Si vous ne savez pas comment les balises sont utilisées avec les ressources AWS ou Aurora, consultez Marquage des ressources Amazon Aurora et Amazon RDS avant d'appliquer des balises dans votre base de données globale.

> **Note**
>
> Comme AWS traite les données des balises dans le cadre de ses mécanismes de création de rapports sur les coûts, n'incluez pas de données sensibles ni de données d'identification personnelle (PII) dans les noms ou les valeurs des balises.

Vous pouvez appliquer des balises aux types de ressources suivants dans une base de données globale :

- L'objet conteneur pour l'ensemble de votre base de données globale. Cette ressource est connue sous le nom de cluster global.

  Après avoir créé le cluster global en effectuant une opération Ajouter une région AWS dans la console, vous pouvez ajouter des balises à l'aide de la page de détails du cluster global. Dans l'onglet Balises de la page de détails du cluster global, vous pouvez ajouter, supprimer ou modifier les balises et leurs valeurs associées en choisissant Gérer les balises.

  Avec l'AWS CLI et l'API RDS, vous pouvez ajouter des balises au cluster global lors de sa création. Vous pouvez également ajouter, supprimer ou modifier des balises pour un cluster global existant.
- Le cluster principal. Vous utilisez les mêmes procédures pour utiliser les balises ici que pour les clusters Aurora autonomes. Vous pouvez configurer les balises avant de convertir le cluster Aurora d'origine en base de données globale. Vous pouvez également ajouter, supprimer ou modifier des balises et leurs valeurs associées en choisissant Gérer les balises dans l'onglet Balises de la page de détails du cluster de bases de données.
- Tous les clusters secondaires. Vous utilisez les mêmes procédures pour utiliser les balises ici que pour les clusters Aurora autonomes. Vous pouvez configurer les balises lors de la création d'un cluster Aurora secondaire à l'aide de l'action Ajouter une région AWS dans la console. Vous pouvez également ajouter, supprimer ou modifier des balises et leurs valeurs associées en choisissant Gérer les balises dans l'onglet Balises de la page de détails du cluster de bases de données.
- Instances de base de données individuelles au sein des clusters principaux ou secondaires. Vous utilisez les mêmes procédures pour travailler avec les balises ici que pour les instances de base de données Aurora ou RDS. Vous pouvez configurer les balises en même temps que vous ajoutez une instance de base de données au cluster Aurora en utilisant l'action Ajouter un lecteur dans la console. Vous pouvez également ajouter, supprimer ou modifier des balises et leurs

valeurs associées en choisissant Gérer les balises dans l'onglet Balises de la page des détails de l'instance de base de données.

Voici quelques exemples des types de balises que vous pouvez attribuer dans une base de données globale :

- Vous pouvez ajouter des balises au cluster global pour enregistrer des informations générales sur votre application, telles que des identifiants anonymes représentant les responsables et les contacts au sein de votre entreprise. Vous pouvez utiliser des balises pour représenter les propriétés de l'application qui utilise la base de données globale.
- Vous pouvez ajouter des balises au cluster principal et aux clusters secondaires pour suivre les coûts de votre application au niveau de la région AWS. Pour plus d'informations sur cette procédure, consultez Comment fonctionne AWS la facturation avec les tags dans Amazon RDS.
- Vous pouvez ajouter des balises à des instances de base de données spécifiques à l'aide des clusters Aurora pour indiquer leur but particulier. Par exemple, au sein du cluster principal, vous pouvez avoir une instance de lecteur avec une faible priorité de basculement qui est utilisée exclusivement pour la génération de rapports. Une balise permet de distinguer cette instance de base de données spécialisée des autres instances dédiées à la haute disponibilité au sein du cluster principal.
- Vous pouvez utiliser des balises à tous les niveaux des ressources de votre base de données globale afin de contrôler l'accès par le biais de politiques IAM. Pour plus d'informations, consultez Contrôle de l'accès aux ressources AWS dans le Guide de l'utilisateur AWS Identity and Access Management.

  > **Tip**
  >
  > Dans la AWS Management Console, vous ajoutez des balises au conteneur de cluster global en tant qu'étape distincte après l'avoir créé. Si vous souhaitez éviter que le cluster global soit sans balises de contrôle d'accès à sa création, vous pouvez appliquer les balises pendant l'opération `CreateGlobalCluster` en créant cette ressource via l'AWS CLI, l'API RDS ou un modèle CloudFormation.

- Vous pouvez utiliser des balises au niveau du cluster, ou pour le cluster global, afin d'enregistrer des informations relatives à l'assurance qualité et aux tests de votre application. Par exemple, vous pouvez spécifier une balise sur un cluster de bases de données pour enregistrer la dernière fois que vous avez effectué une opération de bascule vers ce cluster. Vous pouvez spécifier une balise

sur le cluster global pour enregistrer l'heure du dernier événement de reprise après sinistre pour l'ensemble de l'application.

Exemples de balisage des bases de données globales avec l'AWS CLI

Les exemples suivants de l'AWS CLI montrent comment vous pouvez spécifier et examiner les balises pour tous les types de ressources Aurora dans votre base de données globale.

Vous pouvez spécifier des balises pour le conteneur de cluster global à l'aide de la commande `create-global-cluster`. L'exemple suivant crée un cluster global et lui attribue deux balises. Ces balises ont les clés `tag1` et `tag2`.

```
$  aws rds create-global-cluster --global-cluster-identifier my_global_cluster_id \
  --engine aurora-mysql --tags Key=tag1,Value=val1 Key=tag2,Value=val2
```

Vous pouvez répertorier les balises du conteneur de cluster global à l'aide de la commande `describe-global-clusters`. Lorsque vous utilisez des balises, vous exécutez généralement d'abord la commande permettant de récupérer l'Amazon Resource Name (ARN) du cluster global. Vous utiliserez cet ARN comme paramètre dans les commandes suivantes pour travailler avec les balises. La commande suivante affiche les informations de balise dans l'attribut `TagList`. Elle montre également l'ARN, qui sera utilisé comme paramètre dans les exemples suivants.

```
$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            "GlobalClusterArn": "my_global_cluster_arn",
            ...
            "TagList": [
                {
                    "Value": "val1",
                    "Key": "tag1"
                },
                {
                    "Value": "val2",
                    "Key": "tag2"
                }
            ]
```

```
        }
    ]
}
```

Vous pouvez ajouter de nouvelles balises à l'aide de la commande `add-tags-to-resource`. Avec cette commande, vous spécifiez l'Amazon Resource Name (ARN) du cluster global plutôt que son identifiant. L'ajout d'une balise ayant le même nom qu'une balise existante remplace la valeur de cette balise. Si vous incluez des espaces ou des caractères spéciaux dans les valeurs des balises, citez ces valeurs de manière appropriée en fonction de votre système d'exploitation ou de votre shell de commande. L'exemple suivant modifie les balises du cluster global de l'exemple précédent. À l'origine, les balises du cluster avaient les clés `tag1` et `tag2`. Une fois la commande terminée, le cluster global a une nouvelle balise avec clé `tag3`, et la balise avec la clé `tag1` a une valeur différente.

```
$  aws rds add-tags-to-resource --resource-name my_global_cluster_arn \
  --tags Key=tag1,Value="new value for tag1" Key=tag3,Value="entirely new tag"

$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            ...
            "TagList": [
                {
                    "Value": "new value for tag1",
                    "Key": "tag1"
                },
                {
                    "Value": "val2",
                    "Key": "tag2"
                },
                {
                    "Value": "entirely new tag",
                    "Key": "tag3"
                }
            ]
        }
    ]
}
```

Vous pouvez supprimer une balise du cluster global à l'aide de la commande `remove-tags-from-resource`. Avec cette commande, vous spécifiez uniquement un ensemble de clés de balise, sans aucune valeur de balise. L'exemple suivant modifie les balises du cluster global de l'exemple précédent. À l'origine, les balises du cluster avaient les clés `tag1`, `tag2` et `tag3`. Une fois la commande terminée, il ne reste que la balise avec la clé `tag1`.

```
$  aws rds remove-tags-from-resource --resource-name my_global_cluster_arn --tag-keys
 tag2 tag3

$  aws rds describe-global-clusters --global-cluster-identifier my_global_cluster_id
{
    "GlobalClusters": [
        {
            "Status": "available",
            "Engine": "aurora-mysql",
            ...
            "TagList": [
                {
                    "Value": "new value for tag1",
                    "Key": "tag1"
                }
            ]
        }
    ]
}
```

Connexion à Amazon Aurora Global Database

Chaque base de données Aurora Global Database est fournie avec un point de terminaison d'enregistreur qui est automatiquement mis à jour par Aurora pour acheminer les demandes vers l'instance d'enregistreur actuelle du cluster de bases de données principal. Avec le point de terminaison d'enregistreur, vous n'avez pas à modifier la chaîne de connexion après avoir modifié l'emplacement de la région principale à l'aide des fonctionnalités gérées de bascule et de basculement Aurora Global Database. Pour en savoir plus sur l'utilisation du point de terminaison d'enregistreur avec la bascule et le basculement Aurora Global Database, consultez Utilisation de la bascule ou du basculement dans une base de données Amazon Aurora Global Database. Pour plus d'informations sur la connexion à une base de données Aurora Global Database avec un proxy RDS, consultez Utilisation du proxy RDS avec les bases de données globales Aurora.

Rubriques

Sélection du point de terminaison qui répond aux besoins de votre application

La connexion à une base de données Aurora Global Database dépend de ce que vous devez y faire (lecture ou écriture à partir de la base de données) et de la région AWS vers laquelle vous souhaitez acheminer vos demandes. Voici quelques cas d'utilisation typiques :

- Acheminement des demandes vers l'instance d'enregistreur : connectez-vous au point de terminaison d'enregistreur Aurora Global Database si vous devez exécuter des instructions de langage de manipulation de données (DML) et de langage de définition de données (DDL), ou si vous avez besoin d'une cohérence renforcée entre les lectures et les écritures. Ce point de terminaison achemine les demandes vers l'instance d'enregistreur du cluster principal de votre base de données globale. Ce point de terminaison est automatiquement mis à jour pour acheminer les demandes vers l'instance d'enregistreur, ce qui évite d'avoir à mettre à jour votre application chaque fois que vous modifiez l'emplacement de l'enregistreur dans votre cluster global. Vous pouvez également utiliser le point de terminaison global pour envoyer des demandes de lecture/écriture entre régions à l'enregistreur.

> **Note**
>
> Si vous avez configuré votre base de données globale avant que le point de terminaison d'enregistreur Aurora Global Database ne soit disponible, il est possible que votre application se connecte au point de terminaison du cluster principal. Dans ce cas, nous vous recommandons de modifier les paramètres de connexion pour que le point de terminaison d'enregistreur global soit utilisé à la place. Cela évite d'avoir à modifier les paramètres de connexion après chaque bascule ou basculement Aurora Global Database. La première partie du nom du point de terminaison d'enregistreur est le nom de votre base de données Aurora Global Database. Dès lors, si vous renommez votre base de données Aurora Global Database, le nom du point de terminaison d'enregistreur changera, et tout code qui l'utilise devra être mis à jour avec le nouveau nom.

- Mise à l'échelle des lectures plus proche de la région de votre application : pour mettre à l'échelle les demandes en lecture seule dans la même région AWS que votre application ou dans une région à proximité, connectez-vous au point de terminaison de lecteur des clusters Aurora principaux ou secondaires.
- Mise à l'échelle des lectures avec des écritures entre régions occasionnelles : pour les instructions DML occasionnelles, telles que pour la maintenance et le nettoyage des données, connectez-vous au point de terminaison du lecteur d'un cluster secondaire sur lequel le transfert d'écriture est activé. Avec le transfert d'écriture, Aurora transmet automatiquement les instructions d'écriture à l'enregistreur de la région principale de votre base de données Aurora Global Database. Le transfert d'écriture offre les avantages suivants :
  - Vous n'avez pas besoin de faire le gros du travail pour établir la connectivité entre les clusters secondaire et principal afin d'envoyer des écritures entre régions.
  - Il n'est pas nécessaire de séparer les demandes de lecture et les demandes d'écriture dans l'application.
  - Il n'est pas nécessaire de développer une logique complexe pour gérer la cohérence des demandes de lecture après écriture.

  Toutefois, avec le transfert d'écriture, vous devrez mettre à jour le code ou la configuration de votre application pour pouvoir vous connecter au point de terminaison du lecteur de la région principale qui vient d'être promue après avoir effectué une bascule ou un basculement entre régions. Nous vous recommandons de surveiller la latence des opérations effectuées par le biais du transfert d'écriture, afin de contrôler la surcharge liée au traitement des demandes d'écriture. Enfin, le transfert d'écriture ne prend pas en charge certaines opérations MySQL ou PostgreSQL, telles que les modifications du langage de définition des données (DDL) ou les instructions `SELECT FOR UPDATE`.

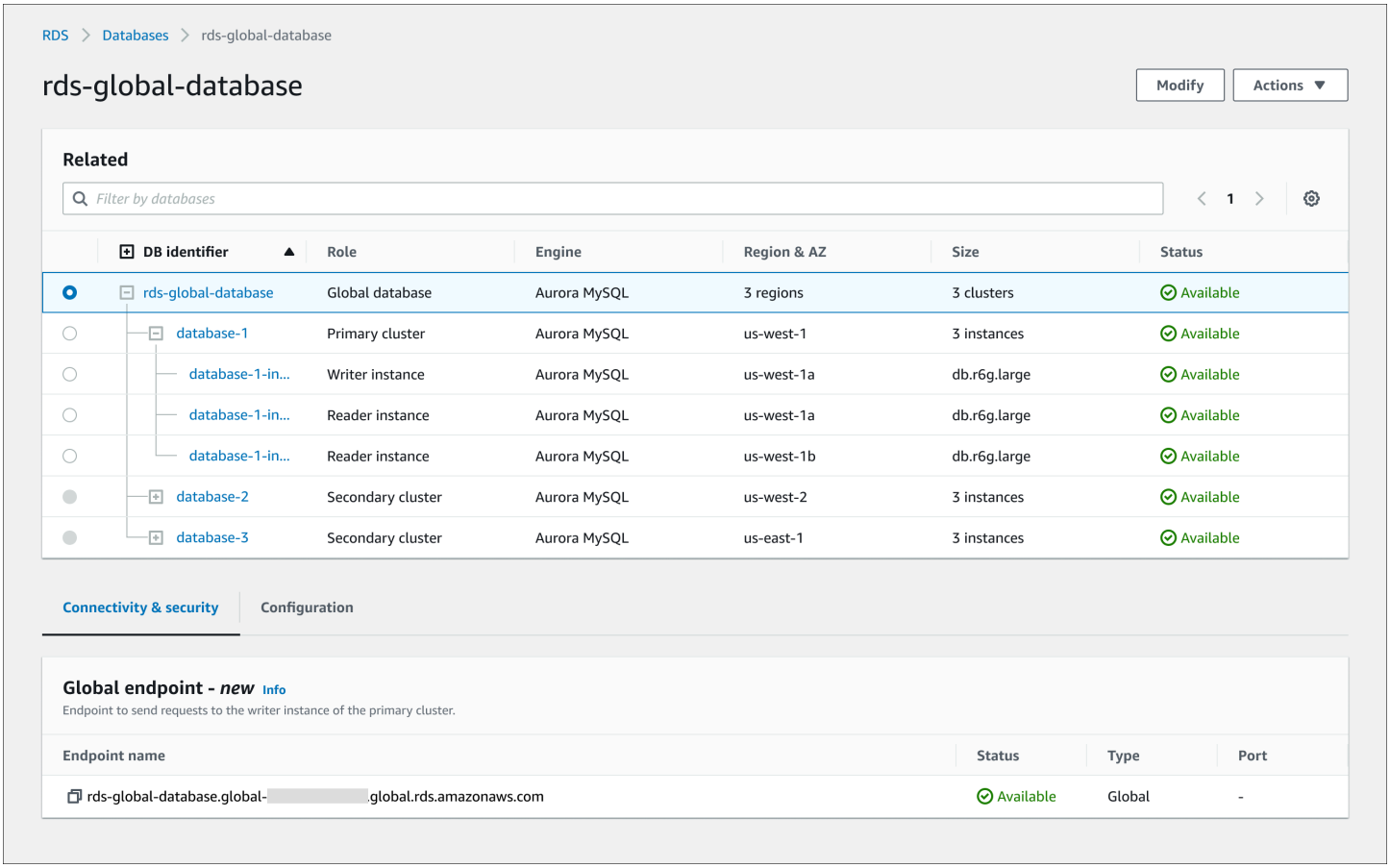
  Pour en savoir plus sur l'utilisation du transfert d'écriture entre régions AWS, consultez Utilisation du transfert d'écriture dans une base de données globale Amazon Aurora.

Pour plus de détails sur les différents types de points de terminaison Aurora, consultez Connexion à un cluster de bases de données Amazon Aurora.

Affichage des points de terminaison d'une base de données Amazon Aurora Global Database

Lorsque vous examinez une base de données Aurora Global Database dans la console, vous pouvez voir tous les points de terminaison associés à tous ses clusters. La figure suivante montre un exemple des types de points de terminaison que vous voyez lorsque vous consultez les détails de votre cluster de bases de données principal :

- Enregistreur global : point de terminaison unique de lecture/écriture qui pointe toujours vers l'instance de base de données d'enregistreur actuelle pour le cluster de bases de données global.
- Enregistreur : point de terminaison de connexion pour les demandes de lecture/écriture adressées au cluster de bases de données principal du cluster de bases de données global.
- Lecteur : point de terminaison de connexion pour les demandes de lecture seule adressées à un cluster de bases de données principal ou secondaire du cluster de bases de données global. Pour limiter la latence, spécifiez le point de terminaison du lecteur de votre Région AWS ou de l'Région AWS la plus proche.



Console

Pour afficher les points de terminaison d'une base de données globale

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Dans la liste, sélectionnez la base de données globale ou le cluster de bases de données principal ou secondaire dont vous souhaitez afficher les points de terminaison.
4. Choisissez l'onglet Connectivité et sécurité pour voir les détails du point de terminaison. Les points de terminaison affichés dépendent du type de cluster que vous avez sélectionné, comme suit :

   - Base de données globale : point de terminaison d'enregistreur global.
   - Cluster de bases de données principal : point de terminaison d'enregistreur global, point de terminaison du cluster et point de terminaison de lecteur pour le cluster principal.
   - Cluster de bases de données secondaire : point de terminaison de cluster et point de terminaison de lecteur pour le cluster secondaire. Sur un cluster secondaire, le point de terminaison du cluster affiche le statut inactif, car il ne gère pas les demandes d'écriture. Vous pouvez tout de même vous connecter au point de terminaison du cluster, mais uniquement pour les requêtes de lecture.

AWS CLI

Pour afficher le point de terminaison d'enregistreur du cluster global, utilisez la commande AWS CLI describe-global-clusters, comme dans l'exemple suivant.

```
aws rds describe-global-clusters --region aws_region
{
    "GlobalClusters": [
        {
            "GlobalClusterIdentifier": "global_cluster_id",
            "GlobalClusterResourceId": "cluster-unique_string",
            "GlobalClusterArn": "arn:aws:rds::123456789012:global-
cluster:global_cluster_id",
            "Status": "available",
            "Engine": "aurora-mysql",
            "EngineVersion": "5.7.mysql_aurora.2.11.2",
            "GlobalClusterMembers": [
```

```
            ...
        ],
        "Endpoint":
 "global_cluster_id.global-unique_string.global.rds.amazonaws.com"
    }
  ]
}
```

Pour afficher les points de terminaison du cluster et du lecteur pour les clusters de bases de données membres du cluster global, utilisez la commande AWS CLI describe-db-clusters, comme dans l'exemple suivant. Les valeurs renvoyées pour `Endpoint` et `ReaderEndpoint` sont les points de terminaison du cluster et du lecteur, respectivement.

```
aws rds describe-db-clusters --region primary_region --db-cluster-
identifier db_cluster_id
{
    "DBClusters": [
        {
            "AllocatedStorage": 1,
            "AvailabilityZones": [
                "az_1",
                "az_2",
                "az_3"
            ],
            "BackupRetentionPeriod": 1,
            "DBClusterIdentifier": "db_cluster_id",
            "DBClusterParameterGroup": "default.aurora-mysql5.7",
            "DBSubnetGroup": "default",
            "Status": "available",
            "EarliestRestorableTime": "2023-08-01T18:21:11.301Z",
            "Endpoint":
 "db_cluster_id.cluster-unique_string.primary_region.rds.amazonaws.com",
            "ReaderEndpoint": "db_cluster_id.cluster-
ro-unique_string.primary_region.rds.amazonaws.com",
            "MultiAZ": false,
            "Engine": "aurora-mysql",
            "EngineVersion": "5.7.mysql_aurora.2.11.2",

            "ReadReplicaIdentifiers": [
                "arn:aws:rds:secondary_region:123456789012:cluster:db_cluster_id"
            ],
            "DBClusterMembers": [
```

```
                {
                    "DBInstanceIdentifier": "db_instance_id",
                    "IsClusterWriter": true,
                    "DBClusterParameterGroupStatus": "in-sync",
                    "PromotionTier": 1
                }
            ],

            ...
            "TagList": [],
            "GlobalWriteForwardingRequested": false
        }
    ]
}
```

API RDS

Pour afficher le point de terminaison d'enregistreur du cluster global, utilisez l'opération DescribeGlobalClusters de l'API RDS. Pour afficher les points de terminaison du cluster et du lecteur pour les clusters de bases de données membres du cluster global, utilisez l'opération DescribeDBClusters de l'API RDS.

Considérations relatives à l'utilisation des points de terminaison d'enregistreur global

Pour utiliser efficacement les points de terminaison d'enregistreur Aurora Global Database, reportez-vous aux consignes et aux bonnes pratiques suivantes :

- Pour minimiser les perturbations après une bascule ou un basculement entre régions, vous pouvez configurer la connectivité du VPC entre le calcul de votre application et vos régions AWS principale et secondaire. Supposons, par exemple, que des applications ou des systèmes clients s'exécutent dans le même VPC que le cluster principal. Si le cluster secondaire est promu, le point de terminaison d'enregistreur global change automatiquement pour pointer vers ce cluster. Bien que le point de terminaison d'enregistreur global vous permette d'éviter de devoir modifier les paramètres de connexion de votre application, celle-ci ne peut pas accéder aux adresses IP du VPC de la région AWS principale qui vient d'être promue tant que vous n'avez pas configuré le réseau entre les deux VPC. Consultez Options de connectivité entre VPC Amazon pour évaluer les différentes options de configuration de cette connectivité.
- La mise à jour du point de terminaison d'enregistreur global après une bascule ou un basculement global de la base de données peut prendre du temps en fonction de la durée de mise en cache

de votre service de noms de domaine (DNS). Consultez le Manuel de l'administrateur de base de données Amazon Aurora MySQL pour en savoir plus. La base de données Aurora Global Database émet un événement RDS lorsqu'elle constate la modification du DNS sur le point de terminaison d'enregistreur global. Vous pouvez utiliser cet événement pour concevoir des stratégies visant à garantir que le cache DNS ne s'étende pas au-delà de la période qui suit la génération de l'événement. Pour plus d'informations, consultez Événements de cluster de bases de données.

- La base de données Aurora Global Database réplique les données de manière asynchrone. Les méthodes de basculement entre régions peuvent entraîner la réplication de certaines données de transaction d'écriture qui n'ont pas été répliquées dans la région secondaire choisie avant que le basculement se produise. Bien qu'Aurora fasse de son mieux pour bloquer les écritures dans la région AWS principale d'origine, le basculement peut être source de problèmes d'incohérence des données, aussi appelée split-brain. Les considérations visant à minimiser les pertes de données et le risque d'incohérence des données (ou split-brain) s'appliquent également aux points de terminaison d'enregistreur Aurora Global Database. Pour plus d'informations, consultez Réalisation de basculements gérés pour les bases de données globales Aurora.

Utilisation du transfert d'écriture dans une base de données globale Amazon Aurora

Vous pouvez réduire le nombre de points de terminaison que vous devez gérer pour les applications exécutées sur votre base de données globale Aurora, en utilisant le transfert d'écriture. Avec le transfert d'écriture activé, les clusters secondaires d'une base de données globale Aurora transfèrent des instructions SQL qui effectuent des opérations d'écriture vers le cluster principal. Le cluster principal met à jour la source, puis propage les modifications résultantes à toutes les régions AWS secondaires.

Grâce à cette configuration, vous n'avez pas à implémenter votre propre mécanisme pour envoyer des opérations d'écriture d'une région AWS secondaire à la région principale. Aurora gère la configuration réseau entre régions. Aurora transmet également tout le contexte de session et transactionnel nécessaire pour chaque instruction. Les données sont toujours modifiées d'abord sur le cluster principal, puis répliquées sur les clusters secondaires de la base de données globale Aurora. De cette façon, le cluster principal est toujours la source fiable avec la copie la plus récente de toutes vos données.

Rubriques

- Utilisation du transfert d'écriture dans une base de données globale Aurora MySQL

- Utilisation du transfert d'écriture dans une base de données globale Aurora PostgreSQL

Utilisation du transfert d'écriture dans une base de données globale Aurora MySQL

Rubriques

- Régions et versions disponibles pour le transfert d'écriture dans Aurora MySQL
- Activation du transfert d'écriture dans Aurora MySQL
- Vérification de l'activation du transfert d'écriture dans un cluster secondaire dans Aurora MySQL
- Compatibilité des applications et de SQL avec le transfert d'écriture dans Aurora MySQL
- Isolement et cohérence pour le transfert d'écriture dans Aurora MySQL
- Exécution d'instructions en plusieurs parties avec le transfert d'écriture dans Aurora MySQL
- Transactions avec transfert d'écriture dans Aurora MySQL
- Paramètres de configuration pour le transfert d'écriture dans Aurora MySQL
- Métriques Amazon CloudWatch pour le transfert d'écriture dans Aurora MySQL
- Variables d'état Aurora MySQL pour le transfert d'écriture

Régions et versions disponibles pour le transfert d'écriture dans Aurora MySQL

Le transfert d'écriture est pris en charge par Aurora MySQL 2.08.1 et versions ultérieures, dans toutes les régions où les bases de données globales Aurora MySQL sont présentes.

Pour en savoir plus sur les régions et les versions disponibles des bases de données globales Aurora MySQL, consultez Bases de données Aurora globales avec Aurora MySQL.

Activation du transfert d'écriture dans Aurora MySQL

Par défaut, le transfert d'écriture n'est pas activé lorsque vous ajoutez un cluster secondaire à une base de données globale Aurora.

Pour activer le transfert d'écriture à l'aide de la AWS Management Console, cochez la case Activer le transfert d'écriture global sous Transfert d'écriture de réplica en lecture lorsque vous ajoutez une région pour une base de données globale. Pour un cluster secondaire existant, modifiez le cluster pour Activer le transfert d'écriture global. Pour désactiver le transfert d'écriture, décochez la

case Activer le transfert d'écriture global lors de l'ajout de la région ou de la modification du cluster secondaire.

Pour activer le transfert d'écriture à l'aide de l'AWS CLI, utilisez l'option `--enable-global-write-forwarding`. Cette option est utile lorsque vous créez un nouveau cluster secondaire à l'aide de la commande `create-db-cluster`. Elle est également utile lorsque vous modifiez un cluster secondaire existant à l'aide de la commande `modify-db-cluster`. Elle nécessite que la base de données globale utilise une version d'Aurora qui prend en charge le transfert d'écriture. Vous pouvez désactiver le transfert d'écriture en utilisant l'option `--no-enable-global-write-forwarding` avec ces mêmes commandes de CLI.

Pour activer le transfert d'écriture à l'aide de l'API Amazon RDS, définissez le paramètre `EnableGlobalWriteForwarding` sur `true`. Ce paramètre agit lorsque vous créez un cluster secondaire à l'aide de l'opération `CreateDBCluster`. Il agit également lorsque vous modifiez un cluster secondaire existant à l'aide de l'opération `ModifyDBCluster`. Il nécessite que la base de données globale utilise une version d'Aurora qui prend en charge le transfert d'écriture. Vous pouvez désactiver le transfert d'écriture en définissant le paramètre `EnableGlobalWriteForwarding` sur `false`.

> **Note**
>
> Pour qu'une session de base de données utilise le transfert d'écriture, spécifiez un paramètre pour le paramètre de configuration `aurora_replica_read_consistency`. Faites-le lors de chaque session qui utilise la fonctionnalité de transfert d'écriture. Pour plus d'informations sur ce paramètre, consultez Isolement et cohérence pour le transfert d'écriture dans Aurora MySQL.
>
> La fonctionnalité de proxy RDS ne prend pas en charge la valeur `SESSION` de la variable `aurora_replica_read_consistency`. La définition de cette valeur peut entraîner un comportement inattendu.

Les exemples de CLI suivants montrent comment configurer une base de données Aurora globale avec le transfert d'écriture activé ou désactivé. Les éléments en surbrillance représentent les commandes et les options qui sont importantes pour spécifier et conserver la cohérence lors de la configuration de l'infrastructure d'une base de données Aurora globale.

L'exemple suivant crée une base de données Aurora globale, un cluster principal et un cluster secondaire avec le transfert d'écriture activé. Remplacez vos propres choix par le nom d'utilisateur, le mot de passe et les régions AWS principales et secondaires.

```
# Create overall global database.
aws rds create-global-cluster --global-cluster-identifier write-forwarding-test \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

# Create primary cluster, in the same AWS Region as the global database.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-1 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --master-username user_name --master-user-password password \
  --region us-east-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-1 \
  --db-instance-identifier write-forwarding-test-cluster-1-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-1 \
  --db-instance-identifier write-forwarding-test-cluster-1-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-1

# Create secondary cluster, in a different AWS Region than the global database,
# with write forwarding enabled.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-2 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2 \
  --enable-global-write-forwarding

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
```

```
  --db-instance-identifier write-forwarding-test-cluster-2-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-east-2
```

L'exemple suivant est la suite de l'exemple précédent. Il crée un cluster secondaire sans le transfert d'écriture activé, puis active le transfert d'écriture. A la fin de cet exemple, le transfert d'écriture est activé sur tous les clusters secondaires de la base de données globale.

```
# Create secondary cluster, in a different AWS Region than the global database,
# without write forwarding enabled.
aws rds create-db-cluster --global-cluster-identifier write-forwarding-test \
  --db-cluster-identifier write-forwarding-test-cluster-2 \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-1 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds create-db-instance --db-cluster-identifier write-forwarding-test-cluster-2 \
  --db-instance-identifier write-forwarding-test-cluster-2-instance-2 \
  --db-instance-class db.r5.large \
  --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.1 \
  --region us-west-1

aws rds modify-db-cluster --db-cluster-identifier write-forwarding-test-cluster-2 \
  --region us-east-2 \
  --enable-global-write-forwarding
```

Vérification de l'activation du transfert d'écriture dans un cluster secondaire dans Aurora MySQL

Pour déterminer si vous pouvez utiliser le transfert d'écriture à partir d'un cluster secondaire, vous pouvez vérifier si le cluster possède l'attribut `"GlobalWriteForwardingStatus": "enabled"`.

Dans la AWS Management Console, dans l'onglet Configuration de la page de détails du cluster, vous pouvez voir l'état Activé pour Transfert global d'écriture de réplica en lecture.

Pour voir l'état du paramètre global de transfert d'écriture pour tous vos clusters, exécutez la commande d'AWS CLI suivante.

Un cluster secondaire affiche la valeur `"enabled"` ou `"disabled"` pour indiquer si le transfert d'écriture est activé ou désactivé. La valeur `null` indique que le transfert d'écriture n'est pas disponible pour ce cluster. Soit le cluster ne fait pas partie d'une base de données globale, soit il s'agit du cluster principal et non d'un cluster secondaire. La valeur peut également être `"enabling"` ou `"disabling"` si le transfert d'écriture est en cours d'activation ou de désactivation.

Example

```
aws rds describe-db-clusters \
--query '*[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu

[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    },
    {
        "GlobalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-2"
    },
    {
        "GlobalWriteForwardingStatus": null,
        "DBClusterIdentifier": "non-global-cluster"
    }
]
```

Pour rechercher uniquement les clusters secondaires pour lesquels le transfert d'écriture global est activé, exécutez la commande suivante. Cette commande renvoie également le point de terminaison du lecteur du cluster. Vous utilisez le point de terminaison du lecteur du cluster secondaire lorsque vous utilisez le transfert d'écriture du secondaire vers le principal dans votre base de données Aurora globale.

Example

```
aws rds describe-db-clusters --query 'DBClusters[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
 | [?GlobalWriteForwardingStatus == `enabled`]'
```

```
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "ReaderEndpoint": "aurora-write-forwarding-test-replica-1.cluster-ro-
cnpexample.us-west-2.rds.amazonaws.com",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    }
]
```

Compatibilité des applications et de SQL avec le transfert d'écriture dans Aurora MySQL

Vous pouvez utiliser les types d'instructions SQL suivants avec le transfert d'écriture :

- Instructions DML (Data Manipulation Language) comme `INSERT`, `DELETE` et `UPDATE`. Il existe certaines restrictions concernant les propriétés de ces instructions que vous pouvez utiliser avec le transfert d'écriture, comme décrit ci-dessous.
- Instructions `SELECT ... LOCK IN SHARE MODE` et `SELECT FOR UPDATE`.
- Instructions `PREPARE` et `EXECUTE`.

Utilisées dans une base de données globale avec transfert d'écriture, certaines instructions ne sont pas autorisées ou peuvent produire des résultats obsolètes. Par conséquent, le paramètre `EnableGlobalWriteForwarding` est désactivé par défaut pour les clusters secondaires. Avant de l'activer, vérifiez que votre code d'application n'est affecté par aucune de ces restrictions.

Les restrictions suivantes s'appliquent aux instructions SQL que vous utilisez avec le transfert d'écriture. Dans certains cas, vous pouvez utiliser les instructions sur des clusters secondaires avec le transfert d'écriture activé au niveau du cluster. Cette approche fonctionne si le transfert d'écriture n'est pas activé dans la session par le paramètre de configuration `aurora_replica_read_consistency`. Essayer d'utiliser une instruction lorsqu'elle n'est pas autorisée en raison du transfert d'écriture provoque un message d'erreur au format suivant.

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'operation with write
 forwarding'.
```

Langage de définition de données (DDL)

Connectez-vous au cluster principal pour exécuter des instructions DDL. Vous ne pouvez pas les exécuter à partir des instances de base de données de lecteur.

Mise à jour d’une table permanente à l’aide des données d’une table temporaire

Vous pouvez utiliser des tables temporaires sur des clusters secondaires avec le transfert d’écriture activé. Toutefois, vous ne pouvez pas utiliser une instruction DML pour modifier une table permanente si l’instruction fait référence à une table temporaire. Par exemple, vous ne pouvez pas utiliser une instruction `INSERT ... SELECT` qui prend les données d’une table temporaire. La table temporaire existe sur le cluster secondaire et n’est pas disponible lorsque l’instruction s’exécute sur le cluster principal.

Transactions XA

Vous ne pouvez pas utiliser les instructions suivantes sur un cluster secondaire lorsque le transfert d’écriture est activé dans la session. Vous pouvez utiliser ces instructions sur des clusters secondaires pour lesquels le transfert d’écriture n’est pas activé, ou dans des sessions où le paramètre `aurora_replica_read_consistency` est vide. Avant d’activer le transfert d’écriture dans une session, vérifiez si votre code utilise ces instructions.

```
XA {START|BEGIN} xid [JOIN|RESUME]
XA END xid [SUSPEND [FOR MIGRATE]]
XA PREPARE xid
XA COMMIT xid [ONE PHASE]
XA ROLLBACK xid
XA RECOVER [CONVERT XID]
```

Instructions LOAD pour des tables permanentes

Vous ne pouvez pas utiliser les instructions suivantes sur un cluster secondaire avec le transfert d’écriture activé.

```
LOAD DATA INFILE 'data.txt' INTO TABLE t1;
        LOAD XML LOCAL INFILE 'test.xml' INTO TABLE t1;
```

Vous pouvez charger des données dans une table temporaire sur un cluster secondaire. Toutefois, assurez-vous d’exécuter toutes les instructions L0AD qui font référence aux tables permanentes uniquement sur le cluster principal.

Instructions de plugin

Vous ne pouvez pas utiliser les instructions suivantes sur un cluster secondaire avec le transfert d’écriture activé.

```
INSTALL PLUGIN example SONAME 'ha_example.so';
```

```
UNINSTALL PLUGIN example;
```

Instructions SAVEPOINT

Vous ne pouvez pas utiliser les instructions suivantes sur un cluster secondaire lorsque le transfert d'écriture est activé dans la session. Vous pouvez utiliser ces instructions sur des clusters secondaires pour lesquels le transfert d'écriture n'est pas activé, ou dans des sessions où le `aurora_replica_read_consistency` paramètre est vide. Avant d'activer le transfert d'écriture dans une session, vérifiez si votre code utilise ces instructions.

```
SAVEPOINT t1_save;
ROLLBACK TO SAVEPOINT t1_save;
RELEASE SAVEPOINT t1_save;
```

Isolement et cohérence pour le transfert d'écriture dans Aurora MySQL

Dans les sessions qui utilisent le transfert d'écriture, vous ne pouvez utiliser uniquement le niveau d'isolement `REPEATABLE READ`. Bien que vous puissiez également utiliser le niveau d'isolement `READ COMMITTED` avec des clusters en lecture seule dans des régions AWS secondaires, ce niveau d'isolement ne fonctionne pas avec le transfert d'écriture. Pour plus d'informations sur les niveaux d'isolement `REPEATABLE READ` et `READ COMMITTED`, consultez Niveaux d'isolement Aurora MySQL.

Vous pouvez contrôler le degré de cohérence en lecture sur un cluster secondaire. Le niveau de cohérence en lecture détermine la durée d'attente du cluster secondaire avant chaque opération de lecture, afin de s'assurer que certaines ou toutes les modifications sont répliquées à partir du cluster principal. Vous pouvez ajuster le niveau de cohérence en lecture pour vous assurer que toutes les opérations d'écriture transférées de votre session sont visibles dans le cluster secondaire avant toute requête ultérieure. Vous pouvez également utiliser ce paramètre pour vous assurer que les requêtes sur le cluster secondaire voient toujours les mises à jour les plus récentes du cluster principal. C'est le cas même pour celles soumises par d'autres sessions ou d'autres clusters. Pour spécifier ce type de comportement pour votre application, vous choisissez une valeur pour le paramètre de niveau session `aurora_replica_read_consistency`.

> **Important**
>
> Définissez toujours le paramètre `aurora_replica_read_consistency` pour toute session pour laquelle vous souhaitez transférer des écritures. Dans le cas contraire, Aurora

n’active pas le transfert d’écriture pour cette session. Ce paramètre a une valeur vide par défaut, alors choisissez une valeur spécifique lorsque vous utilisez ce paramètre. Le paramètre `aurora_replica_read_consistency` n’a un effet que sur les clusters secondaires dans lesquels le transfert d’écriture est activé. Pour Aurora MySQL version 2 et version 3 inférieure à 3.04, utilisez `aurora_replica_read_consistency` en tant que variable de session. Pour Aurora MySQL version 3.04 ou ultérieure, vous pouvez utiliser `aurora_replica_read_consistency` en tant que variable de session ou en tant que paramètre de cluster de bases de données.

Pour le paramètre `aurora_replica_read_consistency`, vous pouvez spécifier les valeurs `EVENTUAL`, `SESSION` et `GLOBAL`.

À mesure que vous augmentez le niveau de cohérence, votre application passe plus de temps à attendre que les modifications soient propagées entre les régions AWS. Vous pouvez choisir l’équilibre entre le temps de réponse rapide et l’assurance que les modifications apportées à d’autres emplacements sont entièrement disponibles avant l’exécution de vos requêtes.

Avec la cohérence de lecture définie sur `EVENTUAL`, les requêtes dans une région AWS secondaire qui utilise le transfert d’écriture peuvent voir des données légèrement obsolètes en raison d’un décalage de réplication. Les résultats des opérations d’écriture dans la même session ne sont pas visibles tant que l’opération d’écriture n’est pas effectuée dans la région principale et répliquée dans la région actuelle. La requête n’attend pas que les résultats mis à jour soient disponibles. Ainsi, elle peut récupérer les données plus anciennes ou les données mises à jour, en fonction de l’heure des instructions et de la durée du décalage de réplication.

Avec la cohérence en lecture définie sur `SESSION`, toutes les requêtes dans une région AWS secondaire qui utilise le transfert d’écriture voient les résultats de toutes les modifications apportées au cours de cette session. Les modifications sont visibles que la transaction soit validée ou non. Si nécessaire, la requête attend que les résultats des opérations d’écriture transférées soient répliqués dans la région actuelle. Elle n’attend pas les résultats mis à jour des opérations d’écriture effectuées dans d’autres régions ou dans d’autres sessions au sein de la région actuelle.

Avec la cohérence de lecture définie sur `GLOBAL`, une session dans une région AWS secondaire voit les modifications apportées par cette session. Elle voit également toutes les modifications engagées à partir de la région AWS principale et des autres régions AWS secondaires. Chaque requête peut attendre pendant une période qui varie en fonction du décalage de la session. La requête se poursuit

lorsque le cluster secondaire est à jour avec toutes les données validées du cluster principal, à compter du début de la requête.

Pour plus d'informations sur tous les paramètres impliqués dans le transfert d'écriture, consultez Paramètres de configuration pour le transfert d'écriture dans Aurora MySQL.

Exemples d'utilisation du transfert d'écriture

Ces exemples utilisent `aurora_replica_read_consistency` en tant que variable de session. Pour Aurora MySQL version 3.04 ou ultérieure, vous pouvez utiliser `aurora_replica_read_consistency` en tant que variable de session ou en tant que paramètre de cluster de bases de données.

Dans l'exemple suivant, le cluster principal se trouve dans la région USA Est (Virginie du Nord). Le cluster secondaire se trouve dans la région USA Est (Ohio). L'exemple montre les effets de l'exécution d'instructions `INSERT` suivies d'instructions`SELECT`. Selon la valeur du paramètre `aurora_replica_read_consistency`, les résultats peuvent différer en fonction de l'heure des instructions. Pour obtenir une plus grande cohérence, vous pouvez attendre brièvement avant d'émettre l'instruction `SELECT`. Sinon, Aurora peut attendre automatiquement la fin de la réplication des résultats avant d'effectuer l'instruction `SELECT`.

Cet exemple comporte un paramètre de cohérence en lecture de `eventual`. L'exécution d'une instruction `INSERT` immédiatement suivie d'une instruction `SELECT` renvoie toujours la valeur de `COUNT(*)`. Cette valeur reflète le nombre de lignes avant l'insertion de la nouvelle ligne. L'exécution répétée de l'instruction `SELECT` après une courte période renvoie le nombre de lignes mis à jour. Les instructions `SELECT` n'attendent pas.

```
mysql> set aurora_replica_read_consistency = 'eventual';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)
mysql> insert into t1 values (6); select count(*) from t1;
+----------+
| count(*) |
+----------+
|        5 |
```

```
+----------+
1 row in set (0.00 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

Avec un paramètre de cohérence en lecture `session`, une instruction `SELECT` immédiatement après une instruction `INSERT` attend que les modifications de l'instruction `INSERT` soient visibles. Les instructions `SELECT` suivantes n'attendent pas.

```
mysql> set aurora_replica_read_consistency = 'session';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.01 sec)
mysql> insert into t1 values (6); select count(*) from t1; select count(*) from t1;
Query OK, 1 row affected (0.08 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.37 sec)
+----------+
| count(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Le paramètre de cohérence en lecture étant toujours défini sur `session`, l'introduction d'une brève attente après l'exécution d'une instruction `INSERT` rend le nombre de lignes mis à jour disponible au moment de l'exécution de l'instruction `SELECT` suivante.

```
mysql> insert into t1 values (6); select sleep(2); select count(*) from t1;
```

```
Query OK, 1 row affected (0.07 sec)
+----------+
| sleep(2) |
+----------+
|        0 |
+----------+
1 row in set (2.01 sec)
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.00 sec)
```

Avec un paramètre de cohérence en lecture `global`, chaque instruction `SELECT` attend de s'assurer que toutes les modifications de données au début de l'instruction sont visibles avant d'effectuer la requête. La longueur de l'attente pour chaque instruction `SELECT` varie en fonction du décalage de réplication entre les clusters principal et secondaire.

```
mysql> set aurora_replica_read_consistency = 'global';
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.75 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.37 sec)
mysql> select count(*) from t1;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.66 sec)
```

Exécution d'instructions en plusieurs parties avec le transfert d'écriture dans Aurora MySQL

Une instruction DML peut être composée de plusieurs parties, notamment une instruction `INSERT ... SELECT` ou une instruction `DELETE ... WHERE`. Dans ce cas, l'instruction entière est transférée vers le cluster principal pour y être exécutée.

Transactions avec transfert d'écriture dans Aurora MySQL

Le transfert de la transaction vers le cluster principal dépend du mode d'accès de la transaction. Vous pouvez spécifier le mode d'accès de la transaction à l'aide de l'instruction `SET TRANSACTION` ou de l'instruction `START TRANSACTION`. Vous pouvez également spécifier le mode d'accès aux transactions en modifiant la valeur de la variable de session transaction_read_only. Vous pouvez modifier cette valeur de session seulement lorsque vous êtes connecté à un cluster de bases de données pour lequel le transfert d'écriture est activé.

Si une transaction de longue durée n'émet aucune instruction pendant une longue période, elle peut dépasser le délai d'inactivité. La valeur par défaut de cette période est d'une minute. Vous pouvez l'augmenter jusqu'à un jour. Une transaction qui dépasse le délai d'inactivité est annulée par le cluster principal. L'instruction suivante que vous soumettez reçoit une erreur de délai d'attente. Puis Aurora restaure la transaction.

Ce type d'erreur peut se produire dans d'autres cas, lorsque le transfert d'écriture devient indisponible. Par exemple, Aurora annule toutes les transactions qui utilisent le transfert d'écriture si vous redémarrez le cluster principal ou si vous désactivez le paramètre de configuration de transfert d'écriture.

Paramètres de configuration pour le transfert d'écriture dans Aurora MySQL

Les groupes de paramètres de cluster Aurora incluent des paramètres pour la fonction de transfert d'écriture. Comme il s'agit de paramètres de cluster, toutes les instances de base de données de chaque cluster ont les mêmes valeurs pour ces variables. Les détails sur ces paramètres sont résumés dans le tableau suivant, avec des notes d'utilisation après le tableau.

Nom	Portée	Type	Valeur par défaut	Valeurs valides
`aurora_fwd_master_idle_timeout` (Aurora MySQL version 2)	Globale	entier non signé	60	1–86 400

Nom	Portée	Type	Valeur par défaut	Valeurs valides
`aurora_fwd_master_max_connections_pct` (Aurora MySQL version 2)	Globale	entier non signé	10	0–90
`aurora_fwd_writer_idle_timeout` (Aurora MySQL version 3)	Globale	entier non signé	60	1–86 400
`aurora_fwd_writer_max_connections_pct` (Aurora MySQL version 3)	Globale	entier non signé	10	0–90
`aurora_replica_read_consistency`	Session pour version 2 et version 3 inférieure à 3.04, globale pour version 3.04 et supérieure	Enum	'' (null)	`EVENTUAL`, `SESSION`, `GLOBAL`

Pour contrôler les demandes d'écriture entrantes à partir de clusters secondaires, utilisez ces paramètres sur le cluster principal :

- `aurora_fwd_master_idle_timeout`, `aurora_fwd_writer_idle_timeout` : nombre de secondes pendant lesquelles le cluster principal attend l'activité d'une connexion transférée à partir d'un cluster secondaire avant de la fermer. Si la session reste inactive au-delà de cette période, Aurora l'annule.
- `aurora_fwd_master_max_connections_pct`, `aurora_fwd_writer_max_connections_pct` : limite supérieure des connexions à la base de données que l'on peut utiliser sur une instance de base de données de rédacteur pour gérer les requêtes transmises à partir des lecteurs. Il est

exprimé en pourcentage du paramètre `max_connections` pour l'instance de base de données du rédacteur dans le cluster principal. Par exemple, si la valeur `max_connections` est 800 et `aurora_fwd_master_max_connections_pct` ou `aurora_fwd_writer_max_connections_pct` 10, le rédacteur autorise un maximum de 80 sessions transférées simultanées. Ces connexions proviennent du même groupe de connexions géré par le paramètre `max_connections`.

Ce paramètre s'applique uniquement sur le cluster principal, lorsqu'un ou plusieurs clusters secondaires ont le transfert d'écriture activé. Si vous diminuez la valeur, les connexions existantes ne sont pas affectées. Aurora prend en compte la nouvelle valeur du paramètre lors de la tentative de création d'une nouvelle connexion à partir d'un cluster secondaire. La valeur par défaut est 10, ce qui représente 10 % de la valeur `max_connections`. Si vous activez le transfert de requêtes sur l'un des clusters secondaires, ce paramètre doit avoir une valeur différente de zéro pour les opérations d'écriture à partir de clusters secondaires pour réussir. Si la valeur est zéro, les opérations d'écriture reçoivent le code d'erreur `ER_CON_COUNT_ERROR` avec le message `Not enough connections on writer to handle your request`.

Le paramètre `aurora_replica_read_consistency` active le transfert d'écriture. Vous l'utilisez dans chaque session. Vous pouvez spécifier `EVENTUAL`, `SESSION` ou `GLOBAL` pour le niveau de cohérence de lecture. Pour en savoir plus sur les niveaux de cohérence, consultez la section Isolement et cohérence pour le transfert d'écriture dans Aurora MySQL. Les règles suivantes s'appliquent à ce paramètre :

- La valeur par défaut est '' (vide).
- Le transfert d'écriture est seulement disponible dans une session si `aurora_replica_read_consistency` est défini sur `EVENTUAL` ou `SESSION` ou `GLOBAL`. Ce paramètre n'est pertinent que dans les instances de lecteur de clusters secondaires dont le transfert d'écriture est activé et qui se trouvent dans une base de données Aurora globale.
- Vous ne pouvez pas définir cette variable (lorsqu'elle est vide) ou supprimer sa définition (lorsqu'elle est déjà définie) dans une transaction multi-instructions. Toutefois, vous pouvez en modifier la valeur , en passant d'une valeur valide (`EVENTUAL`, `SESSION` ou `GLOBAL`) à une autre valeur valide (`EVENTUAL`, `SESSION` ou `GLOBAL`) lors d'une telle transaction.
- La variable ne peut pas être `SET` lorsque le transfert d'écriture n'est pas activé sur le cluster secondaire.

Métriques Amazon CloudWatch pour le transfert d’écriture dans Aurora MySQL

Les métriques Amazon CloudWatch suivantes s’appliquent au cluster principal lorsque vous utilisez le transfert d’écriture sur un ou plusieurs clusters secondaires. Ces métriques sont toutes mesurées sur l’instance de base de données d’enregistreur dans le cluster principal.

Métrique CloudWatch	Unité	Description
`AuroraDMLRejectedMasterFull`	Nombre	Nombre de requêtes transférées qui sont rejetées, car la session est pleine sur l’instance de base de données d’enregistreur. Pour Aurora MySQL version 2
`AuroraDMLRejectedWriterFull`	Nombre	Nombre de requêtes transférées qui sont rejetées, car la session est pleine sur l’instance de base de données d’enregistreur. Pour Aurora MySQL version 3.
`ForwardingMasterDMLLatency`	Millisecondes	Temps moyen pour traiter chaque instruction DML transférée sur l’instance de base de données d’enregistreur. Il n’inclut pas le temps nécessaire au cluster secondaire pour transférer la demande d’écriture, ni le temps nécessaire pour répliquer les modifications et les renvoyer au cluster secondaire.

Métrique CloudWatch	Unité	Description
		Pour Aurora MySQL version 2
`ForwardingMasterDMLThroughput`	Nombre par seconde	Nombre d'instructions DML transférées traitées chaque seconde par cette instance de base de données d'enregistreur. Pour Aurora MySQL version 2
`ForwardingMasterOpenSessions`	Nombre	Nombre de sessions transférées sur l'instance de base de données d'enregistreur. Pour Aurora MySQL version 2
`ForwardingWriterDMLLatency`	Millisecondes	Temps moyen pour traiter chaque instruction DML transférée sur l'instance de base de données d'enregistreur. Il n'inclut pas le temps nécessaire au cluster secondaire pour transférer la demande d'écriture, ni le temps nécessaire pour répliquer les modifications et les renvoyer au cluster secondaire. Pour Aurora MySQL version 3.

Métrique CloudWatch	Unité	Description
`ForwardingWriterDMLThroughput`	Nombre par seconde	Nombre d'instructions DML transférées traitées chaque seconde par cette instance de base de données d'enregistreur. Pour Aurora MySQL version 3.
`ForwardingWriterOpenSessions`	Nombre	Nombre de sessions transférées sur l'instance de base de données d'enregistreur. Pour Aurora MySQL version 3.

Les métriques CloudWatch suivantes s'appliquent à chaque cluster secondaire. Ces métriques sont mesurées sur chaque instance de base de données de lecteur dans un cluster secondaire avec le transfert d'écriture activé.

Métrique CloudWatch	Unité	Description
`ForwardingReplicaDMLLatency`	Millisecondes	Temps de réponse moyen des DML transférées sur le réplica.
`ForwardingReplicaDMLThroughput`	Nombre par seconde	Nombre d'instructions DML transférées traitées par seconde.
`ForwardingReplicaOpenSessions`	Nombre	Nombre de sessions qui utilisent le transfert d'écriture sur une instance de base de données de lecteur.
`ForwardingReplicaReadWaitLatency`	Millisecondes	Temps moyen qu'une instruction `SELECT` sur une instance de base de données de lecteur attend pour rattraper le cluster principal.

Métrique CloudWatch	Unité	Description
		La longueur de l'attente de l'instance de base de données du lecteur avant de traiter une requête dépend du paramètre `aurora_replica_read_consistency` .
`ForwardingReplicaReadWaitThroughput`	Nombre par seconde	Nombre total d'instructions `SELECT` traitées chaque seconde dans toutes les sessions qui transportent des écritures.
`ForwardingReplicaSelectLatency`	Millisecondes	Latence `SELECT` de transmission, moyenne sur toutes les instructions `SELECT` transmises au cours de la période de surveillance.
`ForwardingReplicaSelectThroughput`	Nombre par seconde	Débit d'instruction `SELECT` transférée par seconde, dont la moyenne est calculée au cours de la période de surveillance.

Variables d'état Aurora MySQL pour le transfert d'écriture

Les variables d'état Aurora MySQL suivantes s'appliquent au cluster principal lorsque vous utilisez le transfert d'écriture sur un ou plusieurs clusters secondaires. Ces métriques sont toutes mesurées sur l'instance de base de données d'enregistreur dans le cluster principal.

Variable d'état Aurora MySQL	Unité	Description
`Aurora_fwd_master_dml_stmt_count`	Nombre	Nombre total d'instructions DML transférées à cette

Variable d’état Aurora MySQL	Unité	Description
		instance de base de données d’enregistreur. Pour Aurora MySQL version 2
`Aurora_fwd_master_dml_stmt_duration`	Microsecondes	Durée totale des instructions DML transférées à cette instance de base de données d’enregistreur. Pour Aurora MySQL version 2
`Aurora_fwd_master_open_sessions`	Nombre	Nombre de sessions transférées sur l’instance de base de données d’enregistreur. Pour Aurora MySQL version 2
`Aurora_fwd_master_select_stmt_count`	Nombre	Nombre total d’instructions `SELECT` transférées à cette instance de base de données d’enregistreur. Pour Aurora MySQL version 2
`Aurora_fwd_master_select_stmt_duration`	Microsecondes	Durée totale des instructions `SELECT` transférées à cette instance de base de données d’enregistreur. Pour Aurora MySQL version 2
`Aurora_fwd_writer_dml_stmt_count`	Nombre	Nombre total d’instructions DML transférées à cette instance de base de données d’enregistreur. Pour Aurora MySQL version 3.

Variable d’état Aurora MySQL	Unité	Description
`Aurora_fwd_writer_dml_stmt_duration`	Microsecondes	Durée totale des instructions DML transférées à cette instance de base de données d’enregistreur.
`Aurora_fwd_writer_open_sessions`	Nombre	Nombre de sessions transférées sur l’instance de base de données d’enregistreur. Pour Aurora MySQL version 3.
`Aurora_fwd_writer_select_stmt_count`	Nombre	Nombre total d’instructions `SELECT` transférées à cette instance de base de données d’enregistreur. Pour Aurora MySQL version 3.
`Aurora_fwd_writer_select_stmt_duration`	Microsecondes	Durée totale des instructions `SELECT` transférées à cette instance de base de données d’enregistreur. Pour Aurora MySQL version 3.

Les variables d’état Aurora MySQL suivantes s’appliquent à chaque cluster secondaire. Ces métriques sont mesurées sur chaque instance de base de données de lecteur dans un cluster secondaire avec le transfert d’écriture activé.

Variable d’état Aurora MySQL	Unité	Description
`Aurora_fwd_replica_dml_stmt_count`	Nombre	Nombre total d’instructions DML transférées à partir de cette instance de base de données de lecteur.

Variable d’état Aurora MySQL	Unité	Description
`Aurora_fwd_replica_dml_stmt_duration`	Microsecondes	Durée totale de toutes les instructions DML transférées à partir de cette instance de base de données de lecteur.
`Aurora_fwd_replica_errors_session_limit`	Nombre	Nombre de sessions rejetées par le cluster principal en raison de l’une des conditions d’erreur suivantes : • enregistreur complet • Trop d’instructions transférées en cours
`Aurora_fwd_replica_open_sessions`	Nombre	Nombre de sessions qui utilisent le transfert d’écriture sur une instance de base de données de lecteur.
`Aurora_fwd_replica_read_wait_count`	Nombre	Nombre total d’attentes en lecture-après écriture sur cette instance de base de données de lecteur.
`Aurora_fwd_replica_read_wait_duration`	Microsecondes	Durée totale des attentes dues au paramètre de cohérence en lecture sur cette instance de base de données de lecteur.
`Aurora_fwd_replica_select_stmt_count`	Nombre	Nombre total d’instructions `SELECT` transférées à partir de cette instance de base de données de lecteur.

Variable d'état Aurora MySQL	Unité	Description
`Aurora_fwd_replica_select_stmt_duration`	Microsecondes	Durée totale des instructions `SELECT` transférées à partir de cette instance de base de données de lecteur.

Utilisation du transfert d'écriture dans une base de données globale Aurora PostgreSQL

Rubriques

Régions et versions disponibles pour le transfert d'écriture dans Aurora PostgreSQL

Dans Aurora PostgreSQL 16 et versions ultérieures, le transfert d'écriture global est pris en charge dans toutes les versions mineures. Dans les versions antérieures d'Aurora PostgreSQL, le transfert d'écriture est pris en charge dans les versions 14.9 et 15.4 ainsi que leurs versions mineures ultérieures. Cette fonctionnalité est disponible dans toutes les régions AWS où les bases de données globales Aurora PostgreSQL sont présentes.

Pour en savoir plus sur les régions et les versions disponibles des bases de données globales Aurora PostgreSQL, consultez Bases de données globales Aurora avec Aurora PostgreSQL.

Activation du transfert d’écriture dans Aurora PostgreSQL

Par défaut, le transfert d’écriture n’est pas activé lorsque vous ajoutez un cluster secondaire à une base de données globale Aurora. Vous pouvez l’activer pendant que vous créez votre cluster de bases de données secondaire ou ultérieurement à tout moment. Si nécessaire, vous pouvez le désactiver plus tard. L’activation ou la désactivation du transfert d’écriture n’entraîne pas de durée d’indisponibilité ni de redémarrage.

> **Note**
>
> Vous pouvez utiliser le transfert d’écriture local pour les applications qui font l’objet d’écritures occasionnelles et qui nécessitent une cohérence de lecture après écriture, ce qui permet de lire la dernière écriture dans une transaction.

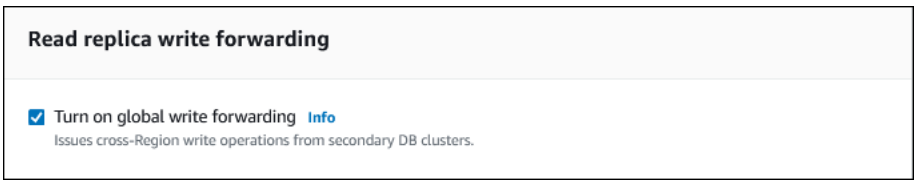
Console

Dans la console, vous pouvez activer ou désactiver le transfert d’écriture lorsque vous créez ou modifiez un cluster de bases de données secondaire.

Activation ou désactivation du transfert d’écriture lors de la création d’un cluster de bases de données secondaire

Lorsque vous créez un cluster de bases de données secondaire, activez le transfert d’écriture en cochant la case Activer le transfert d’écriture global sous Transfert d’écriture de réplica en lecture. Ou décochez la case pour le désactiver. Pour créer un cluster de bases de données secondaire, suivez les instructions pour votre moteur de base de données dans Création d’un cluster de bases de données Amazon Aurora.

La capture d’écran suivante montre la section Transfert d’écriture de réplica en lecture avec la case Activer le transfert d’écriture global cochée.



Activation ou désactivation du transfert d'écriture lors de la modification d'un cluster de bases de données secondaire

Dans la console, vous pouvez modifier un cluster de bases de données secondaire pour activer ou désactiver le transfert d'écriture.

Pour activer ou désactiver le transfert d'écriture sur un cluster de bases de données secondaire à l'aide de la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données.
3. Choisissez le cluster de bases de données secondaire, puis Modifier.
4. Dans la section Transfert d'écriture de réplica en lecture, cochez ou décochez la case Activer le transfert d'écriture global.
5. Choisissez Continuer.
6. Pour Planification des modifications, choisissez Appliquer immédiatement. Si vous choisissez Au cours de la prochaine fenêtre de maintenance planifiée, Aurora ignore ce paramètre et active immédiatement le transfert d'écriture.
7. Choisissez Modifier le cluster.

AWS CLI

Pour activer le transfert d'écriture à l'aide de l'AWS CLI, utilisez l'option `--enable-global-write-forwarding`. Cette option est utile lorsque vous créez un nouveau cluster secondaire à l'aide de la commande create-db-cluster. Elle est également utile lorsque vous modifiez un cluster secondaire à l'aide de la commande modify-db-cluster. Elle nécessite que la base de données globale utilise une version d'Aurora qui prend en charge le transfert d'écriture. Vous pouvez désactiver le transfert d'écriture en utilisant l'option `--no-enable-global-write-forwarding` avec ces mêmes commandes CLI.

Les procédures suivantes expliquent comment activer ou désactiver le transfert d'écriture sur un cluster de bases de données secondaire dans votre cluster global à l'aide d'AWS CLI.

Pour activer ou désactiver le transfert d'écriture d'un cluster de bases de données secondaire

- Appelez la commande modify-db-cluster de l'AWS CLI et indiquez les valeurs suivantes :

- `--db-cluster-identifier` : nom du cluster de bases de données.
- `--enable-global-write-forwarding` pour l'activer ou `--no-enable-global-write-forwarding` pour le désactiver.

L'exemple suivant active le transfert d'écriture pour le cluster de bases de données `sample-secondary-db-cluster`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-secondary-db-cluster \
    --enable-global-write-forwarding
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-secondary-db-cluster ^
    --enable-global-write-forwarding
```

API RDS

Pour activer le transfert d'écriture à l'aide de l'API Amazon RDS, définissez le paramètre `EnableGlobalWriteForwarding` sur `true`. Ce paramètre s'applique lorsque vous créez un cluster secondaire à l'aide de l'opération CreateDBCluster. Il s'applique également lorsque vous modifiez un cluster secondaire à l'aide de l'opération ModifyDBCluster. Il nécessite que la base de données globale utilise une version d'Aurora qui prend en charge le transfert d'écriture. Vous pouvez désactiver le transfert d'écriture en définissant le paramètre `EnableGlobalWriteForwarding` sur `false`.

Vérification de l'activation du transfert d'écriture dans un cluster secondaire dans Aurora PostgreSQL

Pour déterminer si vous pouvez utiliser le transfert d'écriture à partir d'un cluster secondaire, vous pouvez vérifier si le cluster possède l'attribut `"GlobalWriteForwardingStatus": "enabled"`.

Dans la AWS Management Console, vous voyez `Read replica write forwarding` dans l'onglet Configuration de la page de détails du cluster. Pour voir l'état du paramètre global de transfert d'écriture pour tous vos clusters, exécutez la commande d'AWS CLI suivante.

Un cluster secondaire affiche la valeur `"enabled"` ou `"disabled"` pour indiquer si le transfert d'écriture est activé ou désactivé. La valeur `null` indique que le transfert d'écriture n'est pas disponible pour ce cluster. Soit le cluster ne fait pas partie d'une base de données globale, soit il s'agit du cluster principal et non d'un cluster secondaire. La valeur peut également être `"enabling"` ou `"disabling"` si le transfert d'écriture est en cours d'activation ou de désactivation.

Example

```
aws rds describe-db-clusters --query '*[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
[
    {
        "GlobalWriteForwardingStatus": "enabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    },
    {
        "GlobalWriteForwardingStatus": "disabled",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-2"
    },
    {
        "GlobalWriteForwardingStatus": null,
        "DBClusterIdentifier": "non-global-cluster"
    }
]
```

Pour rechercher uniquement les clusters secondaires pour lesquels le transfert d'écriture global est activé, exécutez la commande suivante. Cette commande renvoie également le point de terminaison du lecteur du cluster. Vous utilisez le point de terminaison du lecteur du cluster secondaire lorsque vous utilisez le transfert d'écriture du secondaire vers le principal dans votre base de données Aurora globale.

Example

```
aws rds describe-db-clusters --query 'DBClusters[].
{DBClusterIdentifier:DBClusterIdentifier,GlobalWriteForwardingStatus:GlobalWriteForwardingStatu
 | [?GlobalWriteForwardingStatus == `enabled`]'
[
```

```
    {
        "GlobalWriteForwardingStatus": "enabled",
        "ReaderEndpoint": "aurora-write-forwarding-test-replica-1.cluster-ro-
cnpexample.us-west-2.rds.amazonaws.com",
        "DBClusterIdentifier": "aurora-write-forwarding-test-replica-1"
    }
]
```

Compatibilité des applications et de SQL avec le transfert d'écriture dans Aurora PostgreSQ

Utilisées dans une base de données globale avec transfert d'écriture, certaines instructions ne sont pas autorisées ou peuvent produire des résultats obsolètes. En outre, les fonctions et les procédures définies par l'utilisateur ne sont pas prises en charge. Par conséquent, le paramètre `EnableGlobalWriteForwarding` est désactivé par défaut pour les clusters secondaires. Avant de l'activer, vérifiez que votre code d'application n'est affecté par aucune de ces restrictions.

Vous pouvez utiliser les types d'instructions SQL suivants avec le transfert d'écriture :

- Instructions DML (Data Manipulation Language) comme `INSERT`, `DELETE` et `UPDATE`
- Instructions `SELECT FOR { UPDATE | NO KEY UPDATE | SHARE | KEY SHARE }`
- Instructions `PREPARE` et `EXECUTE`
- Instructions `EXPLAIN` comprenant les instructions de cette liste

Les types d'instructions SQL suivants ne sont pas pris en charge par le transfert d'écriture :

- Instructions DDL (Data Definition Language)
- `ANALYZE`
- `CLUSTER`
- `COPY`
- Curseurs : les curseurs ne sont pas pris en charge. Assurez-vous donc de les fermer avant d'utiliser le transfert d'écriture.
- `GRANT|REVOKE|REASSIGN OWNED|SECURITY LABEL`
- `LOCK`
- Instructions `SAVEPOINT`

- `SELECT INTO`
- `SET CONSTRAINTS`
- `TRUNCATE`
- `VACUUM`

Isolement et cohérence pour le transfert d'écriture dans Aurora PostgreSQL

Dans les sessions qui utilisent le transfert d'écriture, vous pouvez utiliser les niveaux d'isolement `REPEATABLE READ` et `READ COMMITTED`. En revanche, le niveau d'isolement `SERIALIZABLE` n'est pas pris en charge.

Vous pouvez contrôler le degré de cohérence en lecture sur un cluster secondaire. Le niveau de cohérence en lecture détermine la durée d'attente du cluster secondaire avant chaque opération de lecture, afin de s'assurer que certaines ou toutes les modifications sont répliquées à partir du cluster principal. Vous pouvez ajuster le niveau de cohérence en lecture pour vous assurer que toutes les opérations d'écriture transférées de votre session sont visibles dans le cluster secondaire avant toute requête ultérieure. Vous pouvez également utiliser ce paramètre pour vous assurer que les requêtes sur le cluster secondaire voient toujours les mises à jour les plus récentes du cluster principal. C'est le cas même pour celles soumises par d'autres sessions ou d'autres clusters. Pour spécifier ce type de comportement pour votre application, choisissez la valeur appropriée pour le paramètre `apg_write_forward.consistency_mode`. Le paramètre `apg_write_forward.consistency_mode` n'a un effet que sur les clusters secondaires dans lesquels le transfert d'écriture est activé.

> **Note**
>
> Pour le paramètre `apg_write_forward.consistency_mode`, vous pouvez spécifier les valeurs `SESSION`, `EVENTUAL`, `GLOBAL` ou `OFF`. Par défaut, cette valeur indique `SESSION`. Le réglage de la valeur sur `OFF` désactive le transfert d'écriture dans la session.

À mesure que vous augmentez le niveau de cohérence, votre application passe plus de temps à attendre que les modifications soient propagées entre les régions AWS. Vous pouvez choisir l'équilibre entre le temps de réponse rapide et l'assurance que les modifications apportées à d'autres emplacements sont entièrement disponibles avant l'exécution de vos requêtes.

Pour chaque paramètre de cohérence disponible, l'effet est le suivant :

- `SESSION` : toutes les requêtes dans une région AWS secondaire qui utilise le transfert d'écriture voient les résultats de toutes les modifications apportées au cours de cette session. Les modifications sont visibles que la transaction soit validée ou non. Si nécessaire, la requête attend que les résultats des opérations d'écriture transférées soient répliqués dans la région actuelle. Elle n'attend pas les résultats mis à jour des opérations d'écriture effectuées dans d'autres régions ou dans d'autres sessions au sein de la région actuelle.
- `EVENTUAL` : les requêtes dans une région AWS secondaire qui utilise le transfert d'écriture peuvent voir des données légèrement obsolètes en raison d'un décalage de réplication. Les résultats des opérations d'écriture dans la même session ne sont pas visibles tant que l'opération d'écriture n'est pas effectuée dans la région principale et répliquée dans la région actuelle. La requête n'attend pas que les résultats mis à jour soient disponibles. Ainsi, elle peut récupérer les données plus anciennes ou les données mises à jour, en fonction de l'heure des instructions et de la durée du décalage de réplication.
- `GLOBAL` : une session dans une région AWS secondaire voit les modifications apportées par cette session. Elle voit également toutes les modifications engagées à partir de la région AWS principale et des autres régions AWS secondaires. Chaque requête peut attendre pendant une période qui varie en fonction du décalage de la session. La requête se poursuit lorsque le cluster secondaire est à jour avec toutes les données validées du cluster principal, à compter du début de la requête.
- `OFF` : le transfert d'écriture est désactivé dans la session.

Pour plus d'informations sur tous les paramètres impliqués dans le transfert d'écriture, consultez Paramètres de configuration pour le transfert d'écriture dans Aurora PostgreSQL.

Modes d'accès aux transactions avec transfert d'écriture

Si le mode d'accès aux transactions est réglé sur lecture seule, le transfert d'écriture n'est pas utilisé. Vous pouvez définir le mode d'accès en lecture et en écriture seules lorsque vous êtes connecté à un cluster de bases de données pour lequel le transfert d'écriture est activé.

Pour plus d'informations sur les modes d'accès aux transactions, consultez SET TRANSACTION.

Exécution d'instructions en plusieurs parties avec le transfert d'écriture dans Aurora PostgreSQL

Une instruction DML peut être composée de plusieurs parties, notamment une instruction `INSERT ... SELECT` ou une instruction `DELETE ... WHERE`. Dans ce cas, l'instruction entière est transférée vers le cluster principal pour y être exécutée.

Paramètres de configuration pour le transfert d’écriture dans Aurora PostgreSQL

Les groupes de paramètres de cluster Aurora incluent des paramètres pour la fonction de transfert d’écriture. Comme il s’agit de paramètres de cluster, toutes les instances de base de données de chaque cluster ont les mêmes valeurs pour ces variables. Les détails sur ces paramètres sont résumés dans le tableau suivant, avec des notes d’utilisation après le tableau.

Nom	Portée	Type	Valeur par défaut	Valeurs valides
`apg_write_forward.connect_timeout`	Session	secondes	30	0–2147483647
`apg_write_forward.consistency_mode`	Session	enum	Session	`SESSION`, `EVENTUAL`, `GLOBAL`, `OFF`
`apg_write_forward.idle_in_transaction_session_timeout`	Session	millisecondes	86400000	0–2147483647
`apg_write_forward.idle_session_timeout`	Session	millisecondes	300000	0–2147483647
`apg_write_forward.max_forwarding_connections_percent`	Globale	int	25	1–100

Le paramètre `apg_write_forward.max_forwarding_connections_percent` est la limite supérieure des emplacements de connexion à la base de données qui peuvent être utilisés pour traiter les requêtes transmises par les lecteurs. Il est exprimé en pourcentage du paramètre `max_connections` de l’instance de base de données d’enregistreur dans le cluster principal. Par exemple, si la valeur de `max_connections` est `800` et celle de `apg_write_forward.max_forwarding_connections_percent` est `10`, l’enregistreur autorise un maximum de 80 sessions transférées simultanées. Ces connexions proviennent du même groupe de connexions géré par le paramètre `max_connections`. Ce paramètre s’applique uniquement au cluster principal, lorsque le transfert d’écriture est activé dans au moins un cluster secondaire.

Utilisez les paramètres suivants sur le cluster secondaire :

- `apg_write_forward.consistency_mode` : paramètre de niveau session qui contrôle le degré de cohérence en lecture sur le cluster secondaire. Les valeurs valides sont `SESSION`, `EVENTUAL`, `GLOBAL` ou `OFF`. Par défaut, cette valeur indique `SESSION`. Le réglage de la valeur sur `OFF` désactive le transfert d'écriture dans la session. Pour en savoir plus sur les niveaux de cohérence, consultez Isolement et cohérence pour le transfert d'écriture dans Aurora PostgreSQL. Ce paramètre n'est pertinent que dans les instances de lecteur de clusters secondaires dont le transfert d'écriture est activé et qui se trouvent dans une base de données Aurora globale.
- `apg_write_forward.connect_timeout` : nombre maximal de secondes pendant lesquelles le cluster secondaire attend lors de l'établissement d'une connexion au cluster principal avant d'abandonner. Une valeur de `0` correspond à un temps d'attente indéfini.
- `apg_write_forward.idle_in_transaction_session_timeout` : nombre de millisecondes pendant lesquelles le cluster principal attend une activité sur une connexion transférée à partir d'un cluster secondaire ayant une transaction ouverte avant de la fermer. Si la session reste inactive au-delà de cette durée, Aurora y met fin. La valeur `0` désactive le délai d'attente.
- `apg_write_forward.idle_session_timeout` : nombre de millisecondes pendant lesquelles le cluster principal attend une activité sur une connexion transférée à partir d'un cluster secondaire avant de la fermer. Si la session reste inactive au-delà de cette durée, Aurora y met fin. La valeur `0` désactive le délai d'attente.

Métriques Amazon CloudWatch pour le transfert d'écriture dans Aurora PostgreSQL

Les métriques Amazon CloudWatch suivantes s'appliquent au cluster principal lorsque vous utilisez le transfert d'écriture sur un ou plusieurs clusters secondaires. Ces métriques sont toutes mesurées sur l'instance de base de données d'enregistreur dans le cluster principal.

Métrique CloudWatch	Unités et description
`AuroraForwardingWriterDMLThroughput`	Nombre (par seconde). Nombre d'instructions DML transférées traitées chaque seconde par cette instance de base de données d'enregistreur.
`AuroraForwardingWriterOpenSessions`	Nombre. Nombre de sessions ouvertes sur cette instance de base de données d'enregistreur traitant les requêtes transmises.

Métrique CloudWatch	Unités et description
`AuroraForwardingWriterTotalSessions`	Nombre. Nombre total de sessions transférées sur cette instance de base de données d'enregistreur.

Les métriques CloudWatch suivantes s'appliquent à chaque cluster secondaire. Ces métriques sont mesurées sur chaque instance de base de données de lecteur dans un cluster secondaire avec le transfert d'écriture activé.

Métrique CloudWatch	Unité et description
`AuroraForwardingReplicaCommitThroughput`	Nombre (par seconde). Nombre d'engagements dans les sessions transmises chaque seconde par ce réplica.
`AuroraForwardingReplicaDMLLatency`	Millisecondes. Temps de réponse moyen en millisecondes des DML transférées sur le réplica.
`AuroraForwardingReplicaDMLThroughput`	Nombre (par seconde). Nombre d'instructions DML transférées que ce réplica traite chaque seconde.
`AuroraForwardingReplicaErrorSessionsLimit`	Nombre. Nombre de sessions rejetées par le cluster principal, car le nombre maximal de connexions ou de connexions de transfert d'écriture a été atteint.
`AuroraForwardingReplicaOpenSessions`	Nombre. Nombre de sessions qui utilisent le transfert d'écriture sur une instance de réplica.
`AuroraForwardingReplicaReadWaitLatency`	Millisecondes. Durée moyenne en millisecondes que le réplica attend pour être cohérent avec le LSN du cluster principal. Le temps d'attente de l'instance de base de données de lecteur dépend du paramètre `apg_write`

Métrique CloudWatch	Unité et description
	`_forward.consistency_mode` . Pour plus d'informations sur ce paramètre, consultez the section called "Paramètres de configuration pour le transfert d'écriture dans Aurora PostgreSQL".

Événements d'attente pour le transfert d'écriture dans Aurora PostgreSQL

Amazon Aurora génère les événements d'attente suivants lorsque vous utilisez le transfert d'écriture avec Aurora PostgreSQL.

Rubriques

- IPC:AuroraWriteForwardConnect
- IPC:AuroraWriteForwardConsistencyPoint
- IPC:AuroraWriteForwardExecute
- IPC:AuroraWriteForwardGetGlobalConsistencyPoint
- IPC:AuroraWriteForwardXactAbort
- IPC:AuroraWriteForwardXactCommit
- IPC:AuroraWriteForwardXactStart

IPC:AuroraWriteForwardConnect

L'événement `IPC:AuroraWriteForwardConnect` se produit lorsqu'un processus dorsal sur le cluster de bases de données secondaire attend l'ouverture d'une connexion au nœud d'enregistreur du cluster de bases de données principal.

Causes probables de l'augmentation du nombre d'événements d'attente

Cet événement augmente à mesure que le nombre de tentatives de connexion du nœud de lecteur d'une région secondaire au nœud d'enregistreur du cluster de bases de données principal augmente.

Actions

Réduisez le nombre de connexions simultanées du nœud secondaire au nœud d'enregistreur de la région principale.

IPC:AuroraWriteForwardConsistencyPoint

L'événement `IPC:AuroraWriteForwardConsistencyPoint` décrit la durée pendant laquelle une requête d'un nœud du cluster de bases de données secondaire attend la réplication des résultats des opérations d'écriture transférées dans la région actuelle. Cet événement n'est généré que si le paramètre de niveau session `apg_write_forward.consistency_mode` est défini sur l'une des valeurs suivantes :

- `SESSION` : les requêtes d'un nœud secondaire attendent les résultats de toutes les modifications apportées au cours de cette session.
- `GLOBAL` : les requêtes d'un nœud secondaire attendent les résultats des modifications apportées par cette session, ainsi que toutes les modifications validées de la région principale et des autres régions secondaires du cluster global.

Pour plus d'informations sur la configuration du paramètre `apg_write_forward.consistency_mode`, consultez the section called "Paramètres de configuration pour le transfert d'écriture dans Aurora PostgreSQL".

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'allongement des temps d'attente sont les suivantes :

- Augmentation du retard de réplica, tel que mesuré par la métrique `ReplicaLag` d'Amazon CloudWatch. Pour plus d'informations sur cette métrique, consultez Surveillance de la réplication Aurora PostgreSQL.
- Charge accrue sur le nœud d'enregistreur de la région principale ou sur le nœud secondaire.

Actions

Modifiez votre mode de cohérence en fonction des besoins de votre application.

IPC:AuroraWriteForwardExecute

L'événement `IPC:AuroraWriteForwardExecute` se produit lorsqu'un processus dorsal sur le cluster de bases de données secondaire attend qu'une requête transférée se termine et obtienne des résultats du nœud d'enregistreur du cluster de bases de données principal.

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'augmentation du nombre d'événements d'attente sont les suivantes :

- Un grand nombre de lignes est récupéré du nœud d’enregistreur de la région principale.
- Une augmentation de la latence du réseau entre le nœud secondaire et le nœud d’enregistreur de la région principale augmente le temps nécessaire au nœud secondaire pour recevoir les données du nœud d’enregistreur.
- Une augmentation de la charge sur le nœud secondaire peut retarder la transmission de la requête du nœud secondaire au nœud d’enregistreur de la région principale.
- Une augmentation de la charge sur le nœud d’enregistreur de la région principale peut retarder la transmission des données du nœud d’enregistreur au nœud secondaire.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d’attente.

- Optimisez les requêtes pour récupérer uniquement les données nécessaires.
- Optimisez les opérations DML (Data Manipulation Language) pour ne modifier que les données nécessaires.
- Si le nœud secondaire ou le nœud d’enregistreur de la région principale est limité par le processeur ou la bande passante du réseau, envisagez de le remplacer par un type d’instance doté d’une plus grande capacité de processeur ou de bande passante.

IPC:AuroraWriteForwardGetGlobalConsistencyPoint

L’événement `IPC:AuroraWriteForwardGetGlobalConsistencyPoint` se produit lorsqu’un processus dorsal sur le cluster de bases de données secondaire qui utilise le mode de cohérence GLOBAL attend d’obtenir le point de cohérence global auprès du nœud d’enregistreur avant d’exécuter une requête.

Causes probables de l’augmentation du nombre d’événements d’attente

Les causes fréquentes de l’augmentation du nombre d’événements d’attente sont les suivantes :

- Une augmentation de la latence du réseau entre le nœud secondaire et le nœud d’enregistreur de la région principale augmente le temps nécessaire au nœud secondaire pour recevoir les données du nœud d’enregistreur.
- Une augmentation de la charge sur le nœud secondaire peut retarder la transmission de la requête du nœud secondaire au nœud d’enregistreur de la région principale.

- Une augmentation de la charge sur le nœud d'enregistreur de la région principale peut retarder la transmission des données du nœud d'enregistreur au nœud secondaire.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

- Modifiez votre mode de cohérence en fonction des besoins de votre application.
- Si le nœud secondaire ou le nœud d'enregistreur de la région principale est limité par le processeur ou la bande passante du réseau, envisagez de le remplacer par un type d'instance doté d'une plus grande capacité de processeur ou de bande passante.

IPC:AuroraWriteForwardXactAbort

L'événement `IPC:AuroraWriteForwardXactAbort` se produit lorsqu'un processus dorsal sur le cluster de bases de données secondaire attend le résultat d'une requête de nettoyage à distance. Des requêtes de nettoyage sont émises pour remettre le processus dans l'état approprié après l'abandon d'une transaction de transfert d'écriture. Amazon Aurora les exécute soit parce qu'une erreur a été détectée, soit parce qu'un utilisateur a émis une commande `ABORT` explicite ou annulé une requête en cours d'exécution.

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'augmentation du nombre d'événements d'attente sont les suivantes :

- Une augmentation de la latence du réseau entre le nœud secondaire et le nœud d'enregistreur de la région principale augmente le temps nécessaire au nœud secondaire pour recevoir les données du nœud d'enregistreur.
- Une augmentation de la charge sur le nœud secondaire peut retarder la transmission de la requête de nettoyage du nœud secondaire au nœud d'enregistreur de la région principale.
- Une augmentation de la charge sur le nœud d'enregistreur de la région principale peut retarder la transmission des données du nœud d'enregistreur au nœud secondaire.

Actions

Nous vous recommandons différentes actions en fonction des causes de votre événement d'attente.

- Recherchez la cause de l'annulation de la transaction.

- Si le nœud secondaire ou le nœud d'enregistreur de la région principale est limité par le processeur ou la bande passante du réseau, envisagez de le remplacer par un type d'instance doté d'une plus grande capacité de processeur ou de bande passante.

IPC:AuroraWriteForwardXactCommit

L'événement `IPC:AuroraWriteForwardXactCommit` se produit lorsqu'un processus dorsal sur le cluster de bases de données secondaire attend le résultat d'une commande commit transaction transférée.

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'augmentation du nombre d'événements d'attente sont les suivantes :

- Une augmentation de la latence du réseau entre le nœud secondaire et le nœud d'enregistreur de la région principale augmente le temps nécessaire au nœud secondaire pour recevoir les données du nœud d'enregistreur.
- Une augmentation de la charge sur le nœud secondaire peut retarder la transmission de la requête du nœud secondaire au nœud d'enregistreur de la région principale.
- Une augmentation de la charge sur le nœud d'enregistreur de la région principale peut retarder la transmission des données du nœud d'enregistreur au nœud secondaire.

Actions

Si le nœud secondaire ou le nœud d'enregistreur de la région principale est limité par le processeur ou la bande passante du réseau, envisagez de le remplacer par un type d'instance doté d'une plus grande capacité de processeur ou de bande passante.

IPC:AuroraWriteForwardXactStart

L'événement `IPC:AuroraWriteForwardXactStart` se produit lorsqu'un processus dorsal sur le cluster de bases de données secondaire attend le résultat d'une commande start transaction transférée.

Causes probables de l'augmentation du nombre d'événements d'attente

Les causes fréquentes de l'augmentation du nombre d'événements d'attente sont les suivantes :

- Une augmentation de la latence du réseau entre le nœud secondaire et le nœud d'enregistreur de la région principale augmente le temps nécessaire au nœud secondaire pour recevoir les données du nœud d'enregistreur.
- Une augmentation de la charge sur le nœud secondaire peut retarder la transmission de la requête du nœud secondaire au nœud d'enregistreur de la région principale.
- Une augmentation de la charge sur le nœud d'enregistreur de la région principale peut retarder la transmission des données du nœud d'enregistreur au nœud secondaire.

Actions

Si le nœud secondaire ou le nœud d'enregistreur de la région principale est limité par le processeur ou la bande passante du réseau, envisagez de le remplacer par un type d'instance doté d'une plus grande capacité de processeur ou de bande passante.

Utilisation de la bascule ou du basculement dans une base de données Amazon Aurora Global Database

Une base de données Aurora Global Database fournit une protection de la continuité des activités et de la reprise après sinistre (BCDR) supérieure à la haute disponibilité standard fournie par un cluster de bases de données Aurora dans une Région AWS individuelle. Avec une base de données Aurora Global Database, vous pouvez planifier et effectuer des reprises plus rapides après des incidents régionaux imprévus ou des pannes complètes de services.

Vous pouvez consulter les directives et procédures suivantes pour planifier, tester et mettre en œuvre votre stratégie BCDR à l'aide de la fonctionnalité de base de données Aurora Global Database.

Rubriques

- Planification de la continuité des activités et de la reprise après sinistre
- Réalisation de bascules pour les bases de données Amazon Aurora Global Database
- Reprise d'une base de données Amazon Aurora globale à partir d'une panne non planifiée
- Gestion des RPO pour les bases de données globales basées sur Aurora PostgreSQL–
- Résilience entre régions pour les clusters secondaires de Global Database

Planification de la continuité des activités et de la reprise après sinistre

Pour planifier votre stratégie de continuité des activités et de reprise après sinistre, il est utile de comprendre la terminologie suivante et le lien entre ces termes et les fonctionnalités de la base de données Aurora Global Database.

La reprise après sinistre est généralement axée sur les deux objectifs stratégiques suivants :

- Objectif de délai de reprise (RTO) : temps nécessaire à un système pour revenir à un état de fonctionnement après un sinistre ou une interruption de services. En d'autres termes, le RTO mesure la durée d'indisponibilité. Pour une base de données Aurora globale, le RTO peut être de l'ordre de quelques minutes.
- Objectif de point de reprise (RPO) : quantité de données pouvant être perdue (mesurée dans le temps) après un sinistre ou une interruption de services. Cette perte de données est généralement due à un retard de réplication asynchrone. Pour une base de données Aurora globale, le RPO est généralement mesuré en secondes. Avec une base de données globale basée sur Aurora PostgreSQL–, vous pouvez utiliser le paramètre `rds.global_db_rpo` pour définir et suivre la limite supérieure du RPO, mais cela peut affecter le traitement des transactions sur le nœud d'écriture du cluster principal. Pour plus d'informations, consultez Gestion des RPO pour les bases de données globales basées sur Aurora PostgreSQL–.

L'exécution d'une bascule ou d'un basculement avec la base de données Aurora Global Database implique de promouvoir un cluster de bases de données secondaire en tant que cluster de bases de données principal. Le terme « panne régionale » est couramment utilisé pour décrire divers scénarios de défaillance. Le pire des scénarios pourrait être une panne généralisée due à un événement catastrophique qui toucherait des centaines de kilomètres carrés. Toutefois, la plupart des pannes sont beaucoup plus localisées et ne concernent qu'un petit sous-ensemble de services cloud ou de systèmes clients. Tenez compte de toute l'étendue de la panne pour vous assurer que le basculement entre régions est la bonne solution et pour choisir la méthode de basculement adaptée à la situation. L'utilisation de l'approche de basculement ou de bascule dépend du scénario de panne spécifique :

- Basculement : utilisez cette approche pour récupérer après une panne imprévue. Avec cette approche, vous effectuez un basculement entre régions vers l'un des clusters de bases de données secondaires de votre base de données globale Aurora. Le RPO pour cette approche est généralement une valeur non nulle mesurée en secondes. L'ampleur de la perte de données dépend du retard de réplication de la base de données globale Aurora entre les Régions AWS au

moment de l'échec. Pour en savoir plus, consultez Reprise d'une base de données Amazon Aurora globale à partir d'une panne non planifiée.

- Bascule : cette opération était précédemment appelée « basculement planifié géré ». Adoptez cette approche pour les scénarios contrôlés, tels que la maintenance opérationnelle et d'autres procédures opérationnelles planifiées, où tous les clusters Aurora et les autres services avec lesquels ils interagissent sont en bon état. Étant donné que cette fonction synchronise les clusters de bases de données secondaires avec le cluster principal avant toute modification, le RPO est égal à 0 (aucune donnée perdue). Pour en savoir plus, consultez Réalisation de bascules pour les bases de données Amazon Aurora Global Database.

> **Note**
>
> Si vous souhaitez effectuer une opération de bascule ou de basculement vers un cluster de bases de données Aurora secondaire sans périphériques, vous devez d'abord y ajouter une instance de base de données. Pour plus d'informations sur les clusters de bases de données sans tête, consultez Création d'un cluster de bases de données Aurora sans tête dans une région secondaire.

Réalisation de bascules pour les bases de données Amazon Aurora Global Database

> **Note**
>
> Les bascules étaient auparavant appelées basculements planifiés gérés.

En utilisant les bascules, vous pouvez modifier de façon routinière la région de votre cluster principal. Cette approche est destinée aux scénarios contrôlés tels que la maintenance opérationnelle et d'autres procédures opérationnelles planifiées.

Il existe trois cas d'utilisation courants pour l'utilisation des bascules.

- Pour les exigences de « rotation régionale » imposées à des secteurs spécifiques. Par exemple, la réglementation des services financiers peut exiger que les systèmes de niveau 0 passent à une autre région pendant plusieurs mois afin de garantir que les procédures de reprise après sinistre sont régulièrement mises à l'épreuve.

- Pour les applications 24 h/24 multirégionales. Par exemple, une entreprise peut souhaiter fournir une latence d'écriture plus faible dans différentes régions en fonction des heures d'ouverture dans différents fuseaux horaires.
- En tant que méthode sans perte de données pour revenir à la région principale d'origine après un basculement.

> **Note**
>
> Les bascules sont conçues pour être utilisées sur une base de données globale Aurora où tous les clusters Aurora et les autres services avec lesquels ils interagissent sont en bon état. Pour récupérer après une panne imprévue, suivez la procédure appropriée dans Reprise d'une base de données Amazon Aurora globale à partir d'une panne non planifiée.
> Vous ne pouvez effectuer une bascule gérée entre régions avec une base de données Aurora Global Database que si les clusters de bases de données principal et secondaires possèdent les mêmes versions de moteur majeures et mineures. Selon le moteur et les versions du moteur, les niveaux de correctifs doivent parfois être identiques ou peuvent être différents. Pour obtenir la liste des moteurs et des versions de moteur qui autorisent ces opérations entre les clusters principaux et secondaires avec différents niveaux de correctif, consultez Compatibilité des niveaux de correctif pour les bascules ou basculements gérés entre régions. Avant de commencer la bascule, vérifiez les versions du moteur de votre cluster global pour vous assurer qu'elles prennent en charge la bascule gérée entre régions, et mettez-les à niveau si nécessaire.

Lors d'une bascule, Aurora fait du cluster de la région secondaire de votre choix le cluster principal. Le mécanisme de bascule conserve la topologie de réplication existante de votre base de données globale : elle possède toujours le même nombre de clusters Aurora dans les mêmes régions. Avant de démarrer le processus de bascule, Aurora attend que tous les clusters des régions secondaires soient entièrement synchronisés avec le cluster de la région principale. Ensuite, le cluster de bases de données de la région principale devient accessible en lecture seule. Le cluster secondaire choisi promeut l'un de ses nœuds en lecture seule à l'état d'enregistreur complet, ce qui permet au cluster d'endosser le rôle de cluster principal. Comme tous les clusters secondaires ont été synchronisés avec le cluster principal au début du processus, le nouveau cluster principal poursuit les opérations de la base de données Aurora globale sans perdre de données. Votre base de données est indisponible pendant une courte période durant laquelle les clusters principaux et secondaires sélectionnés endossent leurs nouveaux rôles.

> **Note**
>
> Pour gérer les emplacements de réplication pour Aurora PostgreSQL après avoir effectué une bascule, consultez Gestion des emplacements logiques pour Aurora PostgreSQL .

Pour optimiser la disponibilité des applications, nous vous recommandons d'effectuer les opérations suivantes avant d'utiliser cette fonctionnalité :

- Effectuez cette opération pendant les heures creuses ou à tout autre moment où les écritures sur le cluster de bases de données principal sont minimales.
- Vérifiez les temps de retard pour tous les clusters de bases de données Aurora secondaires de la base de données Aurora globale. Pour toutes les bases de données globales basées sur Aurora PostgreSQL et pour les bases de données globales basées sur Aurora MySQL à partir des versions de moteur 3.04.0 et ultérieures, ou 2.12.0 et ultérieures, utilisez Amazon CloudWatch pour consulter la métrique `AuroraGlobalDBRPOLag` pour tous les clusters de bases de données secondaires. Pour les versions mineures inférieures des bases de données globales basées sur Aurora MySQL, consultez la métrique `AuroraGlobalDBReplicationLag` à la place. Ces métriques indiquent le retard (en millisecondes) de la réplication vers un cluster secondaire par rapport au cluster de bases de données principal. Cette valeur est directement proportionnelle au temps nécessaire à Aurora pour terminer la bascule. Par conséquent, plus la valeur de retard est élevée, plus la bascule prendra de temps. Lorsque vous examinez ces métriques, faites-le à partir du cluster principal actuel.

  Pour plus d'informations sur les métriques CloudWatch pour Aurora, consultez Métriques de niveau cluster pour Amazon Aurora.
- Le cluster de bases de données secondaire promu lors d'une bascule peut avoir des paramètres de configuration différents de ceux de l'ancien cluster de bases de données principal. Nous vous recommandons d'assurer la cohérence des types de paramètres de configuration suivants dans tous les clusters de vos clusters de bases de données globales Aurora. Cela permet de minimiser les problèmes de performances, les incompatibilités de charge de travail et d'autres comportements anormaux après la bascule.
  - Configurez le groupe de paramètres du cluster de bases de données Aurora pour le nouveau cluster principal, si nécessaire : lorsque vous promouvez un cluster de bases de données secondaire pour endosser le rôle de cluster principal, le groupe de paramètres du cluster secondaire peut être configuré différemment de celui du cluster principal. Si c'est le cas, modifiez le groupe de paramètres du cluster de bases de données secondaire promu afin qu'il soit

conforme aux paramètres de votre cluster principal. Pour savoir comment procéder, consultez Modification des paramètres d'une base de données Aurora globale.

- Configurer les outils et les options de surveillance tels que Amazon CloudWatch Events et les alarmes – Configurez le cluster de bases de données promu avec la même capacité de journalisation, les mêmes alarmes, etc. que nécessaire pour la base de données globale. Comme pour les groupes de paramètres, la configuration de ces fonctionnalités n'est pas héritée du cluster principal durant le processus de bascule. Certaines métriques CloudWatch, telles que le délai de réplication, ne sont disponibles que pour les régions secondaires. Ainsi, une bascule modifie la façon d'afficher ces métriques et de définir des alarmes associées, et peut nécessiter d'apporter des modifications à des tableaux de bord prédéfinis. Pour plus d'informations sur les clusters de bases de données Aurora et la surveillance, consultez Surveillance des métriques Amazon Aurora avec Amazon CloudWatch.
- Configurez des intégrations avec d'autres services AWS : si votre base de données globale Aurora s'intègre avec des services AWS, tels que AWS Secrets Manager, Gestion des identités et des accès AWS, Amazon S3 et AWS Lambda, veillez à configurer vos intégrations avec ces services selon vos besoins. Pour plus d'informations sur l'intégration de bases de données Aurora globales avec IAM, Simple Storage Service (Amazon S3) et Lambda, consultez Utilisation des bases de données Amazon Aurora Global Database avec d'autres services AWS. Pour en savoir plus sur Secrets Manager, consultez Comment automatiser la réplication de secrets dans AWS Secrets Manager entre Régions AWS.

Si vous utilisez le point de terminaison d'enregistreur Aurora Global Database, il n'est pas nécessaire de changer les paramètres de connexion dans votre application. Vérifiez que les modifications DNS se sont propagées et que vous pouvez vous connecter et effectuer des opérations d'écriture sur le nouveau cluster principal. Vous pouvez ensuite reprendre l'exécution complète de votre application.

Supposons que les connexions de votre application utilisent le point de terminaison de l'ancien cluster principal, au lieu du point de terminaison d'enregistreur global. Dans ce cas, assurez-vous de modifier les paramètres de connexion de votre application pour que le point de terminaison du nouveau cluster principal soit utilisé. Si vous avez accepté les noms fournis lors de la création de la base de données Aurora globale, vous pouvez modifier le point de terminaison en supprimant la chaîne `-ro` du point de terminaison du cluster promu dans votre application. Par exemple, le point de terminaison du cluster secondaire `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` devient `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` lorsque ce cluster est promu cluster principal.

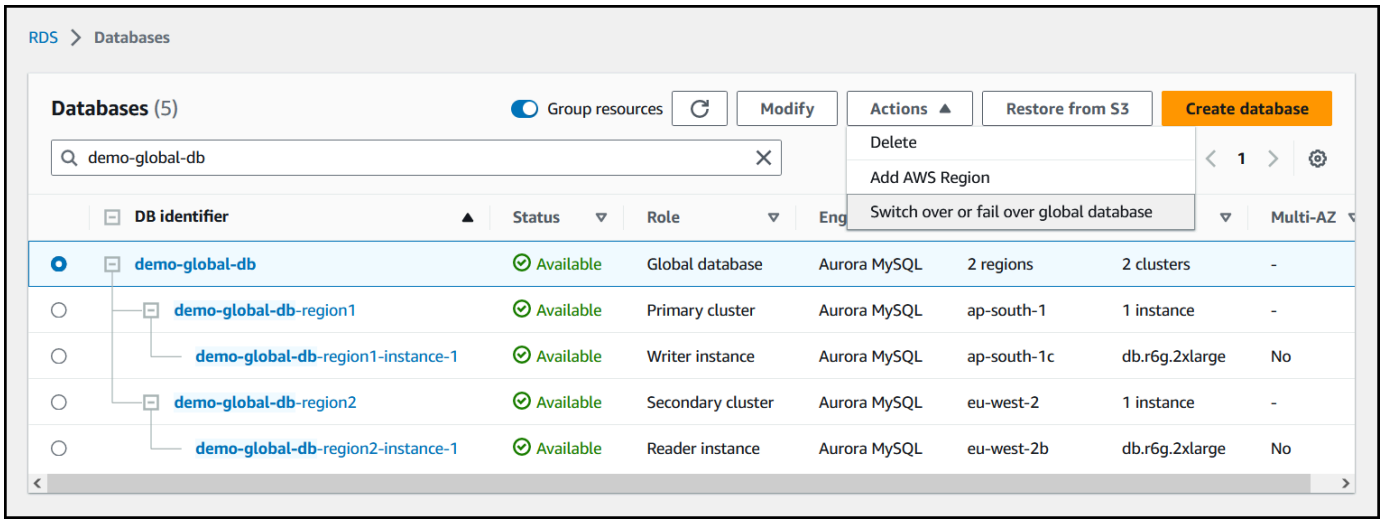
Si vous utilisez un proxy RDS, assurez-vous de rediriger les opérations en écriture de votre application vers le point de terminaison en lecture/écriture approprié du proxy qui est associé au nouveau cluster principal. Ce point de terminaison du proxy peut être le point de terminaison par défaut ou un point de terminaison en lecture/écriture personnalisé. Pour plus d'informations, consultez Fonctionnement des points de terminaison du proxy RDS avec les bases de données globales.

Vous pouvez effectuer une opération de bascule de votre base de données Aurora Global Database à l'aide de la AWS Management Console, de l'interface AWS CLI ou de l'API RDS.

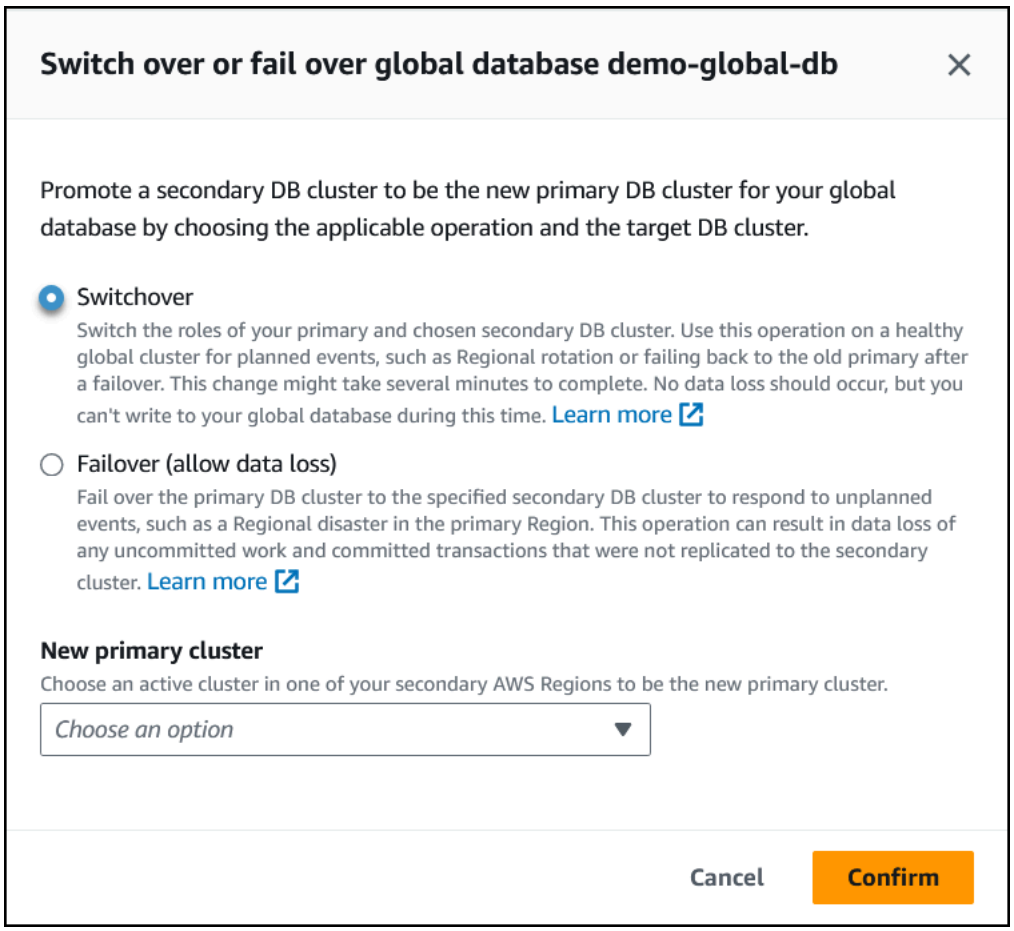
Console

Pour effectuer la bascule sur votre base de données globale Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données et recherchez la base de données globale Aurora pour laquelle vous souhaitez effectuer une opération de bascule.
3. Choisissez Passer ou basculer sur une base de données globale dans le menu Actions.

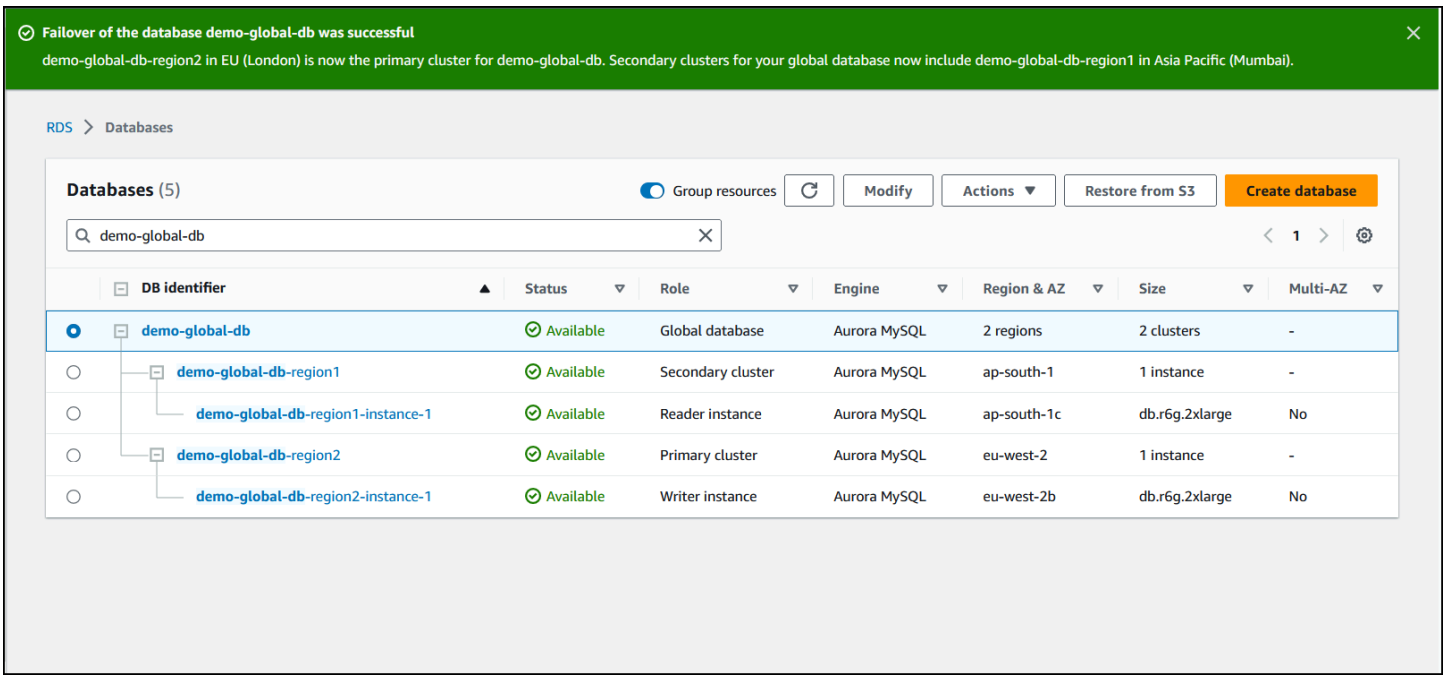


4. Choisissez Bascule.



5. Pour Nouveau cluster principal, choisissez un cluster actif dans l'une de vos Régions AWS secondaires comme nouveau cluster principal.
6. Choisissez Confirmer.

Lorsque la bascule se termine, vous pouvez voir les clusters de bases de données Aurora et leurs rôles actuels dans la liste Bases de données, comme illustré dans l'image suivante.



AWS CLI

Pour effectuer la bascule sur une base de données globale Aurora

Utilisez la commande CLI `switchover-global-cluster` pour effectuer une bascule pour Aurora Global Database. Avec la commande, passez les valeurs pour les paramètres suivants.

- `--region` : spécifiez la Région AWS dans laquelle s'exécute le cluster de bases de données principal de la base de données Aurora globale.
- `--global-cluster-identifier` – Spécifiez le nom de votre base de données Aurora globale.
- `--target-db-cluster-identifier` – Spécifiez l'Amazon Resource Name (ARN) du cluster de bases de données Aurora que vous souhaitez promouvoir comme cluster principal pour la base de données Aurora globale.

Pour Linux, macOS ou Unix :

```
aws rds --region region_of_primary \
   switchover-global-cluster --global-cluster-identifier global_database_id \
  --target-db-cluster-identifier arn_of_secondary_to_promote
```

Pour Windows :

```
aws rds --region region_of_primary ^
```

```
  switchover-global-cluster --global-cluster-identifier global_database_id ^
--target-db-cluster-identifier arn_of_secondary_to_promote
```

API RDS

Pour effectuer une opération de bascule pour une base de données Aurora Global Database, exécutez l'opération d'API SwitchoverGlobalCluster.

Reprise d'une base de données Amazon Aurora globale à partir d'une panne non planifiée

Dans de rares cas, votre base de données globale Aurora peut subir une panne inattendue dans sa Région AWS principale. Si cela se produit, votre cluster de bases de données Aurora principal et son nœud d'enregistreur ne sont pas disponibles, et la réplication entre les clusters de bases de données principal et secondaires s'interrompt. Pour minimiser la durée d'indisponibilité (RTO) et la perte de données (RPO), vous pouvez travailler rapidement pour effectuer un basculement entre régions.

La base de données Aurora Global Database offre deux méthodes de basculement que vous pouvez utiliser en cas de reprise après sinistre :

- Basculement géré : cette méthode est recommandée pour la reprise après sinistre. Lorsque vous utilisez cette méthode, Aurora réintègre automatiquement l'ancienne région principale dans la base de données globale en tant que région secondaire lorsqu'elle redevient disponible. Ainsi, la topologie d'origine de votre cluster global est conservée. Pour apprendre à utiliser cette méthode, consultez Réalisation de basculements gérés pour les bases de données globales Aurora.
- Basculement manuel : cette méthode alternative peut être utilisée quand le basculement géré n'est pas une option, par exemple quand vos régions principale et secondaire exécutent des versions de moteur incompatibles. Pour apprendre à utiliser cette méthode, consultez Réalisation de basculements manuels pour les bases de données globales Aurora.

> **Important**
>
> Les deux méthodes de basculement peuvent entraîner la perte de données de transaction d'écriture qui n'ont pas été répliquées dans la région secondaire choisie avant que le basculement se produise. Toutefois, le processus de récupération qui fait la promotion d'une instance de base de données sur le cluster de bases de données secondaire choisi comme instance de base de données principale d'enregistreur garantit que les données sont dans

un état de cohérence transactionnelle. Les basculements sont également susceptibles de provoquer des problèmes d'incohérence des données, ou split-brain.

Réalisation de basculements gérés pour les bases de données globales Aurora

Cette approche vise à assurer la continuité des activités dans le cas d'une véritable catastrophe régionale ou interruption complète du niveau de service.

Au cours d'un basculement géré, le cluster secondaire de la région secondaire de votre choix devient le nouveau cluster principal. Le cluster secondaire choisi promeut l'un de ses nœuds en lecture seule au statut d'enregistreur complet. Cette étape permet au cluster d'endosser le rôle de cluster principal. Votre base de données est indisponible pendant une courte période pendant que ce cluster endosse son nouveau rôle. Dès que cette ancienne région principale sera saine et à nouveau disponible, Aurora l'ajoutera automatiquement au cluster global en tant que région secondaire. Ainsi, la topologie de réplication existante de votre base de données globale Aurora sera conservée.

> **Note**
>
> Pour gérer les emplacements de réplication pour Aurora PostgreSQL après avoir effectué un basculement, consultez Gestion des emplacements logiques pour Aurora PostgreSQL .

> **Note**
>
> Vous ne pouvez effectuer un basculement géré entre régions avec une base de données Aurora Global Database que si les clusters de bases de données principal et secondaires possèdent les mêmes versions de moteur majeures et mineures. Selon le moteur et les versions du moteur, les niveaux de correctifs doivent parfois être identiques ou peuvent être différents. Pour obtenir la liste des moteurs et des versions de moteur qui autorisent ces opérations entre les clusters principaux et secondaires avec différents niveaux de correctif, consultez Compatibilité des niveaux de correctif pour les bascules ou basculements gérés entre régions. Avant de commencer la bascule, vérifiez les versions du moteur de votre cluster global pour vous assurer qu'elles prennent en charge la bascule gérée entre régions, et mettez-les à niveau si nécessaire. Si les versions de votre moteur nécessitent des niveaux de correctifs identiques, mais exécutent des niveaux de correctif différents, vous pouvez

effectuer le basculement manuellement en suivant les étapes décrites dans Réalisation de basculements manuels pour les bases de données globales Aurora.

Le basculement géré n'attend pas que les données soient synchronisées entre la région secondaire choisie et la région principale actuelle. Dans la mesure où la base de données Aurora Global Database réplique les données de manière asynchrone, il est possible que toutes les transactions ne soient pas répliquées dans la région AWS secondaire choisie avant qu'elle ne soit promue pour accepter toutes les fonctionnalités de lecture/écriture.

Pour garantir la cohérence des données, Aurora crée un volume de stockage pour l'ancienne région principale après sa restauration. Avant de créer le volume de stockage dans la région AWS principale, Aurora essaie d'effectuer un instantané de l'ancien volume de stockage au point de défaillance. Vous pourrez ainsi restaurer l'instantané et récupérer les données manquantes. Si cette opération aboutit, Aurora place cet instantané nommé `rds:unplanned-global-failover-`*`name-of-old-primary-DB-cluster`*`-`*`timestamp`* dans la section des instantanés de la AWS Management Console. Vous pouvez également utiliser la commande `describe-db-cluster-snapshots` d'AWS CLI ou l'opération d'API `DescribeDBClusterSnapshots` pour voir les détails de l'instantané.

Lorsque vous lancez un basculement géré, Aurora tente également d'arrêter le trafic d'écriture via la couche de stockage Aurora hautement disponible. C'est ce que l'on appelle le « clôturage d'écriture ». Si le processus aboutit, Aurora émet un événement RDS vous indiquant que les écritures ont été arrêtées. Dans le cas peu probable de plusieurs défaillances de zone de disponibilité dans une région, il est possible que le processus de clôturage d'écriture n'aboutisse pas à temps. Dans ce cas, Aurora émet un événement RDS vous informant que le délai imparti pour arrêter les écritures a expiré. Si l'ancien cluster principal est accessible sur le réseau, Aurora y enregistre ces événements. Dans le cas contraire, Aurora les enregistre sur le nouveau cluster principal. Pour en savoir plus sur ces événements, consultez Événements de cluster de bases de données. Comme le clôturage d'écriture est une tentative exécutée dans la mesure du possible, il est possible que les écritures soient momentanément acceptées dans l'ancienne région principale, ce qui provoquerait des problèmes d'incohérence des données, ou split-brain.

Nous vous recommandons d'effectuer les tâches suivantes avant de procéder à un basculement avec une base de données Aurora Global Database. Cela permet de minimiser le risque d'incohérence des données ou de récupération de données non répliquées à partir de l'instantané de l'ancien cluster principal.

- Pour empêcher l'envoi d'écritures vers le cluster principal de la base de données Aurora Global Database, déconnectez les applications.
- Assurez-vous que toutes les applications qui se connectent au cluster de bases de données principal utilisent le point de terminaison d'enregistreur global. La valeur de ce point de terminaison reste identique même lorsqu'une nouvelle région devient le cluster principal en raison d'une opération de bascule ou de basculement. Aurora met en œuvre des protections supplémentaires afin de minimiser le risque de perte de données pour les opérations d'écriture soumises via le point de terminaison global. Pour plus d'informations sur les points de terminaison d'enregistreur global, consultez Connexion à Amazon Aurora Global Database.
- Si vous utilisez le point de terminaison d'enregistreur global et que votre application ou vos couches réseau mettent en cache les valeurs DNS, réduisez la durée de vie (TTL) du cache du DNS pour qu'il corresponde à une faible valeur, telle que 5 secondes. Ainsi, votre application enregistrera rapidement les modifications du DNS auprès du point de terminaison d'enregistreur global. Bien qu'Aurora tente de bloquer les écritures dans l'ancienne région principale, le succès de l'action n'est pas garanti. La réduction de la durée de vie du cache du DNS limite davantage le risque d'incohérence de données. Vous pouvez également rechercher l'événement RDS qui vous indique quand Aurora a observé des modifications du DNS pour le point de terminaison d'enregistreur global. Vous pouvez ainsi vérifier que votre application a également enregistré la modification du DNS avant de redémarrer le trafic d'écriture de votre application.
- Vérifiez les temps de retard pour tous les clusters de bases de données Aurora secondaires de la base de données Aurora globale. Le choix de la région secondaire présentant le retard de réplication minimum peut minimiser les pertes de données avec la région principale actuellement défaillante.

  Pour toutes les versions de bases de données globales basées sur Aurora PostgreSQL et pour les bases de données globales basées sur Aurora MySQL à partir des versions de moteur 3.04.0 et ultérieures, ou 2.12.0 et ultérieures, utilisez Amazon CloudWatch pour consulter la métrique `AuroraGlobalDBRPOLag` pour tous les clusters de bases de données secondaires. Pour les versions mineures inférieures des bases de données globales basées sur Aurora MySQL, consultez la métrique `AuroraGlobalDBReplicationLag` à la place. Ces métriques indiquent le retard (en millisecondes) de la réplication vers un cluster secondaire par rapport au cluster de bases de données principal.

  Pour plus d'informations sur les métriques CloudWatch pour Aurora, consultez Métriques de niveau cluster pour Amazon Aurora.

Au cours d'un basculement géré, le cluster de bases de données secondaire choisi est promu dans son nouveau rôle de cluster principal. Toutefois, il n'hérite pas des différentes options de configuration du cluster de bases de données principal. Une incompatibilité de configuration peut provoquer des problèmes de performances, des incompatibilités de charge de travail et d'autres comportements anormaux. Pour éviter de tels problèmes, nous vous recommandons de résoudre les différences entre vos clusters de bases de données Aurora globales pour les cas suivants :

- Configurer le groupe de paramètres de cluster de bases de données Aurora pour le nouveau cluster principal, si nécessaire – Vous pouvez configurer vos groupes de paramètres de cluster de bases de données Aurora indépendamment pour chaque cluster Aurora de votre base de données Aurora Global Database. Par conséquent, lorsque vous promouvez un cluster de bases de données secondaire pour endosser le rôle de cluster principal, le groupe de paramètres du cluster secondaire peut être configuré différemment de celui du cluster principal. Si c'est le cas, modifiez le groupe de paramètres du cluster de bases de données secondaire promu afin qu'il soit conforme aux paramètres de votre cluster principal. Pour savoir comment procéder, consultez Modification des paramètres d'une base de données Aurora globale.
- Configurer les outils et les options de surveillance tels que Amazon CloudWatch Events et les alarmes – Configurez le cluster de bases de données promu avec la même capacité de journalisation, les mêmes alarmes, etc. que nécessaire pour la base de données globale. Comme pour les groupes de paramètres, la configuration de ces fonctionnalités n'est pas héritée du cluster principal durant le processus de basculement. Certaines métriques CloudWatch, telles que le délai de réplication, ne sont disponibles que pour les régions secondaires. Ainsi, un basculement modifie la façon d'afficher ces métriques et de définir des alarmes sur celles-ci, et peut nécessiter d'apporter des modifications à des tableaux de bord prédéfinis. Pour plus d'informations sur la surveillance des clusters de bases de données, consultez Surveillance des métriques Amazon Aurora avec Amazon CloudWatch.
- Configurer des intégrations avec d'autres services AWS – Si votre base de données Aurora Global Database s'intègre avec des services AWS comme AWS Secrets Manager, Gestion des identités et des accès AWS, Amazon S3 et AWS Lambda, vous devez vous assurer qu'ils sont correctement configurés de sorte à être accessibles à partir des régions secondaires. Pour plus d'informations sur l'intégration de bases de données Aurora globales avec IAM, Simple Storage Service (Amazon S3) et Lambda, consultez Utilisation des bases de données Amazon Aurora Global Database avec d'autres services AWS. Pour en savoir plus sur Secrets Manager, consultez Comment automatiser la réplication de secrets dans AWS Secrets Manager entre Régions AWS.

Généralement, le cluster secondaire choisi endosse le rôle principal en quelques minutes. Dès que l'instance de base de données d'enregistreur de la nouvelle région principale est disponible, vous pouvez y connecter vos applications et reprendre vos charges de travail. Une fois qu'Aurora a promu le nouveau cluster principal, il reconstruit automatiquement tous les clusters de régions secondaires supplémentaires.

Comme les bases de données globales Aurora utilisent la réplication asynchrone, le retard de réplication dans chaque région secondaire peut varier. Aurora reconstruit ces régions secondaires pour qu'elles disposent exactement des mêmes données ponctuelles que le nouveau cluster de la région principale. La durée de la tâche de reconstruction complète peut prendre de quelques minutes à plusieurs heures, selon la taille du volume de stockage et la distance entre les régions. Lorsque les clusters des régions secondaires ont fini de se reconstruire à partir de la nouvelle région principale, ils sont disponibles pour un accès en lecture.

Dès que le nouvel enregistreur principal est promu et disponible, le cluster de la nouvelle région principale peut gérer les opérations de lecture et d'écriture pour la base de données globale Aurora.

Si vous utilisez le point de terminaison global, il n'est pas nécessaire de changer les paramètres de connexion dans votre application. Vérifiez que les modifications DNS se sont propagées et que vous pouvez vous connecter et effectuer des opérations d'écriture sur le nouveau cluster principal. Vous pouvez ensuite reprendre l'exécution complète de votre application.

Si vous n'utilisez pas le point de terminaison global, assurez-vous de modifier le point de terminaison de votre application afin que le point de terminaison du cluster de bases de données principal récemment promu soit utilisé. Si vous avez accepté les noms fournis lors de la création de la base de données Aurora globale, vous pouvez modifier le point de terminaison en supprimant la chaîne `-ro` du point de terminaison du cluster promu dans votre application.

Par exemple, le point de terminaison du cluster secondaire `my-global.cluster-`**`ro`**`-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` devient `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` lorsque ce cluster est promu cluster principal.

Si vous utilisez un proxy RDS, assurez-vous de rediriger les opérations en écriture de votre application vers le point de terminaison en lecture/écriture approprié du proxy qui est associé au nouveau cluster principal. Ce point de terminaison du proxy peut être le point de terminaison par défaut ou un point de terminaison en lecture/écriture personnalisé. Pour plus d'informations, consultez Fonctionnement des points de terminaison du proxy RDS avec les bases de données globales.

Pour restaurer la topologie d'origine du cluster de bases de données global, Aurora surveille la disponibilité de l'ancienne région principale. Dès que cette région est saine et à nouveau disponible, Aurora l'ajoute automatiquement au cluster global en tant que région secondaire. Avant de créer le nouveau volume de stockage dans l'ancienne région principale, Aurora essaie de prendre un instantané de l'ancien volume de stockage au point de défaillance. Il le fait pour que vous puissiez l'utiliser pour récupérer les données manquantes. Si cette opération aboutit, Aurora crée un instantané nommé `rds:unplanned-global-failover-`*`name-of-old-primary-DB-cluster`*`-`*`timestamp`*. Vous trouverez cet instantané dans la section Instantanés de la AWS Management Console. Vous pouvez également voir cet instantané répertorié dans les informations renvoyées par l'opération d'API DescribeDBClusterSnapshots.

> **Note**
>
> L'instantané de l'ancien volume de stockage est un instantané du système soumis à la période de conservation des sauvegardes configurée sur l'ancien cluster principal. Pour conserver cet instantané en dehors de la période de conservation, vous pouvez le copier pour l'enregistrer en tant qu'instantané manuel. Pour en savoir plus sur la copie des instantanés, y compris la tarification, consultez Copie d'un instantané de cluster de bases de données.

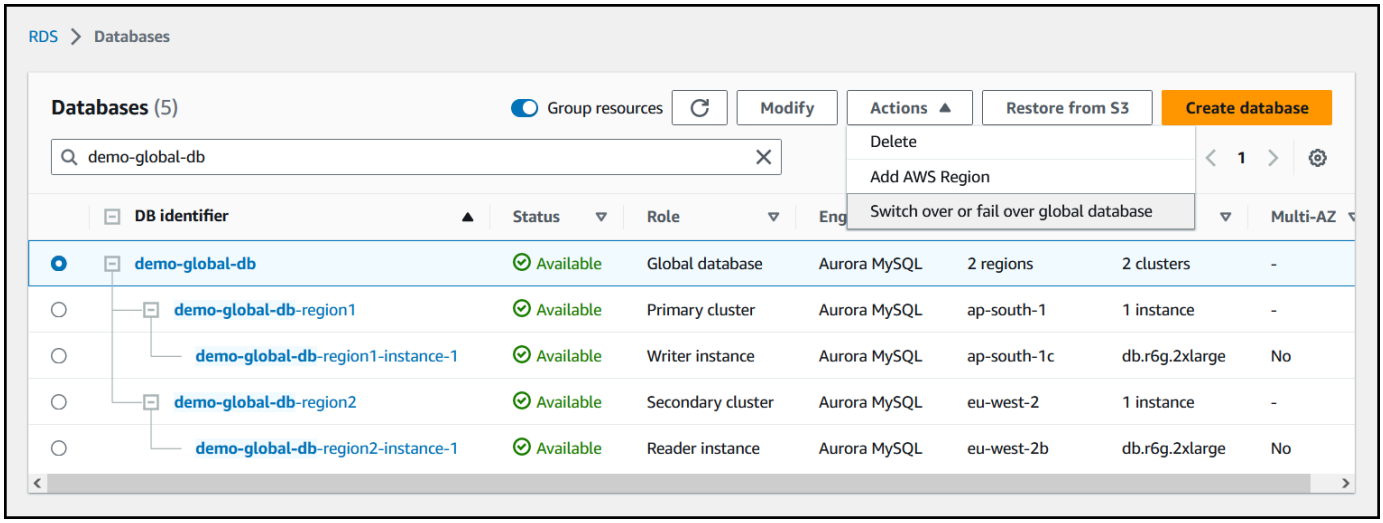
Une fois la topologie d'origine restaurée, vous pouvez rétablir votre base de données globale dans la région principale d'origine en effectuant une opération de bascule au moment qui convient le mieux à votre activité et à votre charge de travail. Pour ce faire, suivez les étapes de Réalisation de bascules pour les bases de données Amazon Aurora Global Database.

Vous pouvez effectuer une opération de basculement de votre base de données Aurora Global Database à l'aide de la AWS Management Console, de l'interface AWS CLI ou de l'API RDS.

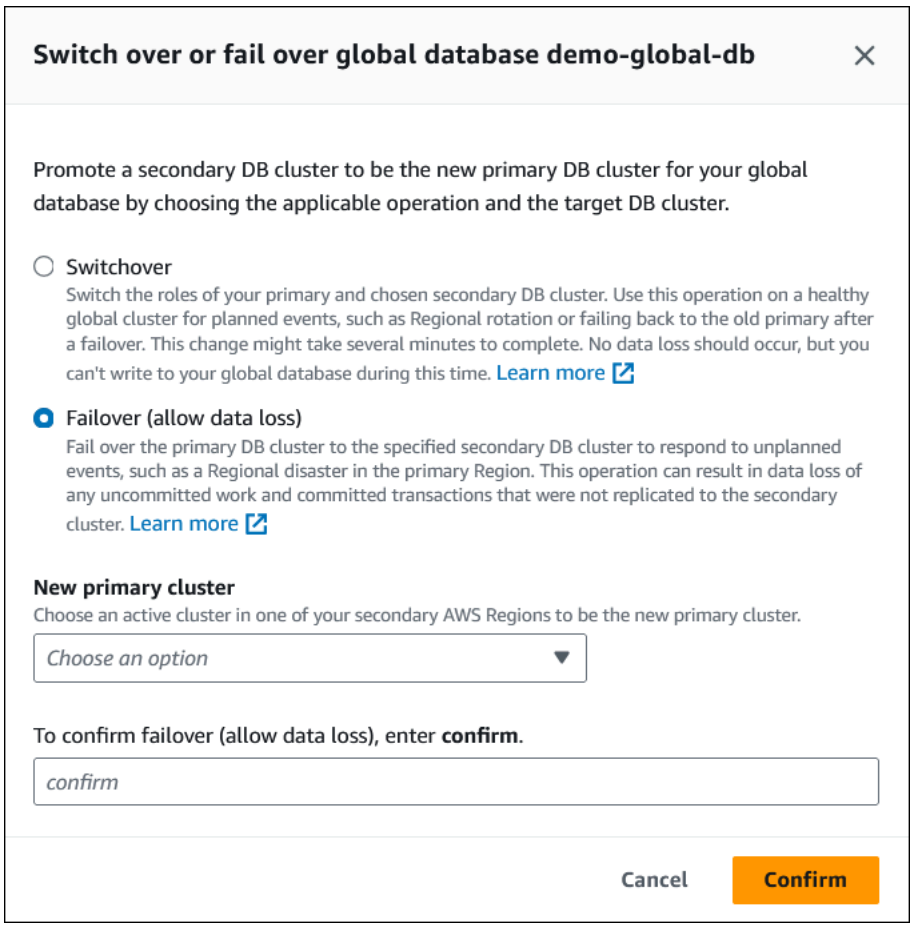
Console

Pour effectuer le basculement géré sur votre base de données globale Aurora

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données et recherchez la base de données globale Aurora que vous souhaitez basculer.
3. Choisissez Passer ou basculer sur une base de données globale dans le menu Actions.

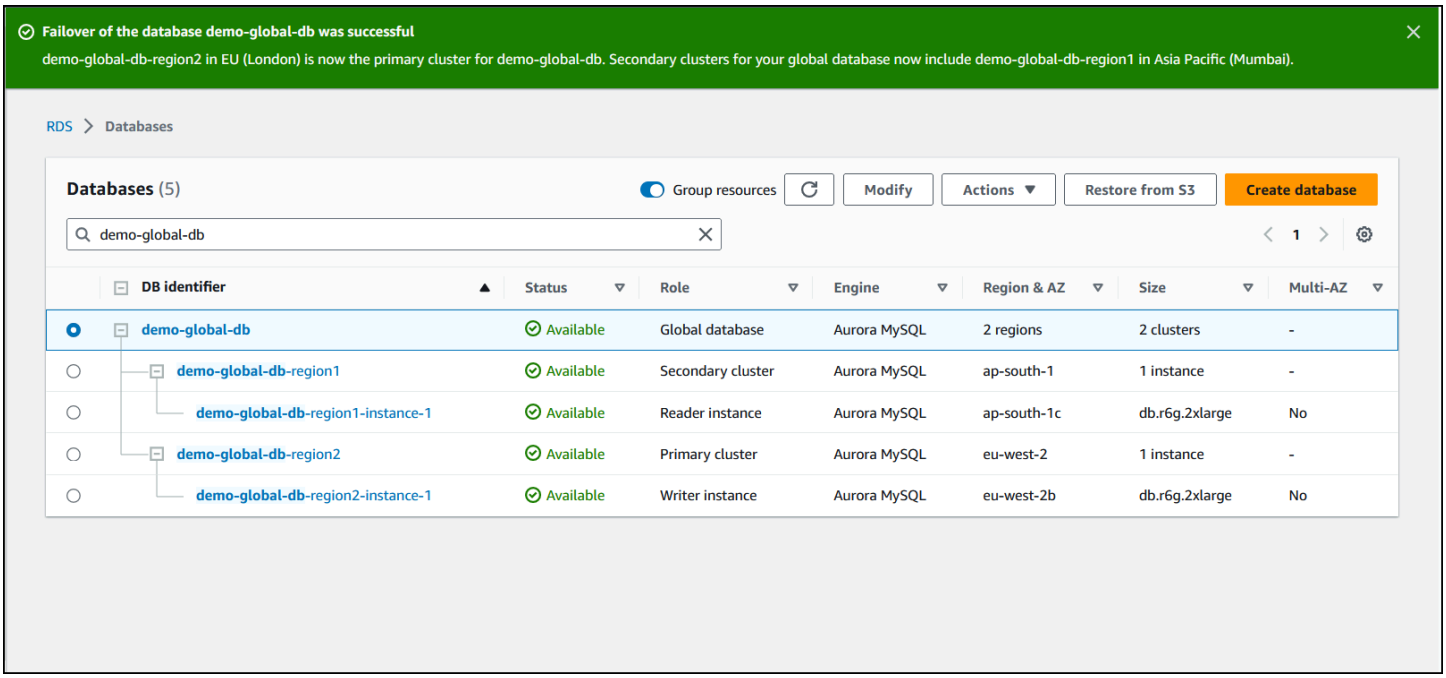


4. Choisissez Basculement (autoriser la perte de données).



5. Pour Nouveau cluster principal, choisissez un cluster actif dans l'une de vos Régions AWS secondaires comme nouveau cluster principal.
6. Saisissez **`confirm`**, puis choisissez Confirmer.

Lorsque le basculement se termine, vous pouvez consulter les clusters de bases de données Aurora et leur état actuel dans la liste Bases de données, comme illustré ci-dessous.



AWS CLI

Pour effectuer le basculement géré sur une base de données globale Aurora

Utilisez la commande CLI `failover-global-cluster` pour basculer vers votre base de données Aurora Global Database. Avec la commande, passez les valeurs pour les paramètres suivants.

- `--region` : spécifiez la Région AWS où le cluster de bases de données secondaire que vous souhaitez utiliser comme nouveau cluster principal pour la base de données globale Aurora est en cours d'exécution.
- `--global-cluster-identifier` – Spécifiez le nom de votre base de données Aurora globale.
- `--target-db-cluster-identifier` : spécifiez l'Amazon Resource Name (ARN) du cluster de bases de données Aurora que vous souhaitez promouvoir comme nouveau cluster principal pour la base de données globale Aurora.
- `--allow-data-loss` : faites-en explicitement une opération de basculement à la place d'une opération de bascule. Une opération de basculement peut entraîner une certaine perte de données si les composants de réplication asynchrone n'ont pas terminé d'envoyer toutes les données répliquées vers la région secondaire.

Pour Linux, macOS ou Unix :

```
aws rds --region region_of_selected_secondary \
   failover-global-cluster --global-cluster-identifier global_database_id \
   --target-db-cluster-identifier arn_of_secondary_to_promote \
   --allow-data-loss
```

Pour Windows :

```
aws rds --region region_of_selected_secondary ^
   failover-global-cluster --global-cluster-identifier global_database_id ^
   --target-db-cluster-identifier arn_of_secondary_to_promote ^
   --allow-data-loss
```

API RDS

Pour effectuer un basculement avec Aurora Global Database, exécutez l'opération d'API FailoverGlobalCluster.

Réalisation de basculements manuels pour les bases de données globales Aurora

Dans certains scénarios, vous ne pourrez peut-être pas utiliser le processus de basculement géré. Par exemple, si vos clusters de bases de données principal et secondaire n'exécutent pas des versions de moteur compatibles. Dans ce cas, vous pouvez suivre ce processus manuel pour effectuer un basculement vers votre région secondaire cible.

> **Tip**
>
> Nous vous recommandons de comprendre ce processus avant de l'utiliser. Ayez un plan prêt pour agir rapidement au premier signe de problème à l'échelle de la région. Vous pouvez être prêt à identifier la région secondaire présentant le moins de retard de réplication en utilisant régulièrement Amazon CloudWatch pour suivre les temps de retard des clusters secondaires. Veillez à tester votre plan pour vérifier que vos procédures sont complètes et précises, et que le personnel est formé pour effectuer un basculement de reprise après sinistre avant que le cas de figure se présente réellement.

Pour effectuer un basculement manuel vers un cluster secondaire après une panne imprévue dans la région principale

1. Arrêtez l'émission d'instructions DML et toute autre opération d'écriture vers le cluster de bases de données Aurora principal dans la Région AWS concernée par l'interruption.

2. Identifiez un cluster de bases de données Aurora à partir d'une Région AWS secondaire à utiliser comme nouveau cluster de bases de données principal. Si vous avez deux Régions AWS secondaires (ou plus) dans votre base de données globale Aurora, choisissez le cluster secondaire avec le retard de réplication minimal.
3. Détachez le cluster de bases de données secondaire choisi de la base de données Aurora globale.

   La suppression d'un cluster de bases de données secondaire d'une base de données Aurora globale interrompt immédiatement la réplication du cluster principal vers le cluster secondaire et le promeut en cluster de bases de données Aurora provisionné autonome avec des capacités complètes en lecture/écriture. Tous les autres clusters de bases de données Aurora secondaires associés au cluster principal dans la région concernée par la panne restent disponibles et peuvent accepter les appels de votre application. Ils consomment également des ressources. Puisque vous recréez la base de données Aurora globale, supprimez les autres clusters de bases de données secondaires avant de créer la nouvelle base de données Aurora globale dans les étapes suivantes. Cela évite les incohérences de données parmi les clusters de bases de données de la base de données Aurora globale (problèmes de split-brain).

   Afin d'obtenir les étapes détaillées du détachement, consultez Dissociation d'un cluster d'une base de données Amazon Aurora globale.
4. Reconfigurez votre application pour envoyer toutes les opérations d'écriture à ce cluster de bases de données Aurora désormais autonome à l'aide de son nouveau point de terminaison. Si vous avez accepté les noms fournis lors de la création de la base de données Aurora globale, vous pouvez modifier le point de terminaison en supprimant la chaîne `-ro` du point de terminaison du cluster promu dans votre application.

   Par exemple, le point de terminaison du cluster secondaire `my-global.cluster-ro-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` devient `my-global.cluster-aaaaaabbbbbb.us-west-1.rds.amazonaws.com` lorsque ce cluster est détaché de la base de données Aurora globale.

   Ce cluster de bases de données Aurora devient le cluster principal d'une nouvelle base de données Aurora globale lorsque vous commencez à y ajouter des Régions lors de l'étape suivante.

   Si vous utilisez un proxy RDS, assurez-vous de rediriger les opérations en écriture de votre application vers le point de terminaison en lecture/écriture approprié du proxy qui est associé au nouveau cluster principal. Ce point de terminaison du proxy peut être le point de terminaison

par défaut ou un point de terminaison en lecture/écriture personnalisé. Pour plus d'informations, consultez Fonctionnement des points de terminaison du proxy RDS avec les bases de données globales.

5. Ajoutez une Région AWS au cluster de bases de données. Lorsque vous effectuez cette opération, le processus de réplication du cluster primaire vers le cluster secondaire commence. Afin d'obtenir les étapes détaillées pour ajouter une région, consultez Ajout d'une Région AWS à une base de données Amazon Aurora globale.
6. Ajoutez d'autres Régions AWS si nécessaire pour recréer la topologie nécessaire à la prise en charge de votre application.

Assurez-vous que les écritures d'application sont envoyées au cluster de bases de données Aurora approprié avant, pendant et après l'application de ces modifications. Cela évite les incohérences de données parmi les clusters de bases de données de la base de données Aurora globale (problèmes de split-brain).

Si vous avez reconfiguré un système en réponse à une panne dans une Région AWS, vous pouvez faire de cette Région AWS le cluster principal une fois la panne résolue. Pour ce faire, ajoutez l'ancienne Région AWS à votre nouvelle base de données globale, puis utilisez le processus de bascule pour échanger son rôle. Votre base de données globale Aurora doit utiliser une version d'Aurora PostgreSQL ou d'Aurora MySQL qui prend en charge les bascules. Pour plus d'informations, consultez Réalisation de bascules pour les bases de données Amazon Aurora Global Database.

Gestion des RPO pour les bases de données globales basées sur Aurora PostgreSQL–

Avec une base de données globale basée sur Aurora PostgreSQL, vous pouvez gérer l'objectif de point de reprise (RPO) de votre base de données globale Aurora à l'aide du paramètre `rds.global_db_rpo`. Le RPO représente la quantité maximale de données pouvant être perdue en cas de panne.

Lorsque vous définissez un RPO pour votre base de données globale basée sur Aurora PostgreSQL–, Aurora surveille le temps de retard RPO de tous les clusters secondaires pour vous assurer qu'au moins un cluster secondaire reste dans la fenêtre RPO cible. Le temps de retard RPO est une autre métrique basée sur le temps.

Le RPO est utilisé lorsque votre base de données reprend ses opérations dans une nouvelle Région AWS après un basculement. Aurora évalue le RPO et les retards de RPO pour valider (ou bloquer) des transactions sur le cluster principal comme suit :

- Effectue la transaction si au moins un cluster de bases de données secondaire a un temps de retard RPO inférieur au RPO.
- Bloque la transaction si tous les clusters de bases de données secondaires ont des temps de retard RPO supérieurs au RPO. Il enregistre également l'événement dans le fichier journal PostgreSQL et émet des événements « wait » qui montrent les sessions bloquées.

En d'autres termes, si tous les clusters secondaires prennent du retard sur le RPO cible, Aurora interrompt les transactions sur le cluster principal jusqu'à ce qu'au moins un des clusters secondaires le rattrape. Les transactions interrompues sont reprises et validées dès que le retard d'au moins un cluster de bases de données secondaire devient inférieur au RPO. Par conséquent, aucune transaction ne peut être validée tant que le RPO n'est pas atteint.

Le paramètre `rds.global_db_rpo` est dynamique. Si vous décidez de ne pas bloquer toutes les transactions d'écriture jusqu'à ce que le retard diminue suffisamment, vous pouvez le réinitialiser rapidement. Dans ce cas, Aurora reconnaît et met en œuvre le changement après un court délai.

> **Important**
>
> Dans une base de données globale assortie de seulement deux régions AWS, nous recommandons de conserver la valeur par défaut du paramètre `rds.global_db_rpo` dans le groupe de paramètres de la région secondaire. Dans le cas contraire, le basculement vers cette région à la suite d'une perte de la région AWS principale pourrait conduire Aurora à interrompre les transactions. Attendez plutôt qu'Aurora ait terminé la reconstruction du cluster dans l'ancienne région AWS défaillante avant de modifier ce paramètre, ceci afin d'appliquer un RPO maximal.

Si vous définissez ce paramètre comme indiqué ci-dessous, vous pouvez également surveiller les métriques qu'il génère. Vous pouvez le faire en utilisant `psql` ou un autre outil d'interrogation du cluster de bases de données principal de la base de données Aurora globale et obtenir des informations détaillées sur les opérations de votre base de données globale basée sur Aurora PostgreSQL–. Pour savoir comment procéder, consultez Surveillance des bases de données globales basées sur Aurora PostgreSQL.

Rubriques

Configuration de l’objectif de point de reprise

Le paramètre `rds.global_db_rpo` contrôle le paramètre RPO pour une base de données PostgreSQL. Ce paramètre est pris en charge par Aurora PostgreSQL. Les valeurs valides pour `rds.global_db_rpo` sont comprises entre 20 secondes et 2 147 483 647 secondes (68 ans). Choisissez une valeur réaliste pour répondre aux besoins de votre entreprise et à votre cas d’utilisation. Par exemple, si vous voulez prévoir jusqu’à 10 minutes pour votre RPO, définissez la valeur sur 600.

Vous pouvez définir cette valeur pour votre base de données globale basée sur Aurora PostgreSQL– à l’aide de l’AWS Management Console, de l’AWS CLI ou de l’API RDS.

Console

Pour définir le RPO

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Choisissez le cluster principal de votre base de données Aurora globale et ouvrez l’onglet Configuration pour rechercher son groupe de paramètres de cluster de bases de données. Par exemple, le groupe de paramètres par défaut pour un cluster de bases de données principal exécutant Aurora PostgreSQL 11.7 est `default.aurora-postgresql11`.

   Les groupes de paramètres ne peuvent pas être modifiés directement. Au lieu de cela, procédez comme suit :

   - Créez un groupe de paramètres de cluster de bases de données personnalisé en utilisant le groupe de paramètres par défaut approprié comme point de départ. Par exemple, créez un groupe de paramètres de cluster de bases de données personnalisé basé sur le `default.aurora-postgresql11`.
   - Sur votre groupe de paramètres de bases de données personnalisé, définissez la valeur du paramètre rds.global_db_rpo pour répondre à votre cas d’utilisation. Les valeurs valides sont comprises entre 20 secondes et la valeur entière maximale 2 147 483 647 secondes (68 ans).

- Appliquez le groupe de paramètres de cluster de bases de données modifié à votre cluster de bases de données Aurora.

Pour plus d'informations, consultez Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

AWS CLI

Pour définir le paramètre `rds.global_db_rpo`, utilisez la commande de l'interface de ligne de commande modify-db-cluster-parameter-group. Dans la commande, spécifiez le nom du groupe de paramètres de votre cluster principal et les valeurs du paramètre RPO.

Dans l'exemple suivant, le RPO est défini sur 600 secondes (10 minutes) pour le groupe de paramètres du cluster de bases de données principal appelé `my_custom_global_parameter_group`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster-parameter-group \
    --db-cluster-parameter-group-name my_custom_global_parameter_group \
    --parameters
 "ParameterName=rds.global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

Pour Windows :

```
aws rds modify-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name my_custom_global_parameter_group ^
    --parameters
 "ParameterName=rds.global_db_rpo,ParameterValue=600,ApplyMethod=immediate"
```

API RDS

Pour modifier le paramètre `rds.global_db_rpo`, utilisez l'opération d'API Amazon RDS ModifyDBClusterParameterGroup.

Affichage de l'objectif de point de reprise

L'objectif de point de reprise (RPO) d'une base de données globale est stocké dans le paramètre `rds.global_db_rpo` pour chaque cluster de bases de données. Vous pouvez vous connecter au point de terminaison du cluster secondaire que vous souhaitez afficher et utiliser `psql` afin d'interroger l'instance pour cette valeur.

```
db-name=>show rds.global_db_rpo;
```

Si ce paramètre n'est pas défini, la requête renvoie le résultat suivant :

```
rds.global_db_rpo
-------------------
 -1
(1 row)
```

La réponse ci-dessous est renvoyée par un cluster de bases de données secondaire pour lequel le paramètre RPO est défini sur 1 minute.

```
rds.global_db_rpo
-------------------
 60
(1 row)
```

Vous pouvez également utiliser l'interface de ligne de commande pour obtenir des valeurs afin de savoir si `rds.global_db_rpo` est actif sur l'un des clusters de bases de données Aurora en utilisant l'interface de ligne de commande pour obtenir les valeurs de tous les paramètres `user` du cluster.

Pour Linux, macOS ou Unix :

```
aws rds describe-db-cluster-parameters \
 --db-cluster-parameter-group-name lab-test-apg-global \
 --source user
```

Pour Windows :

```
aws rds describe-db-cluster-parameters ^
 --db-cluster-parameter-group-name lab-test-apg-global *
 --source user
```

La commande renvoie une sortie similaire à celle ci-dessous pour tous les paramètres `user` différents des paramètres de cluster de bases de données `default-engine` ou `system`.

```
{
    "Parameters": [
```

```
        {
            "ParameterName": "rds.global_db_rpo",
            "ParameterValue": "60",
            "Description": "(s) Recovery point objective threshold, in seconds, that
 blocks user commits when it is violated.",
            "Source": "user",
            "ApplyType": "dynamic",
            "DataType": "integer",
            "AllowedValues": "20-2147483647",
            "IsModifiable": true,
            "ApplyMethod": "immediate",
            "SupportedEngineModes": [
                "provisioned"
            ]
        }
    ]
}
```

Pour en savoir plus sur l'affichage des paramètres du groupe de paramètres de cluster, consultez Affichage des valeurs de paramètres pour un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

Désactivation de l'objectif de point de reprise

Pour désactiver le RPO, réinitialisez le paramètre `rds.global_db_rpo`. Vous pouvez réinitialiser les paramètres à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API RDS.

Console

Pour désactiver le RPO

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Groupes de paramètres.
3. Dans la liste, choisissez votre groupe de paramètres de cluster de bases de données principal.
4. Choisissez Modifier les paramètres.
5. Sélectionnez la case en regard du paramètre rds.global_db_rpo.
6. Choisissez Réinitialiser.
7. Lorsque l'écran affiche Réinitialiser les paramètres dans le groupe de paramètres de base de données, choisissez Réinitialiser les paramètres.

Pour plus d'informations sur la réinitialisation d'un paramètre avec la console, consultez Modification de paramètres dans un groupe de paramètres de cluster de bases de données dans Amazon Aurora.

AWS CLI

Pour réinitialiser le paramètre `rds.global_db_rpo`, utilisez la commande reset-db-cluster-parameter-group.

Pour Linux, macOS ou Unix :

```
aws rds reset-db-cluster-parameter-group \
    --db-cluster-parameter-group-name global_db_cluster_parameter_group \
    --parameters "ParameterName=rds.global_db_rpo,ApplyMethod=immediate"
```

Pour Windows :

```
aws rds reset-db-cluster-parameter-group ^
    --db-cluster-parameter-group-name global_db_cluster_parameter_group ^
    --parameters "ParameterName=rds.global_db_rpo,ApplyMethod=immediate"
```

API RDS

Pour réinitialiser le paramètre `rds.global_db_rpo`, utilisez l'opération d'API Amazon RDS ResetDBClusterParameterGroup.

Résilience entre régions pour les clusters secondaires de Global Database

Les versions 16.6, 15.10, 14.15, 13.18, 12.22 ou supérieures d'Aurora PostgreSQL et les versions 3.09 ou supérieures d'Aurora MySQL contiennent des améliorations de disponibilité qui permettent aux réplicas en lecture des régions secondaires de maintenir la continuité du service en cas d'événements imprévus tels que des pannes matérielles, des perturbations du réseau entre les régions AWS, de gros volumes de transferts de données entre les clusters, etc.

Bien que les réplicas en lecture restent disponibles pour les demandes de votre application, le retard de réplication peut continuer à augmenter jusqu'à la résolution de l'événement imprévu. Vous pouvez surveiller ce retard entre les clusters principaux et secondaires à l'aide de la métrique `AuroraGlobalDBProgressLag` de CloudWatch. Pour mesurer le retard de bout en bout, y compris tout retard entre le volume du cluster et les instances de base de données du cluster secondaire, ajoutez les valeurs des métriques `AuroraGlobalDBProgressLag` et `AuroraReplicaLag` de

CloudWatch. Pour plus d'informations sur les métriques, consultez Référence des métriques pour Amazon Aurora.

La disponibilité en lecture de la base de données globale pour Aurora MySQL et les versions antérieures d'Aurora PostgreSQL peut être affectée lors de tels événements imprévus.

Pour plus d'informations sur les nouvelles fonctionnalités d'Aurora PostgreSQL 16.6, 15.10, 14.15, 13.18 et 12.22, consultez PostgreSQL 16.6 dans Notes de mise à jour d'Aurora PostgreSQL.

Pour plus d'informations sur les nouvelles fonctionnalités d'Aurora MySQL version 3.09 et ultérieures, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 3 dans Notes de mise à jour d'Aurora MySQL.

Surveillance d'une base de données Amazon Aurora Global Database

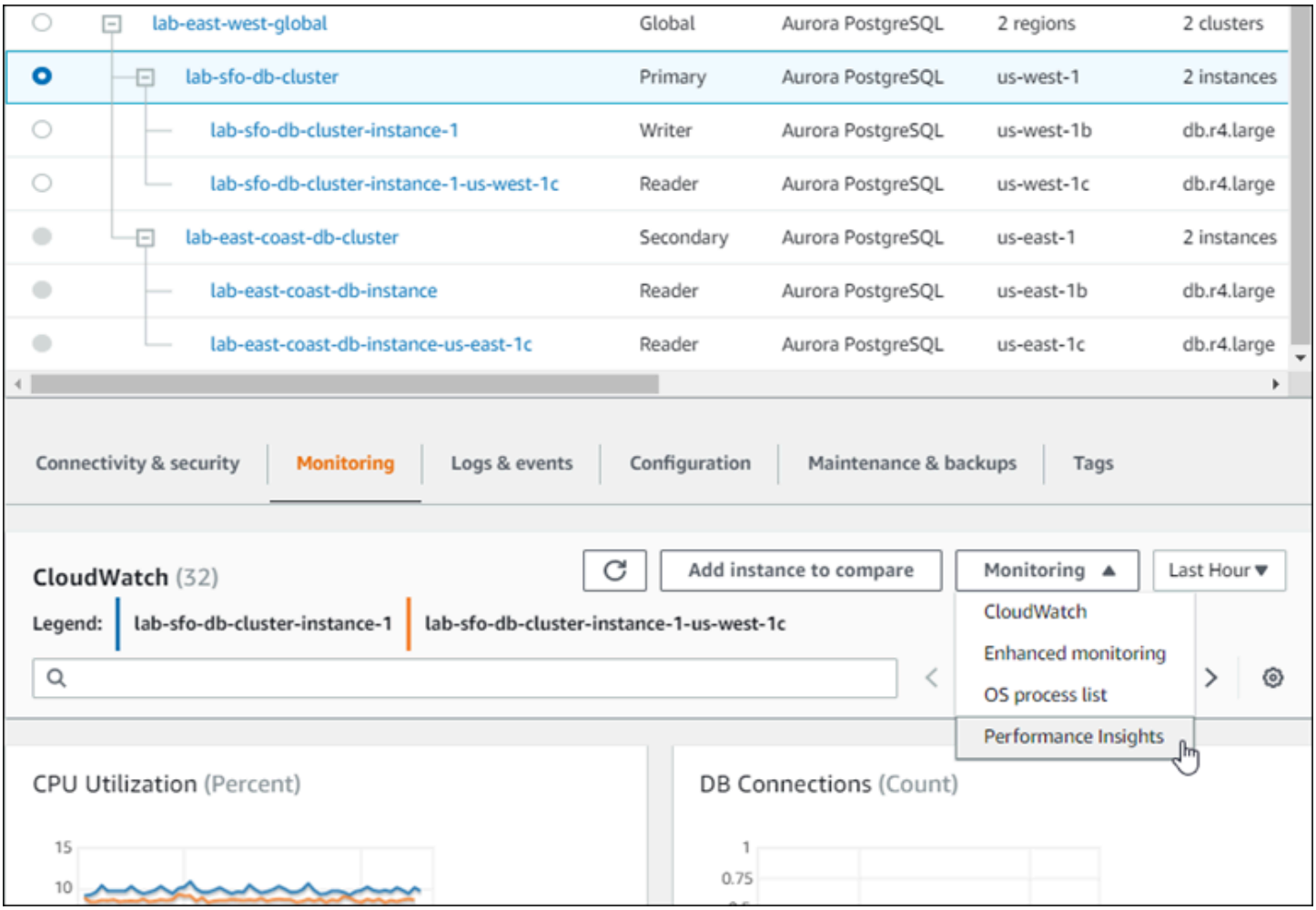
Lorsque vous créez les clusters de bases de données Aurora qui composent votre base de données Aurora globale, vous pouvez choisir de nombreuses options qui vous permettent de surveiller leurs performances. Les options disponibles sont les suivantes :

- Amazon RDS Performance Insights – Active le schéma de performance dans le moteur de base de données Aurora sous-jacent. Pour en savoir plus sur Performance Insights et les bases de données Aurora globales, consultez Surveillance d'une base de données Amazon Aurora globale avec Amazon RDS Performance Insights.
- Surveillance améliorée – Génère des métriques pour l'utilisation des processus ou des threads sur le processeur. Pour en savoir plus sur la surveillance améliorée, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.
- Amazon CloudWatch Logs – Publie les types de journal spécifiés dans CloudWatch Logs. Les journaux d'erreurs sont publiés par défaut, mais vous pouvez choisir d'autres journaux spécifiques à votre moteur de base de données Aurora.
  - Pour les clusters de bases de données Aurora basés sur Aurora MySQL, vous pouvez exporter le journal d'audit, le journal général et le journal des requêtes lentes.
  - Pour les clusters de bases de données Aurora basés sur Aurora PostgreSQL, vous pouvez exporter le journal PostgreSQL.
- Pour les bases de données globales basées sur Aurora MySQL, vous pouvez interroger des tables `information_schema` spécifiques pour vérifier l'état de votre base de données globale

Aurora et de ses instances. Pour savoir comment procéder, consultez Surveillance des bases de données globales basées sur Aurora MySQL.

- Pour les bases de données globales basées sur Aurora PostgreSQL, vous pouvez utiliser des fonctions spécifiques pour vérifier l'état de votre base de données globale Aurora et de ses instances. Pour savoir comment procéder, consultez Surveillance des bases de données globales basées sur Aurora PostgreSQL.

La capture d'écran suivante montre certaines des options disponibles sous l'onglet Surveillance d'un cluster de bases de données Aurora principal dans une base de données Aurora globale.



Pour plus d'informations, consultez Surveillance des métriques d'un cluster de bases de données Amazon Aurora.

Surveillance d’une base de données Amazon Aurora globale avec Amazon RDS Performance Insights

Vous pouvez utiliser Amazon RDS Performance Insights pour vos des bases de données Aurora globales. Vous activez cette fonctionnalité à l’échelle individuelle, pour chaque cluster de bases de données Aurora de votre base de données Aurora globale. Pour ce faire, Sélectionnez Activer Performance Insights dans la section Configuration supplémentaire de la page Créer une base de données. Vous pouvez également modifier vos clusters de bases de données Aurora pour utiliser cette fonctionnalité une fois qu’ils sont opérationnels. Vous pouvez activer ou désactiver Performance Insights pour chaque cluster qui fait partie de la base de données Aurora globale.

Les rapports créés par Performance Insights s’appliquent à chaque cluster de la base de données globale. Lorsque vous ajoutez une nouvelle Région AWS secondaire à une base de données Aurora globale qui utilise déjà Performance Insights, vous devez vous assurer d’activer cette option dans le cluster nouvellement ajouté. Elle n’hérite pas du paramètre Performance Insights de la base de données globale existante.

Vous pouvez passer d’une Régions AWS à une autre sur la page Performance Insights pour une instance de base de données qui est attachée à une base de données globale. Cependant, il se peut que vous ne voyiez pas les informations relatives aux performances immédiatement après avoir basculé d’une Régions AWS à une autre. Même si les instances de base de données peuvent avoir des noms identiques dans chaque Région AWS, l’URL Performance Insights associée est différente pour chaque instance de base de données. Après avoir basculé d’une Régions AWS à une autre, choisissez le nom de l’instance de base de données dans le panneau de navigation Performance Insights.

Pour les instances de base de données associées à une base de données globale, les facteurs affectant les performances peuvent être différents dans chaque Région AWS. Par exemple, les instances de base de données de chaque Région AWS peuvent avoir une capacité différente.

Pour plus d’informations sur l’utilisation de Performance Insights, consultez Surveillance de la charge de la base de données avec Performance Insights sur .

Surveillance des bases de données mondiales d’Aurora grâce aux flux d’activité des bases de données

En utilisant la fonction de flux d’activité de base de données, vous pouvez surveiller et définir des alarmes pour auditer l’activité dans les clusters de bases de données de votre base de données

globale. Vous démarrez un flux d'activité de base de données sur chaque cluster de bases de données séparément. Chaque cluster fournit des données d'audit à son propre flux Kinesis au sein de sa propre Région AWS. Pour plus d'informations, consultez Surveillance d'Amazon Aurora à l'aide des flux d'activité de base de données.

Surveillance des bases de données globales basées sur Aurora MySQL

Pour consulter l'état d'une base de données globale basée sur Aurora MySQL, interrogez les tables information_schema.aurora_global_db_status et information_schema.aurora_global_db_instance_statu

> **Note**
>
> Les tables `information_schema.aurora_global_db_status` et `information_schema.aurora_global_db_instance_status` ne sont disponibles qu'avec Aurora MySQL 3.04.0 et versions ultérieures.

Pour surveiller une base de données globale basée sur Aurora MySQL

1. Connectez-vous au point de terminaison du cluster principal de la base de données globale à l'aide d'un client MySQL. Pour plus d'informations sur la connexion, consultez Connexion à Amazon Aurora Global Database.
2. Interrogez la table `information_schema.aurora_global_db_status` dans une commande mysql pour répertorier les volumes principal et secondaire. Cette requête renvoie les temps de retard des clusters de bases de données secondaires de la base de données globale, comme dans l'exemple suivant.

```
mysql> select * from information_schema.aurora_global_db_status;
```

```
AWS_REGION | HIGHEST_LSN_WRITTEN | DURABILITY_LAG_IN_MILLISECONDS |
 RPO_LAG_IN_MILLISECONDS | LAST_LAG_CALCULATION_TIMESTAMP | OLDEST_READ_VIEW_TRX_ID
-----------+---------------------+--------------------------------
+------------------------+---------------------------------
+------------------------
us-east-1  |           183537946 |                            0   |
      0 |  1970-01-01 00:00:00.000000     |               0
us-west-2  |           183537944 |                            428 |
      0 |  2023-02-18 01:26:41.925000     |               20806982
(2 rows)
```

La sortie inclut une ligne pour chaque cluster de bases de données de la base de données globale contenant les colonnes suivantes :

- AWS_REGION : Région AWS dans laquelle se trouve ce cluster de bases de données. Pour la liste des tables affichant les Régions AWS par moteur, consultez Disponibilité dans les Régions.
- HIGHEST_LSN_WRITTEN : numéro de séquence de journal (LSN) le plus élevé actuellement écrit sur ce cluster de bases de données.

  Un numéro de séquence de journal (LSN) est un numéro séquentiel unique qui identifie un enregistrement dans le journal des transactions de la base de données. Les LSN sont classés de telle sorte qu'un LSN plus grand représente une transaction ultérieure.
- DURABILITY_LAG_IN_MILLISECONDS : différence dans les valeurs d'horodatage entre `HIGHEST_LSN_WRITTEN` sur un cluster de bases de données secondaire et `HIGHEST_LSN_WRITTEN` sur le cluster de bases de données principal. Cette valeur est toujours égale à 0 sur le cluster de bases de données principal de la base de données globale Aurora.
- RPO_LAG_IN_MILLISECONDS : retard de l'objectif de point de reprise (RPO). Le retard RPO correspond au temps nécessaire au stockage de la transaction utilisateur COMMIT la plus récente sur un cluster de bases de données secondaire, après qu'elle a été stockée sur le cluster de bases de données principal de la base de données globale Aurora. Cette valeur est toujours égale à 0 sur le cluster de bases de données principal de la base de données globale Aurora.

  En termes simples, cette métrique calcule l'objectif de point de reprise pour chaque cluster de bases de données Aurora MySQL dans la base de données globale Aurora, c'est-à-dire la quantité de données qui risque d'être perdue en cas de panne. Comme pour la latence, le RPO est mesuré dans le temps.
- LAST_LAG_CALCULATION_TIMESTAMP : horodatage qui indique quand a eu lieu le dernier calcul des valeurs pour `DURABILITY_LAG_IN_MILLISECONDS` et `RPO_LAG_IN_MILLISECONDS`. Une valeur temporelle telle que `1970-01-01 00:00:00+00` signifie qu'il s'agit du cluster de bases de données principal.
- OLDEST_READ_VIEW_TRX_ID : ID de la transaction la plus ancienne vers laquelle l'instance de base de données d'enregistreur peut effectuer une purge.

3. Interrogez la table `information_schema.aurora_global_db_instance_status` pour répertorier toutes les instances de base de données secondaires pour le cluster de bases de données principal et les clusters de bases de données secondaires.

```
mysql> select * from information_schema.aurora_global_db_instance_status;
```

```
SERVER_ID            |              SESSION_ID              | AWS_REGION
 | DURABLE_LSN | HIGHEST_LSN_RECEIVED | OLDEST_READ_VIEW_TRX_ID |
 OLDEST_READ_VIEW_LSN | VISIBILITY_LAG_IN_MSEC
--------------------+--------------------------------------+------------
+-------------+----------------------+-------------------------
+----------------------+-----------------------
ams-gdb-primary-i2   | MASTER_SESSION_ID                    | us-east-1  |
 183537698   |                    0 |                      0 |
 0 |                      0
ams-gdb-secondary-i1 | cc43165b-bdc6-4651-abbf-4f74f08bf931 | us-west-2  |
 183537689   |            183537692 |                20806928 |
 183537682 |                       0
ams-gdb-secondary-i2 | 53303ff0-70b5-411f-bc86-28d7a53f8c19 | us-west-2  |
 183537689   |            183537692 |                20806928 |
 183537682 |                     677
ams-gdb-primary-i1   | 5af1e20f-43db-421f-9f0d-2b92774c7d02 | us-east-1  |
 183537697   |            183537698 |                20806930 |
 183537691 |                      21
(4 rows)
```

La sortie inclut une ligne pour chaque instance de base de données de la base de données globale contenant les colonnes suivantes :

- SERVER_ID : identifiant du serveur pour l'instance de base de données.
- SESSION_ID : identifiant unique pour la session en cours. La valeur `MASTER_SESSION_ID` identifie l'instance de base de données d'enregistreur (principale).
- AWS_REGION : Région AWS dans laquelle se trouve cette instance de base de données. Pour la liste des tables affichant les Régions AWS par moteur, consultez Disponibilité dans les Régions.
- DURABLE_LSN : LSN rendu durable dans le stockage.
- HIGHEST_LSN_RECEIVED : LSN le plus élevé reçu par l'instance de base de données depuis l'instance de base de données d'enregistreur.

- OLDEST_READ_VIEW_TRX_ID : ID de la transaction la plus ancienne vers laquelle l'instance de base de données d'enregistreur peut effectuer une purge.
- OLDEST_READ_VIEW_LSN : LSN le plus ancien utilisé par l'instance de base de données pour lire à partir du stockage.
- VISIBILITY_LAG_IN_MSEC : pour les lecteurs dans le cluster de bases de données principal, retard accumulé par cette instance de base de données par rapport à l'instance de base de données d'enregistreur en millisecondes. Pour les lecteurs dans un cluster de bases de données secondaire, retard accumulé par cette instance de base de données par rapport au volume secondaire en millisecondes.

Pour voir comment ces valeurs changent au fil du temps, examinez le bloc de transaction suivant où une insertion de table prend une heure.

```
mysql> BEGIN;
mysql> INSERT INTO table1 SELECT Large_Data_That_Takes_1_Hr_To_Insert;
mysql> COMMIT;
```

Dans certains cas, une déconnexion réseau peut se produire entre le cluster de bases de données principal et le cluster de bases de données secondaire après l'instruction `BEGIN`. Si tel est le cas, la valeur de DURABILITY_LAG_IN_MILLISECONDS du cluster de bases de données secondaire commence à augmenter. À la fin de l'instruction `INSERT`, la valeur de DURABILITY_LAG_IN_MILLISECONDS est de 1 heure. Toutefois, la valeur de RPO_LAG_IN_MILLISECONDS est 0, car toutes les données utilisateur validées entre le cluster de bases de données principal et le cluster de bases de données secondaire sont encore les mêmes. Dès que l'instruction `COMMIT` se termine, la valeur de RPO_LAG_IN_MILLISECONDS augmente.

Surveillance des bases de données globales basées sur Aurora PostgreSQL

Pour voir l'état d'une base de données globale basée sur Aurora PostgreSQL, utilisez les fonctions `aurora_global_db_status` et `aurora_global_db_instance_status`.

> **Note**
>
> Seul Aurora PostgreSQL prend en charge les fonctions `aurora_global_db_status` et `aurora_global_db_instance_status`.

Pour surveiller une base de données globale basée sur Aurora PostgreSQL

1. Connectez-vous au point de terminaison principal de cluster de bases de données globale à l'aide d'un utilitaire PostgreSQL tel que psql. Pour plus d'informations sur la connexion, consultez Connexion à Amazon Aurora Global Database.
2. Utilisez la fonction `aurora_global_db_status` d'une commande psql pour répertorier les volumes primaire et secondaire. Ceci affiche les temps de latence des clusters de bases de données secondaires de la base de données globale.

```
postgres=> select * from aurora_global_db_status();
```

```
aws_region | highest_lsn_written | durability_lag_in_msec | rpo_lag_in_msec |
 last_lag_calculation_time  | feedback_epoch | feedback_xmin
------------+---------------------+------------------------+-----------------
+----------------------------+----------------+---------------
us-east-1  |         93763984222 |                     -1 |              -1 |
 1970-01-01 00:00:00+00     |              0 |              0
us-west-2  |         93763984222 |                    900 |            1090 |
 2020-05-12 22:49:14.328+00 |              2 |    3315479243
(2 rows)
```

La sortie inclut une ligne pour chaque cluster de bases de données de la base de données globale contenant les colonnes suivantes :

- aws_region : Région AWS dans laquelle se trouve ce cluster de bases de données. Pour la liste des tables affichant les Régions AWS par moteur, consultez Disponibilité dans les Régions.
- highest_lsn_written : numéro de séquence de journal (LSN) le plus élevé actuellement écrit sur ce cluster de bases de données.

  Un numéro de séquence de journal (LSN) est un numéro séquentiel unique qui identifie un enregistrement dans le journal des transactions de la base de données. Les LSN sont classés de telle sorte qu'un LSN plus grand représente une transaction ultérieure.
- durability_lag_in_msec – Différence d'horodatage entre le numéro de séquence de journal le plus élevé écrit sur un cluster de bases de données secondaire (`highest_lsn_written`) et le `highest_lsn_written` sur le cluster de bases de données principal.
- rpo_lag_in_msec : retard de l'objectif de point de reprise (RPO). Cette latence correspond à la différence de temps entre la validation de transaction utilisateur la plus récente stockée sur

un cluster de bases de données secondaire et la validation de transaction utilisateur la plus récente stockée sur le cluster de bases de données principal.

- last_lag_calculation_time – Horodatage lors du dernier calcul des valeurs pour `durability_lag_in_msec` et `rpo_lag_in_msec`.
- feedback_epoch – Époque utilisée par un cluster de bases de données secondaire lorsqu'il génère des informations de veille à chaud.

  On appelle veille à chaud le moment où un cluster de bases de données peut se connecter et interroger pendant que le serveur est en mode de récupération ou veille. Les commentaires de veille à chaud sont des informations sur le cluster de bases de données lorsqu'il est en veille à chaud. Pour plus d'informations, consultez Veille à chaud dans la documentation PostgreSQL.
- feedback_xmin – ID de transaction actif minimum (le plus ancien) utilisé par un cluster de bases de données secondaire.

3. Utilisez la fonction `aurora_global_db_instance_status` pour répertorier toutes les instances de base de données secondaires pour le cluster de bases de données principal et les clusters de bases de données secondaires.

```
postgres=> select * from aurora_global_db_instance_status();
```

```
server_id                                    |              session_id
  | aws_region | durable_lsn | highest_lsn_rcvd | feedback_epoch | feedback_xmin |
 oldest_read_view_lsn | visibility_lag_in_msec
----------------------------------------------+-------------------------------------
+------------+-------------+------------------+----------------+---------------
+----------------------+-----------------------
apg-global-db-rpo-mammothrw-elephantro-1-n1 | MASTER_SESSION_ID
 | us-east-1  | 93763985102 |                  |                |              |
                    |
apg-global-db-rpo-mammothrw-elephantro-1-n2 | f38430cf-6576-479a-b296-dc06b1b1964a
 | us-east-1  | 93763985099 |      93763985102 |              2 |    3315479243 |
        93763985095 |                      10
apg-global-db-rpo-elephantro-mammothrw-n1   | 0d9f1d98-04ad-4aa4-8fdd-e08674cbbbfe
 | us-west-2  | 93763985095 |      93763985099 |              2 |    3315479243 |
        93763985089 |                    1017
(3 rows)
```

La sortie inclut une ligne pour chaque instance de base de données de la base de données globale contenant les colonnes suivantes :

- server_id : identifiant du serveur pour l'instance de base de données.
- session_id – Identifiant unique pour la session en cours.
- aws_region – Région AWS dans laquelle se trouve cette instance de base de données. Pour la liste des tables affichant les Régions AWS par moteur, consultez Disponibilité dans les Régions.
- durable_lsn – Le LSN rendu durable dans le stockage.
- highest_lsn_rcvd – LSN le plus élevé reçu par l'instance de base de données depuis l'instance de base de données de l'enregistreur.
- feedback_epoch – L'époque utilisée par l'instance de base de données lorsqu'elle génère des informations de veille à chaud.

  On appelle veille à chaud le moment où une instance de base de données peut se connecter et interroger pendant que le serveur est en mode de récupération ou veille. Les commentaires de veille à chaud sont des informations sur l'instance de base de données lorsqu'elle est en veille à chaud. Pour plus d'informations, consultez la documentation PostgreSQL sur Veille à chaud.
- feedback_xmin – ID de transaction actif minimum (le plus ancien) utilisé par l'instance de base de données.
- oldest_read_view_lsn : LSN le plus ancien utilisé par l'instance de base de données pour lire à partir du stockage.
- visibility_lag_in_msec – Quelle est la latence de cette instance de base de données par rapport à l'instance de base de données rédacteur.

Pour voir comment ces valeurs changent au fil du temps, examinez le bloc de transaction suivant où une insertion de table prend une heure.

```
psql> BEGIN;
psql> INSERT INTO table1 SELECT Large_Data_That_Takes_1_Hr_To_Insert;
psql> COMMIT;
```

Dans certains cas, une déconnexion réseau peut se produire entre le cluster de bases de données principal et le cluster de bases de données secondaire après l'instruction `BEGIN`. Si c'est le cas, la valeur `durability_lag_in_msec` du cluster de bases de données secondaire commence à augmenter. À la fin de l'instruction `INSERT`, la valeur `durability_lag_in_msec` est 1 heure. Toutefois, la valeur `rpo_lag_in_msec` est 0, car toutes les données utilisateur validées entre le

cluster de bases de données principal et le cluster de bases de données secondaire sont encore les mêmes. Dès que l'instruction `COMMIT` est terminée, la valeur `rpo_lag_in_msec` augmente.

Utilisation de Blue/Green déploiements pour la base de données mondiale Amazon Aurora

Les Blue/Green déploiements Amazon RDS permettent de tester les modifications de base de données en toute sécurité. Pour votre base de données globale, blue/green les déploiements vous permettent d'effectuer des mises à niveau et des opérations de maintenance avec un minimum de temps d'arrêt. Vous pouvez créer un environnement intermédiaire entièrement géré (vert) qui reflète l'ensemble de votre environnement de production (bleu), y compris le cluster principal et toutes les régions secondaires associées dans plusieurs AWS régions. L'environnement de préparation reflète votre configuration de production, ce qui vous permet de tester les modifications de manière fiable avant de passer au nouvel environnement. Tout au long du processus, blue/green les déploiements permettent de synchroniser les environnements de préparation et de production. Lorsque vous êtes prêt à faire de votre environnement de préparation le nouvel environnement de production, effectuez une blue/green transition. Cette opération fait passer votre région principale et toutes les régions secondaires à un environnement vert, avec des temps d'arrêt généralement inférieurs à une minute. Le service renomme automatiquement les clusters, les instances et les points de terminaison pour qu'ils correspondent à l'environnement de production d'origine, ce qui permet à vos applications d'accéder au nouvel environnement de production sans nécessiter de modifications de configuration et en minimisant les frais opérationnels.

Rubriques

Avantages de l'utilisation des Blue/Green déploiements avec la base de données globale Aurora

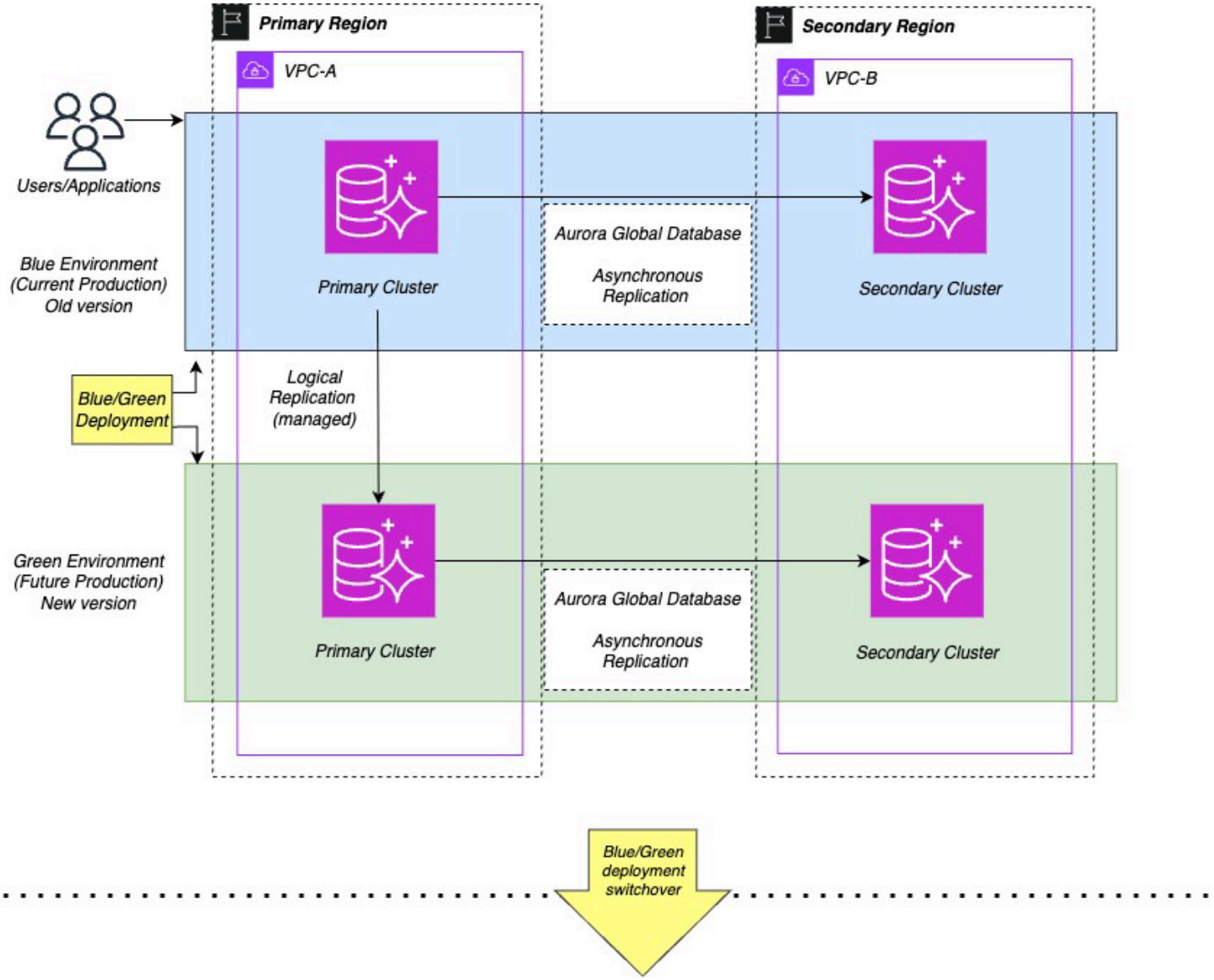
- Effectuez les mises à jour des versions majeures, mineures et du système, y compris les correctifs de base de données et les mises à niveau du système d'exploitation sur les bases de données mondiales Aurora, tout en minimisant les temps d'arrêt.

- Créez un environnement intermédiaire (vert) prêt pour la production dans les régions principale et secondaire de la base de données globale afin de tester et d'implémenter les nouvelles fonctionnalités de base de données.
- Passez au numérique en toute sécurité à l'aide de glissières de sécurité intégrées, avec des temps d'arrêt généralement inférieurs à une minute, en fonction de votre charge de travail.
- Maintient les capacités de reprise après sinistre pendant le processus de transition, ce qui permet le Blue/Green basculement de la base de données globale pendant le basculement. Blue/Green
- Votre trafic sera dirigé vers le nouvel environnement de production sans qu'aucune modification de l'application ne soit nécessaire.

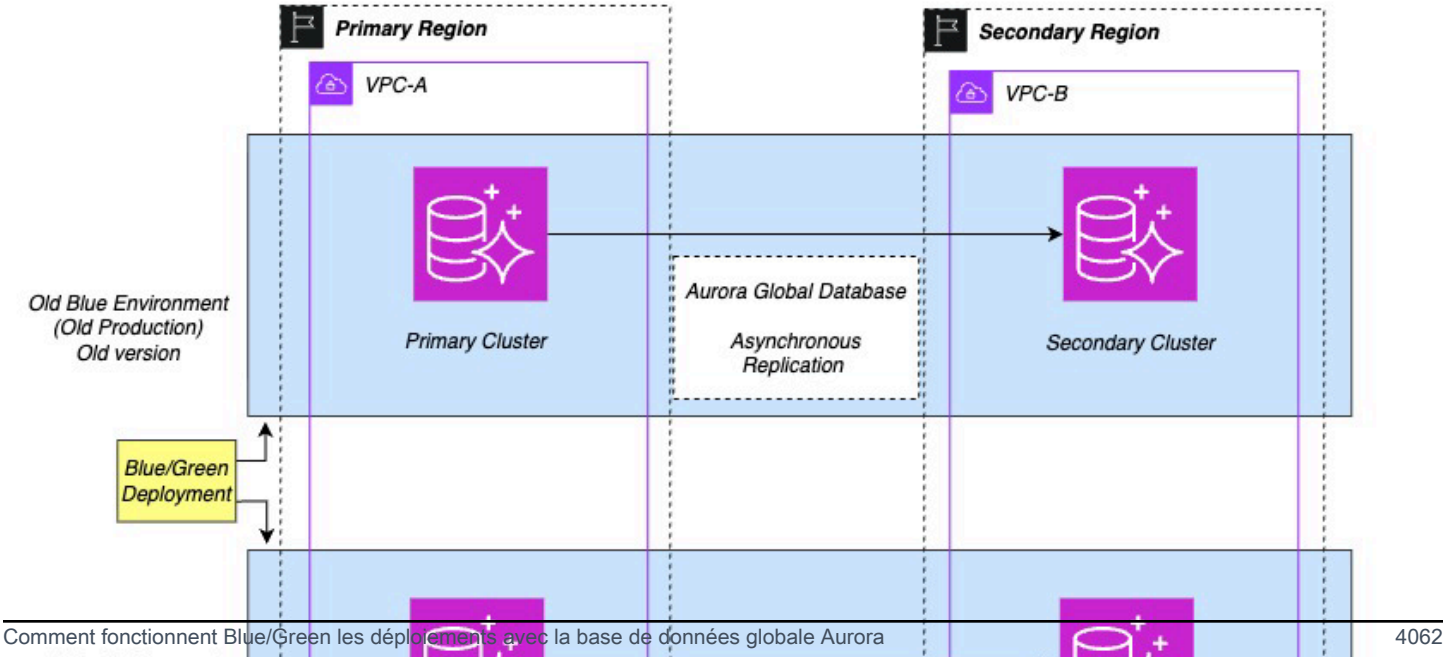
Comment fonctionnent Blue/Green les déploiements avec la base de données globale Aurora

Pour plus de détails sur la création, l'affichage, le changement et la suppression d'un Blue/Green déploiement, consultezUtilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données. Vous pouvez l'utiliser pour les mises à niveau de versions majeures ou mineures, l'amélioration des performances des bases de données grâce à des mises à jour des paramètres et l'adoption de nouvelles fonctionnalités de base de données, avec un temps d'arrêt minimal.

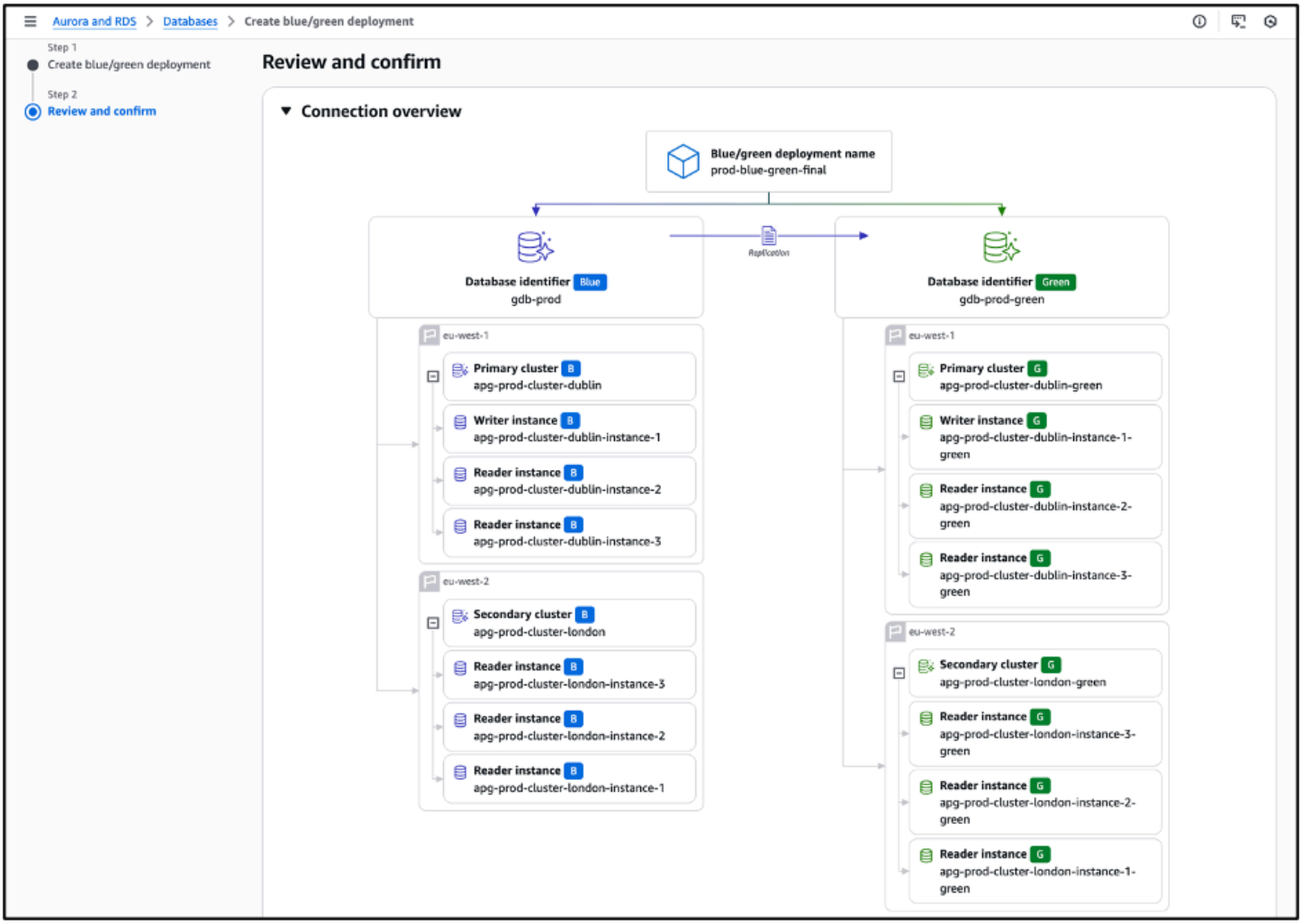
Vous trouverez ci-dessous une représentation de l'apparence d'un blue/green déploiement pour Aurora Global Database avec une région secondaire avant et après un blue/green passage au numérique.


Primary Region
VPC-A
Secondary Region
VPC-B
Users/Applications
Blue Environment
(Current Production)
Old version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
Deployment
Logical
Replication
(managed)
Green Environment
(Future Production)
New version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster

Blue/Green
deployment
switchover


Primary Region
VPC-A
Secondary Region
VPC-B
Old Blue Environment
(Old Production)
Old version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster
Blue/Green
Deployment
Blue Environment
(Current Production)
New version
Primary Cluster
Aurora Global Database
Asynchronous
Replication
Secondary Cluster

Vous pouvez créer un blue/green déploiement à partir de la région principale de votre base de données globale. Sélectionnez les configurations du moteur telles que la version principale ou mineure du moteur, le groupe de paramètres de base de données et le groupe de paramètres du cluster de base de données pour l'environnement écologique. Amazon RDS copie la topologie de l'environnement bleu pour l'environnement vert. Une représentation visuelle dans AWS Management Console est illustrée ci-dessous.



> **Note**
>
> Le basculement global est pris en charge lors d'un blue/green basculement, mais le basculement global n'est pas pris en charge lors d'un basculement. blue/green

Lorsque vous lancez un basculement global lors d'un blue/green basculement RDS, la région cible revient automatiquement à l'environnement bleu ou passe à l'environnement vert avant que le basculement global ne se produise.

Pour plus d'informations sur la création, l'affichage, le changement et la suppression de blue/green déploiements, consultezUtilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données. Suivez le même processus pour les bases de données globales, avec des instructions spécifiques indiquées dans les étapes pertinentes.

Utilisation des bases de données Amazon Aurora Global Database avec d’autres services AWS

Vous pouvez utiliser vos bases de données Aurora globales avec d’autres services AWS, tels qu’Amazon S3 et AWS Lambda. Pour ce faire, tous les clusters de bases de données Aurora de votre base de données globale doivent disposer des mêmes privilèges, fonctions externes, etc. dans les Régions AWS concernées. Étant donné qu’un cluster de bases de données Aurora secondaire en lecture seule dans une base de données Aurora globale peut être promu au rôle principal, nous vous recommandons de configurer à l’avance les privilèges d’écriture sur tous les clusters Aurora pour tous les services que vous envisagez d’utiliser avec votre base de données Aurora globale.

Les procédures suivantes résument les actions à entreprendre pour chaque Service AWS.

Pour appeler des fonctions AWS Lambda à partir d’une base de données Aurora globale

1. Pour tous les clusters Aurora qui constituent la base de données globale Aurora, suivez les procédures décrites dans Appel d’une fonction Lambda à partir d’un cluster de bases de données Amazon Aurora MySQL.
2. Pour chaque cluster de la base de données Aurora globale, définissez l’ARN du nouveau rôle IAM (IAM).
3. Pour autoriser les utilisateurs de base de données d’une base de données globale Aurora à appeler des fonctions Lambda, associez le rôle que vous avez créé dans Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS à chaque cluster de la base de données globale Aurora.
4. Configurez chaque cluster de la base de données globale Aurora pour autoriser les connexions sortantes à Lambda. Pour obtenir des instructions, consultez Activation de la communication réseau entre Amazon Aurora et d’autres services AWS.

Pour charger des données à partir de Amazon S3

1. Pour tous les clusters Aurora qui constituent la base de données globale Aurora, suivez les procédures décrites dans Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte stockés dans un compartiment Amazon S3.
2. Pour chaque cluster Aurora de la base de données globale, spécifiez l'Amazon Resource Name (ARN) du nouveau rôle IAM pour le paramètre de cluster de bases de données `aurora_load_from_s3_role` ou `aws_default_s3_role`. Si un rôle IAM n'est pas spécifié pour `aurora_load_from_s3_role`, Aurora utilise le rôle IAM spécifié dans `aws_default_s3_role`.
3. Pour autoriser les utilisateurs de base de données d'une base de données Aurora globale à accéder à S3, associez le rôle que vous avez créé dans Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS à chaque cluster Aurora de la base de données globale.
4. Configurez chaque cluster Aurora de la base de données globale pour autoriser les connexions sortantes à S3. Pour obtenir des instructions, consultez Activation de la communication réseau entre Amazon Aurora et d'autres services AWS.

Pour enregistrer les données interrogées dans Amazon S3

1. Pour tous les clusters Aurora qui constituent la base de données globale Aurora, suivez les procédures décrites dans Enregistrement de données d'un cluster de bases de données Amazon Aurora MySQL dans des fichiers texte stockés dans un compartiment Amazon S3 ou Exportation de données à partir d'un cluster de bases de données Aurora PostgreSQL vers Amazon S3.
2. Pour chaque cluster Aurora de la base de données globale, spécifiez l'Amazon Resource Name (ARN) du nouveau rôle IAM pour le paramètre de cluster de bases de données `aurora_select_into_s3_role` ou `aws_default_s3_role`. Si un rôle IAM n'est pas spécifié pour `aurora_select_into_s3_role`, Aurora utilise le rôle IAM spécifié dans `aws_default_s3_role`.
3. Pour autoriser les utilisateurs de base de données d'une base de données Aurora globale à accéder à S3, associez le rôle que vous avez créé dans Création d'un rôle IAM pour autoriser Amazon Aurora à accéder aux services AWS à chaque cluster Aurora de la base de données globale.
4. Configurez chaque cluster Aurora de la base de données globale pour autoriser les connexions sortantes à S3. Pour obtenir des instructions, consultez Activation de la communication réseau entre Amazon Aurora et d'autres services AWS.

Création d’une Amazon Aurora Global Database

La mise à niveau d’une base de données globale Aurora suit les mêmes procédures celles utilisées pour la mise à niveau des clusters de bases de données Aurora. Toutefois, voici quelques différences importantes à prendre en compte avant de démarrer le processus.

Nous vous recommandons de mettre à niveau les clusters de bases de données principaux et secondaires vers la même version. Vous ne pouvez effectuer un basculement de base de données entre régions géré sur une base de données globale Aurora que si les clusters de bases de données principal et secondaire possèdent les mêmes versions de moteur majeures, mineures et de niveaux de correctif. Cependant, les niveaux de correctif peuvent varier en fonction de la version mineure du moteur. Pour plus d’informations, consultez Compatibilité des niveaux de correctif pour les bascules ou basculements gérés entre régions.

Mises à niveau de version majeure.

Lorsque vous effectuez une mise à niveau de version majeure d’une Amazon Aurora Global Database, vous mettez à niveau le cluster de bases de données global plutôt que les clusters individuels qu’il contient.

Pour savoir comment mettre à niveau une base de données globale Aurora PostgreSQL vers une version majeure supérieure, consultez Mises à niveau majeures des bases de données globales.

> **Note**
>
> Avec une base de données globale Aurora basée sur Aurora PostgreSQL, vous ne pouvez pas effectuer de mise à niveau majeure du moteur de base de données Aurora si la fonction Objectif de point de reprise (RPO) est activée. Pour en savoir plus sur la fonction RPO, consultez Gestion des RPO pour les bases de données globales basées sur Aurora PostgreSQL–.

Pour savoir comment mettre à niveau une base de données globale Aurora MySQL vers une version majeure supérieure, consultez Mises à niveau majeures sur place des bases de données globales.

> **Note**
>
> Avec une base de données globale Aurora basée sur Aurora MySQL, vous ne pouvez effectuer une mise à niveau sur place d’Aurora MySQL version 2 vers la version 3 que si

le paramètre `lower_case_table_names` est défini sur sa valeur par défaut et si vous redémarrez la base de données globale.
Pour effectuer la mise à niveau d'une version majeure vers Aurora MySQL version 3 lors de l'utilisation de `lower_case_table_names`, procédez comme suit :

1. Supprimez toutes les régions secondaires du cluster global. Suivez les étapes de Dissociation d'un cluster d'une base de données Amazon Aurora globale.
2. Mettez à niveau la version du moteur de la région principale vers Aurora MySQL version 3. Suivez les étapes de Comment effectuer une mise à niveau sur place.
3. Ajoutez des régions secondaires au cluster global. Suivez les étapes de Ajout d'une Région AWS à une base de données Amazon Aurora globale.

Vous pouvez utiliser plutôt la méthode de restauration des instantanés. Pour plus d'informations, consultez Restauration à partir d'un instantané de cluster de bases de données.

Mises à niveau de version mineure.

Si vous effectuez une mise à niveau mineure sur une base de données Aurora globale, mettez à niveau tous les clusters secondaires avant de mettre à niveau le cluster principal.

Pour savoir comment mettre à niveau une base de données globale Aurora PostgreSQL vers une version mineure ultérieure, consultez Comment effectuer des mises à niveau de versions mineures et appliquer des correctifs. Pour savoir comment mettre à niveau une base de données globale Aurora MySQL vers une version mineure ultérieure, consultez Mise à niveau d'Aurora MySQL par modification de la version du moteur.

Avant d'effectuer la mise à niveau, passez en revue les considérations suivantes :

- La mise à niveau de la version mineure d'un cluster secondaire n'a aucune incidence sur la disponibilité ni l'utilisation du cluster principal.
- Un cluster secondaire doit disposer d'au moins une instance de base de données pour effectuer une mise à niveau mineure.
- Si vous mettez à niveau une base de données globale Aurora MySQL vers la version 2.11.*, vous devez mettre à niveau vos clusters de bases de données principal et secondaires vers exactement la même version, y compris le niveau de correctif.

- Si vous mettez à niveau une base de données globale Aurora MySQL, vous devez mettre à niveau vos clusters de bases de données principal et secondaires vers exactement la même version et le même niveau de correctif. Pour mettre à jour le niveau du correctif, appliquez toutes les actions de maintenance en attente sur le cluster secondaire.
- Pour permettre les bascules ou basculements entre régions, vous devrez mettre à niveau vos clusters de bases de données principal et secondaires pour qu'ils aient exactement la même version, y compris le niveau de correctif. Cette exigence s'applique à Aurora MySQL et à certaines versions d'Aurora PostgreSQL. Pour obtenir la liste des versions qui autorisent les bascules et les basculements entre des clusters exécutant différents niveaux de correctifs, consultez Compatibilité des niveaux de correctif pour les bascules ou basculements gérés entre régions.

Compatibilité des niveaux de correctif pour les bascules ou basculements gérés entre régions

Si votre base de données Aurora Global Database exécute l'une des versions de moteur mineures suivantes, vous pouvez effectuer des bascules ou basculements gérés entre régions même si les niveaux de correctif de vos clusters de bases de données principal et secondaires ne correspondent pas. Pour les versions de moteur mineures antérieures à celles de cette liste, vos clusters de bases de données principal et secondaires doivent exécuter les mêmes versions majeures et mineures, et les mêmes niveaux de correctif, afin de pouvoir effectuer des bascules ou basculements gérés entre régions. Assurez-vous de consulter les informations et les notes de version figurant dans le tableau suivant lorsque vous planifiez des mises à niveau pour votre cluster principal, vos clusters secondaires ou les deux.

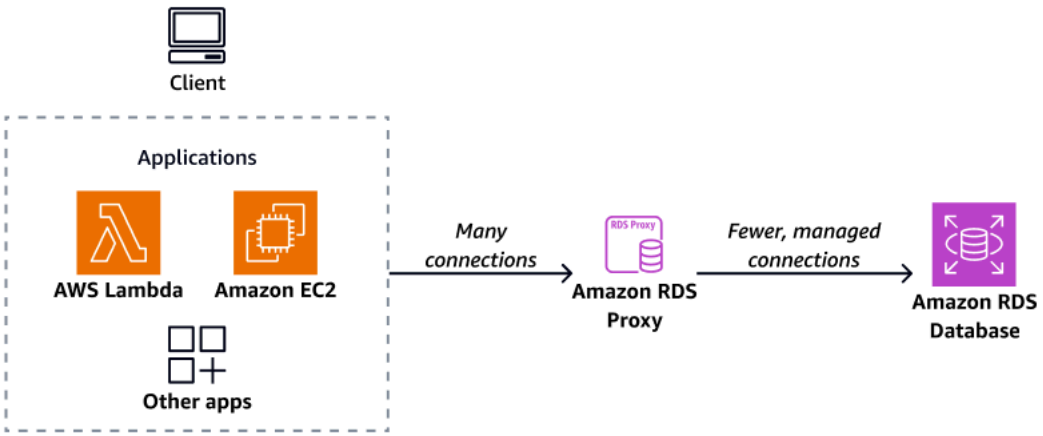
> **Note**
>
> Pour les basculements manuels entre régions, vous pouvez effectuer le processus de basculement tant que le cluster de bases de données secondaire cible exécute les mêmes versions majeure et mineure de moteur que le cluster de bases de données principal. Dans ce cas, il n'est pas nécessaire que les niveaux de correctifs soient les mêmes.
> Si les versions de votre moteur nécessitent des niveaux de correctifs identiques, vous pouvez effectuer le basculement manuellement en suivant les étapes décrites dans Réalisation de basculements manuels pour les bases de données globales Aurora.

Moteur de base de données	Versions de moteur mineures	Remarques
Aurora MySQL	Pas de version mineure	Aucune des versions mineures d'Aurora MySQL ne permet d'effectuer des bascules ou basculements géré entre régions si les niveaux de correctif des clusters de bases de données principal et secondaires diffèrent.
Aurora PostgreSQL	• Version 15 ou versions majeures ultérieures • Version 14.5 ou versions mineures ultérieures • Version 13.8 ou versions mineures ultérieures • Version 12.12 ou versions mineures ultérieures • Version 11.17 ou versions mineures ultérieures	Avec les versions de moteur répertoriées dans la colonne précédente, vous pouvez effectuer des bascules ou basculements gérés entre régions à partir d'un cluster de bases de données principal doté d'un niveau de correctif spécifique vers un cluster de bases de données secondaire doté d'un niveau de correctif différent . Avec les versions mineures antérieures à celles-ci, vous ne pouvez effectuer des bascules ou basculements gérés entre régions que si les niveaux de correctif des clusters de bases de données principal et secondaires sont les mêmes. **Warning** Lorsque vous mettez à jour un cluster de votre base de données globale vers l'une des versions de correctifs suivantes, vous ne pouvez

Moteur de base de données	Versions de moteur mineures	Remarques
		pas effectuer de bascule ni de basculement entre régions tant que tous les clusters de votre base de données globale n'exécutent pas l'une de ces versions de correctif ou une version plus récente. • Versions de correctif 16.1.6, 16.2.4, 16.3.2 et 16.4.2 • Versions de correctif 15.3.8, 15.4.9, 15.5.6, 15.6.4, 15.7.2 et 15.8.2 • Versions de correctif 14.8.8, 14.9.9, 14.10.6, 14.11.4, 14.12.2 et 14.13.2

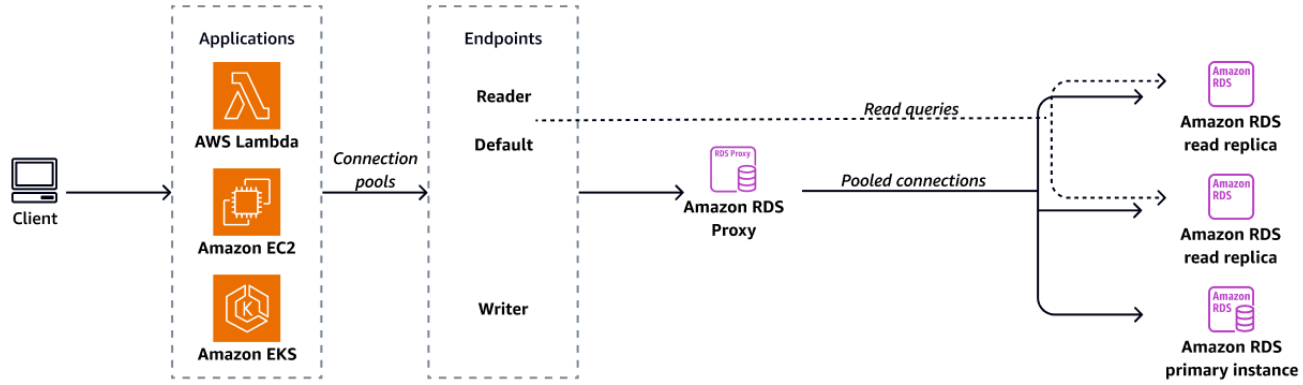
Proxy Amazon RDS pour Aurora

Le proxy Amazon RDS vous permet d'autoriser vos applications à grouper et à partager des connexions de bases de données pour améliorer leur capacité de mise à l'échelle. Le proxy RDS rend les applications plus résistantes aux échecs de base de données en les connectant automatiquement à une instance de base de données de secours tout en préservant les connexions des applications. En utilisant le proxy RDS, vous pouvez appliquer l'authentification Gestion des identités et des accès AWS (IAM) aux clients qui se connectent au proxy, et le proxy peut se connecter aux bases de données à l'aide de l'authentification de base de données IAM ou des informations d'identification stockées dans. AWS Secrets Manager



Avec le proxy RDS, vous pouvez gérer des pics imprévisibles de trafic des bases de données. Ces pics peuvent également occasionner des problèmes liés à la surallocation de connexions ou à la création de connexions à un rythme élevé. Le proxy RDS établit un groupe de connexions de bases de données et réutilise les connexions de ce groupe. Cette approche évite la surcharge de mémoire et d'UC liée à l'ouverture d'une nouvelle connexion de base de données à chaque fois. Pour protéger une base de données contre cette congestion, vous pouvez contrôler le nombre de connexions créées.

Le proxy RDS met en file d'attente ou limite les connexions des applications qui ne peuvent pas être servies immédiatement à partir du groupe de connexions. Bien que les latences puissent augmenter, votre application peut continuer à évoluer sans échouer brusquement ou surcharger la base de données. Si les demandes de connexion dépassent les limites que vous définissez, le proxy RDS rejette les connexions des applications (en d'autres termes, il déleste la charge). Parallèlement, il conserve des performances prévisibles pour la charge pouvant être desservie par RDS avec la capacité disponible.



Vous pouvez réduire la surcharge pour traiter des informations d'identification et établir une connexion sécurisée pour chaque nouvelle connexion. Le proxy RDS peut gérer une partie de ce travail pour le compte de la base de données.

Le proxy RDS est entièrement compatible avec les versions de moteur qu'il prend en charge. Vous pouvez activer le proxy RDS pour la plupart des applications sans modifier le code. Pour obtenir la liste des versions de moteur prises en charge, consultez Régions et moteurs de base de données Aurora pris en charge pour Proxy Amazon RDS.

Rubriques

- Disponibilité des régions et des versions
- Quotas et limites pour le proxy RDS
- Planification de l'emplacement où utiliser le proxy RDS
- Concepts et terminologie RDS Proxy
- Démarrage avec le proxy RDS
- Gestion d'un RDS Proxy
- Utilisation des points de terminaison du proxy Amazon RDS
- Surveillance des métriques RDS Proxy avec Amazon CloudWatch
- Utilisation des des événements RDS Proxy
- Exemples de ligne de commande pour le proxy RDS
- Résolution des problèmes liés au proxy RDS
- Utilisation de RDS Proxy avec AWS CloudFormation
- Utilisation du proxy RDS avec les bases de données globales Aurora

Disponibilité des régions et des versions

Pour plus d'informations sur la prise en charge des versions du moteur de base de données et la disponibilité du proxy RDS dans un environnement donnéRégion AWS, consultezRégions et moteurs de base de données Aurora pris en charge pour Proxy Amazon RDS.

Quotas et limites pour le proxy RDS

Voici les quotas et les limites qui s’appliquent au proxy RDS :

- Chaque Compte AWS identifiant est limité à 20 proxies. Si votre application nécessite davantage de proxies, demandez une augmentation via la page Service Quotas dans la AWS Management Console. Sur la page Service Quotas, sélectionnez Amazon Relational Database Service (Amazon RDS) et localisez les proxys pour demander une augmentation de quota. AWSpeut automatiquement augmenter votre quota ou attendre l’examen de votre demande parSupport.
- Chaque proxy peut avoir jusqu'à 200 secrets associés à Secrets Manager, limitant ainsi les connexions à un maximum de 200 comptes utilisateurs différents lors de l'utilisation de secrets.
- Chaque proxy a un point de terminaison par défaut. Vous pouvez également ajouter jusqu’à 20 points de terminaison de proxy pour chaque proxy. Vous pouvez créer, afficher, modifier et supprimer ces points de terminaison.
- Dans un cluster Aurora, toutes les connexions qui utilisent le point de terminaison proxy par défaut sont gérées par l’instance de rédacteur Aurora. Pour effectuer l’équilibrage des charges de travail exigeantes en lecture, vous pouvez créer un point de terminaison en lecture seule pour un proxy. Ce point de terminaison transmet des connexions au point de terminaison du lecteur du cluster. De cette façon, vos connexions proxy peuvent tirer avantage de l’évolutivité de lecture Aurora. Pour plus d’informations, consultez Présentation des points de terminaison proxy.
- Vous pouvez utiliser un proxy RDS avec des clusters Aurora sans serveur v2, mais pas avec des clusters Aurora sans serveur v1.
- Votre proxy RDS doit se trouver dans le même cloud privé virtuel (VPC) que la base de données. Le proxy peut ne pas être accessible au public, contrairement à la base de données. Par exemple, si vous effectuez un prototypage de votre base de données sur un hôte local, vous ne pouvez pas vous connecter à votre proxy à moins de mettre en place les exigences de réseau nécessaires pour autoriser la connexion au proxy. C’est le cas, car votre hôte local se trouve en dehors du VPC du proxy.

> **Note**
>
> Pour les clusters de bases de données Aurora, vous pouvez activer l’accès entre VPC. Pour ce faire, créez un point de terminaison supplémentaire pour un proxy et spécifiez un VPC, des sous-réseaux et des groupes de sécurité différents avec ce point de terminaison. Pour plus d’informations, consultez Accès aux bases de données Aurora via VPCs.

- Vous ne pouvez pas utiliser le proxy RDS avec un VPC dont la location est définie sur `dedicated`.
- Vous ne pouvez pas utiliser le proxy RDS dans un VPC dont les contrôles `Enforce Mode` de chiffrement sont activés.
- Pour les types de réseaux de points de IPv6 terminaison, configurez votre VPC et vos sous-réseaux pour qu'ils soient pris en charge uniquement. IPv6 Pour les deux types de réseaux de connexion IPv4 et ceux de connexion IPv6 cible, configurez votre VPC et vos sous-réseaux pour qu'ils prennent en charge le mode double pile.
- Si vous utilisez le proxy RDS avec un cluster d'instance de base de données sur lequel l'authentification IAM est activée, le proxy peut se connecter à la base de données à l'aide de l'authentification IAM ou des informations d'identification stockées dans Secrets Manager. Les clients qui se connectent au proxy doivent s'authentifier à l'aide des informations d'identification IAM. Pour obtenir des instructions de configuration détaillées, consultez Configuration des informations d'identification de base de données pour le proxy RDS et Configuration de l’authentification IAM pour RDS Proxy
- Vous ne pouvez pas utiliser le proxy RDS avec DNS personnalisé lorsque vous utilisez la validation du nom d’hôte SSL.
- Chaque proxy peut être associé à un cluster de bases de données unique. Toutefois, vous pouvez associer plusieurs proxies au même cluster de bases de données.
- Le proxy épingle la session à la connexion en cours si la taille de texte de l’instruction est supérieure à 16 ko.
- Certaines régions ont des restrictions de zone de disponibilité (AZ) à prendre en compte lors de la création de votre proxy. La région USA Est (Virginie du Nord) ne prend pas en charge le proxy RDS dans la zone de disponibilité `use1-az3`. La région USA Ouest (Californie du Nord) ne prend pas en charge le proxy RDS dans la zone de disponibilité `usw1-az2`. Lorsque vous sélectionnez des sous-réseaux lors de la création de votre proxy, assurez-vous de ne pas sélectionner de sous-réseaux dans les zones de disponibilité mentionnées ci-dessus.
- Actuellement, le proxy RDS ne prend en charge aucune clé de contexte de condition globale.

Pour plus d'informations sur les clés de contexte de condition globale, consultez Clés de contexte de condition globales AWS dans le Guide de l'utilisateur IAM.

- Vous ne pouvez pas utiliser le proxy RDS avec RDS Custom for SQL Server.
- Pour refléter toute modification du groupe de paramètres de base de données sur votre proxy, un redémarrage de l'instance est nécessaire même si vous avez choisi d'appliquer vos modifications immédiatement. Pour les paramètres au niveau du cluster, un redémarrage à l'échelle du cluster est requis.
- Votre proxy crée automatiquement l'utilisateur de base de données `rdsproxyadmin` lorsque vous enregistrez une cible proxy. Il s'agit d'un utilisateur protégé essentiel à la fonctionnalité du proxy. Vous devez éviter d'altérer l'utilisateur `rdsproxyadmin` de quelque manière que ce soit. La suppression ou la modification de l'utilisateur `rdsproxyadmin` ou de ses autorisations peut entraîner l'indisponibilité totale du proxy pour votre application.

Pour connaître les limites supplémentaires pour chaque moteur de base de données, consultez les sections suivantes :

- Limites supplémentaires pour Aurora MySQL
- Limites supplémentaires pour Aurora PostgreSQL

Limites supplémentaires pour Aurora MySQL

Les limites supplémentaires suivantes s'appliquent au proxy RDS avec les bases de données Aurora MySQL :

- La prise en charge du proxy RDS pour l'authentification `caching_sha2_password` nécessite une connexion sécurisée (TLS).
- La prise en charge du proxy RDS pour `caching_sha2_password` est connue pour présenter des problèmes de compatibilité avec certaines versions du pilote go-sql.
- Lorsque vous utilisez le pilote MySQL 8.4 C, l'API `mysql_stmt_bind_named_param` peut former des paquets mal formés si le nombre de paramètres dépasse le nombre d'espaces réservés dans une instruction préparée. Cela entraîne des réponses incorrectes. Pour plus d'informations, consultez Signalement de bogue MySQL.

- Actuellement, tous les proxies écoutent sur le port 3306 pour MySQL. Les proxys se connectent toujours à votre base de données à l'aide du port spécifié dans les paramètres de la base de données.
- Vous ne pouvez pas utiliser le proxy RDS avec des bases de données MySQL autogérées dans EC2 des instances.
- Vous ne pouvez pas utiliser le proxy RDS avec une instance de base de données RDS for MySQL dont le paramètre `read_only` dans son groupe de paramètres de base de données est défini sur `1`.
- Le proxy RDSvne prend pas en charge le mode compressé de MySQL. Par exemple, il ne prend pas en charge la compression utilisée par les options `--compress` ou `-C` de la commande `mysql`.
- Les connexions à la base de données qui traitent une commande `GET DIAGNOSTIC`peuvent renvoyer des informations inexactes lorsque le proxy RDS réutilise la même connexion à la base de données pour exécuter une autre requête. Cela peut se produire quand le proxy RDS multiplexe les connexions à la base de données.
- Certaines instructions et fonctions SQL, telles que `SET LOCAL`, peuvent modifier l'état de connexion sans provoquer d'épinglage. Pour connaître le comportement d'épinglage le plus récent, consultez Contournement de l'épinglage d'un proxy RDS.
- L'utilisation de la fonction `ROW_COUNT()` dans une requête à instructions multiples n'est pas prise en charge.
- Le proxy RDS ne prend pas en charge les applications clientes qui ne peuvent pas gérer plusieurs messages de réponse dans un seul enregistrement TLS.
- Le proxy RDS ne prend pas en charge les mots de passe doubles MySQL.
- Le proxy RDS peut ne pas fonctionner comme prévu lorsque vous configurez le paramètre `init_connect` dans votre groupe de paramètres de base de données RDS pour définir les variables d'état de session. Définissez plutôt la requête d'initialisation pour que votre proxy exécute les instructions d'initialisation de session lorsque vous utilisez le proxy pour vous connecter à votre base de données.

> **Important**
>
> Pour les proxies associés aux bases de données MySQL, ne définissez pas le paramètre de configuration `sql_auto_is_null` sur `true` ou sur une valeur différente de zéro dans la requête d'initialisation. Cela pourrait entraîner un comportement incorrect de l'application.

Limites supplémentaires pour Aurora PostgreSQL

Les limites supplémentaires suivantes s'appliquent au proxy RDS avec les bases de données Aurora PostgreSQL :

- Le proxy RDS ne prend pas en charge les filtres d'épinglage de session pour PostgreSQL.
- Actuellement, tous des proxies écoutent sur le port 5432 pour PostgreSQL.
- Pour PostgreSQL, RDS Proxy ne prend actuellement pas en charge l'annulation d'une requête d'un client en émettant un `CancelRequest`. C'est le cas par exemple lorsque vous annulez une requête longue dans une session psql interactive à l'aide de Ctrl+C.
- Les résultats de la fonctionnalité PostgreSQL lastval ne sont pas toujours précis. Pour contourner ce problème, utilisez l'instruction INSERT avec la clause `RETURNING`.
- Le proxy RDS ne prend actuellement pas en charge le mode de réplication de streaming.
- La `postgres` base de données par défaut doit exister sur l'instance RDS pour PostgreSQL pour que le proxy RDS fonctionne. Ne supprimez pas cette base de données même si votre application utilise des bases de données différentes.
- Si vous utilisez `ALTER ROLE` pour modifier le rôle de l'utilisateur avec `SET ROLE`, les connexions suivantes entre cet utilisateur et le proxy risquent de ne pas utiliser ce paramètre de rôle, si ces connexions sont épinglées. Pour éviter cela, lorsque vous utilisez un proxy, utilisez `SET ROLE` dans la requête d'initialisation du proxy. Pour plus d'informations, consultez Requête d'initialisation dans Création d'un proxy pour Amazon Aurora.

> **Important**
>
> Pour des proxies existants avec des bases de données PostgreSQL, si vous modifiez l'authentification de la base de données pour utiliser uniquement `SCRAM`, le proxy devient indisponible pendant 60 secondes maximum. Pour éviter ce problème, effectuez l'une des actions suivantes :
>
> - Veillez à ce que la base de données permette à la fois l'authentification `SCRAM` et `MD5`.
> - Pour utiliser uniquement l'authentification `SCRAM`, créez un nouveau proxy, migrez le trafic de votre application vers ce nouveau proxy, puis supprimez le proxy précédemment associé à la base de données.

Planification de l’emplacement où utiliser le proxy RDS

Vous pouvez déterminer les instances de base de données, clusters et applications qui pourraient bénéficier le plus de l’utilisation du proxy RDS. Pour ce faire, tenez compte des facteurs suivants :

- Il est judicieux d’associer à un proxy tout cluster de bases de données qui rencontre des erreurs liées à un « nombre de connexions trop élevé ». Cela se caractérise souvent par une valeur élevée de la `ConnectionAttempts` CloudWatch métrique. Le proxy permet aux applications d’ouvrir de nombreuses connexions client, tandis que le proxy gère un plus petit nombre de connexions à long terme au cluster de bases de données.
- Pour les clusters d' de base de données qui utilisent des classes d' AWS instance plus petites, telles que T2 ou T3, l'utilisation d'un proxy peut permettre d'éviter certaines out-of-memory conditions. Il peut également contribuer à réduire la surcharge de l’UC lors de l’établissement des connexions. Ces conditions peuvent se produire lorsque vous faites face à un grand nombre de connexions.
- Vous pouvez surveiller certaines CloudWatch métriques Amazon pour déterminer si un cluster d' de base de données approche certains types de limites. Ces limites concernent le nombre de connexions et la mémoire associées à la gestion des connexions. Vous pouvez également surveiller certaines CloudWatch mesures pour déterminer si un cluster d' de base de données gère de nombreuses connexions de courte durée. L’ouverture et la fermeture de telles connexions peuvent entraîner une surcharge de performances sur votre base de données. Pour en savoir plus sur les métriques à surveiller, consultez Surveillance des métriques RDS Proxy avec Amazon CloudWatch.
- AWS Lambda les fonctions peuvent également être de bons candidats pour l'utilisation d'un proxy. Ces fonctions réalisent fréquemment des connexions de base de données courtes qui bénéficient du regroupement de connexions offert par le proxy RDS. Vous pouvez profiter de toute authentification IAM dont vous disposez déjà pour les fonctions Lambda, plutôt que de gérer les informations d’identification de base de données dans votre code d’application Lambda.
- Les applications qui ouvrent et ferment généralement un grand nombre de connexions à des bases de données et qui ne disposent pas de mécanismes intégrés de regroupement des connexions sont de bons candidats pour l’utilisation d’un proxy.
- Il est souvent judicieux d’utiliser les applications qui maintiennent un grand nombre de connexions ouvertes pendant de longues périodes avec un proxy. Les applications dans des secteurs tels que le logiciel en tant que service (SaaS) ou le e-commerce réduisent souvent la latence pour les demandes de base de données en laissant les connexions ouvertes. Avec le proxy RDS, une

application peut garder davantage de connexions ouvertes que lorsqu'elle se connecte directement au cluster de bases de données.

- Vous n'avez peut-être pas adopté l'authentification IAM et Secrets Manager en raison de la complexité de la configuration de cette authentification pour tous les clusters d' de base de données. Le proxy peut appliquer les politiques d'authentification relatives aux connexions client pour des applications spécifiques. Vous pouvez profiter de toute authentification IAM dont vous disposez déjà pour les fonctions Lambda, plutôt que de gérer les informations d'identification de base de données dans votre code d'application Lambda.
- Le proxy RDS peut aider à rendre les applications plus résilientes et plus transparentes face aux pannes de bases de données. Le proxy RDS contourne les caches du système de nom de domaine (DNS) afin de réduire les temps de basculement jusqu'à 66 % pour les bases de données Aurora Multi-AZ. Le proxy RDS achemine automatiquement le trafic vers une nouvelle instance de base de données tout en préservant les connexions aux applications. Cela rend les basculements plus transparents pour les applications.

Concepts et terminologie RDS Proxy

Vous pouvez simplifier la gestion des connexions pour vos clusters de bases de données Amazon Aurora à l'aide de RDS Proxy.

RDS Proxy gère le trafic réseau entre l'application cliente et la base de données. Il le fait d'abord de manière active en comprenant le protocole de la base de données. Il ajuste ensuite son comportement en fonction des opérations SQL de votre application et des jeux de résultats de la base de données.

RDS Proxy réduit la charge de mémoire et d'UC pour la gestion des connexions sur votre base de données. La base de données a besoin de moins de mémoire et de ressources de l'UC lorsque les applications ouvrent de nombreuses connexions simultanées. La logique n'est pas non plus nécessaire dans vos applications pour fermer et rouvrir les connexions qui restent inactives pendant longtemps. De même, il faut logique d'application moindre pour rétablir les connexions en cas de problème de base de données.

L'infrastructure du proxy RDS est hautement disponible et déployée sur plusieurs zones de disponibilité (AZs). Le calcul, la mémoire et le stockage de RDS Proxy sont indépendants de votre cluster de bases de données Aurora. Cette séparation permet de réduire la surcharge sur vos serveurs de base de données, afin qu'ils puissent dédier leurs ressources à la gestion des

charges de travail de base de données. Les ressources de calcul de RDS Proxy sont sans serveur et automatiquement mises à l'échelle en fonction de la charge de travail de votre base de données.

Rubriques

Présentation des concepts RDS Proxy

RDS Proxy gère l'infrastructure pour effectuer le regroupement de connexions et les autres fonctions décrites dans les sections qui suivent. Vous voyez les serveurs proxy qui figurent dans la console RDS sur la page Proxys.

Chaque proxy gère les connexions à un seul cluster de bases de données Aurora. Le proxy détermine automatiquement l'instance d'enregistreur actuelle pour les clusters provisionnés Aurora.

Les connexions qu'un proxy garde ouvertes et disponibles pour vos applications de base de données constituent le groupe de connexions.

Par défaut, RDS Proxy peut réutiliser une connexion après chaque transaction dans votre session. « multiplexage » est le terme utilisé pour cette réutilisation au niveau de la transaction. Lorsque RDS Proxy supprime temporairement une connexion du groupe de connexions pour la réutiliser, cette opération est appelée un emprunt de connexion. lorsque l'opération peut être effectuée sans risque, RDS Proxy renvoie cette connexion au groupe de connexions.

Dans certains cas, RDS Proxy ne peut pas s'assurer que la réutilisation d'une connexion à une base de données en dehors de la session en cours peut être effectuée sans risque. Dans ce cas, il maintient la session sur la même connexion jusqu'à la fin. Ce comportement de secours est appelé épinglage.

Un proxy a un point de terminaison par défaut. Vous vous connectez à ce point de terminaison lorsque vous utilisez un cluster de bases de données Aurora. Vous le faites au lieu de vous connecter au read/write point de terminaison qui se connecte directement au cluster d'. Les points de terminaison à usage spécial d'un cluster Aurora restent disponibles pour vous. Pour les clusters

de base de données Aurora (clusters de), vous pouvez également créer des points de terminaison supplémentaires read/write en lecture seule. Pour de plus amples informations, veuillez consulter Présentation des points de terminaison proxy.

Par exemple, vous pouvez toujours vous connecter au point de terminaison du cluster pour read/write les connexions sans regroupement de connexions. Vous pouvez toujours vous connecter au point de terminaison du lecteur pour des connexions en lecture seule à charge équilibrée. Vous pouvez toujours vous connecter aux points de terminaison de l'instance pour le diagnostic et le dépannage d'instances de base de données spécifiques d'un cluster. Si vous utilisez d'autres AWS services, par exemple pour vous connecter AWS Lambda aux bases de données RDS, modifiez leurs paramètres de connexion pour utiliser le point de terminaison du proxy. Par exemple, vous indiquez au point de terminaison proxy de permettre aux fonctions de Lambda d'accéder à votre base de données tout en profitant des fonctionnalités de RDS Proxy.

Chaque proxy contient un groupe cible. Ce groupe cible incarne le cluster de bases de données Aurora auquel le proxy peut se connecter. Pour un cluster Aurora, le groupe cible est associé par défaut à toutes les instances de base de données de ce cluster. Le proxy peut ainsi se connecter à l'instance de base de données Aurora promue comme instance d'enregistreur dans le cluster. Le cluster de bases de données Aurora associé à un proxy est appelée cibles de ce proxy. Pour des raisons pratiques, lorsque vous créez un proxy via la console, RDS Proxy crée également le groupe cible correspondant et enregistre automatiquement les cibles associées.

Une famille de moteurs est un ensemble associé de moteurs de base de données qui utilisent le même protocole de base de données. Vous choisissez la famille de moteurs pour chaque proxy que vous créez.

Regroupement de connexions

Chaque proxy effectue le regroupement de connexions séparément pour l'instance d'enregistreur et de lecteur de sa base de données Aurora associée. Le regroupement de connexions est une optimisation qui réduit la surcharge associée à l'ouverture et à la fermeture des connexions, tout en maintenant plusieurs connexions ouvertes simultanément. Cette surcharge inclut la mémoire nécessaire pour gérer chaque nouvelle connexion. Cela implique également une surcharge du processeur pour fermer chaque connexion et en ouvrir une nouvelle. Les exemples incluent l'établissement de contacts Security/Secure TLS/SSL (Transport Layer Sockets Layer), l'authentification, les capacités de négociation, etc. Le regroupement de connexions simplifie la logique de votre application. Vous n'avez pas besoin d'écrire de code d'application pour minimiser le nombre de connexions ouvertes simultanées.

Chaque proxy effectue aussi le multiplexage de connexion, également connu sous le nom de réutilisation de connexion. Grâce au multiplexage, RDS Proxy exécute toutes les opérations d'une transaction à l'aide d'une connexion de base de données sous-jacente. RDS peut ensuite utiliser une connexion différente pour la transaction suivante. Si vous ouvrez de nombreuses connexions simultanées au proxy, celui-ci conserve un plus petit nombre de connexions ouvertes à l'instance ou au cluster de bases de données. Cela permet de réduire davantage la surcharge de mémoire pour les connexions sur le serveur de base de données. Cette technique réduit également le risque que des erreurs liées au « nombre de connexions trop élevé » se produisent.

Sécurité RDS Proxy

Le proxy RDS utilise les mécanismes de sécurité RDS existants tels que TLS/SSL et Gestion des identités et des accès AWS (IAM). Pour obtenir des informations générales sur ces fonctionnalités de sécurité, reportez-vous à la section Sécurité dans Amazon Aurora. Par ailleurs, commencez par découvrir la façon dont Aurora utilise l'authentification, l'autorisation et d'autres domaines de sécurité.

RDS Proxy peut agir comme une couche de sécurité supplémentaire entre les applications clientes et la base de données sous-jacente. Par exemple, vous pouvez vous connecter au proxy à l'aide de TLS 1.3, même si l'instance de base de données sous-jacente prend en charge une version antérieure de TLS. Vous pouvez vous connecter au proxy à l'aide d'un rôle IAM même si le proxy se connecte à la base de données à l'aide de la méthode d'authentification de l'utilisateur et du mot de passe de la base de données. Grâce à cette technique, vous pouvez appliquer de fortes exigences d'authentification pour les applications de base de données sans avoir à fournir un effort de migration coûteux pour les instances de base de données elles-mêmes.

Vous pouvez utiliser les méthodes d'authentification suivantes avec le proxy RDS :

- Informations d'identification de base
- Authentification IAM standard
- End-to-end Authentification IAM

Utilisation d'IAM avec RDS Proxy

RDS Proxy propose deux méthodes d'authentification IAM :

- Authentification IAM standard : appliquez l'authentification IAM pour les connexions à votre proxy pendant que le proxy se connecte à la base de données à l'aide des informations d'identification stockées dans Secrets Manager. Cela permet d'appliquer l'authentification IAM pour l'accès aux

bases de données, même si les bases de données utilisent l'authentification par mot de passe native. Le proxy récupère les informations d'identification de la base de données auprès de Secrets Manager et gère l'authentification auprès de la base de données pour le compte de votre application.

- End-to-end Authentification IAM : applique l'authentification IAM pour les connexions directement entre vos applications et votre base de données via le proxy. End-to-end L'authentification IAM simplifie votre configuration de sécurité et évite la gestion des informations d'identification de base de données dans Secrets Manager. Cette couche de sécurité supplémentaire renforce le contrôle d'accès basé sur l'IAM depuis l'application cliente vers la base de données.

Pour utiliser l'authentification IAM standard, configurez votre proxy pour utiliser les secrets de Secrets Manager pour l'authentification et activez l'authentification IAM pour les connexions client. Vos applications s'authentifient auprès du proxy via IAM, tandis que le proxy s'authentifie auprès de la base de données à l'aide des informations d'identification extraites de Secrets Manager.

Pour utiliser l'authentification end-to-end IAM, configurez votre proxy pour qu'il utilise l'authentification IAM lors de la définition du schéma d'authentification par défaut lors de la création ou de la modification de votre proxy.

Pour l'authentification end-to-end IAM, vous devez mettre à jour le rôle IAM associé au proxy pour accorder l'`rds-db:connect`autorisation. Grâce à l'authentification end-to-end IAM, il n'est plus nécessaire d'enregistrer des utilisateurs de base de données individuels auprès du proxy via les secrets de Secrets Manager.

Utilisation TLS/SSL avec RDS Proxy

Vous pouvez vous connecter au proxy RDS à l'aide du TLS/SSL protocole.

> **Note**
>
> Le proxy RDS utilise les certificats du AWS Certificate Manager (ACM). Si vous utilisez RDS Proxy, vous n’avez pas besoin de télécharger des certificats Amazon RDS ou de mettre à jour des applications utilisant des connexions RDS Proxy.

Afin d’appliquer TLS pour toutes les connexions entre le proxy et votre base de données, vous pouvez spécifier un paramètre Exiger la sécurité de la couche de transport lorsque vous créez ou modifiez un proxy dans la AWS Management Console.

Le proxy RDS peut également garantir que votre session est utilisée TLS/SSL entre votre client et le point de terminaison du proxy RDS. Pour que RDS Proxy procède ainsi, spécifiez l'exigence côté client. Les variables de session SSL ne sont pas définies pour les connexions SSL à une base de données utilisant RDS Proxy.

- Pour Aurora MySQL, spécifiez l'exigence côté client avec le paramètre `--ssl-mode` lorsque vous exécutez la commande `mysql`.
- Pour et Aurora PostgreSQL, spécifiez `sslmode=require` comme partie de la chaîne `conninfo` lorsque vous exécutez la commande `psql`.

Le proxy RDS prend en charge le protocole TLS versions 1.0, 1.1 ; 1.2 et 1.3. Vous pouvez vous connecter au proxy à l'aide d'une version de TLS supérieure à celle utilisée dans la base de données sous-jacente.

Par défaut, les programmes client établissent une connexion chiffrée avec RDS Proxy. L'option `--ssl-mode` fournit davantage de contrôle. Du côté client, RDS Proxy prend en charge tous les modes SSL.

Pour le client, les modes SSL sont les suivants :

PREFERRED

SSL est le premier choix, mais n'est pas obligatoire.

DISABLED

Aucun mode SSL n'est autorisé.

REQUIRED

SSL est obligatoire.

VERIFY_CA

SSL est obligatoire et une vérification de l'autorité de certification (CA) est effectuée.

VERIFY_IDENTITY

SSL est obligatoire et une vérification de l'autorité de certification (CA) et de son nom d'hôte est effectuée.

Lorsque vous utilisez un client avec `--ssl-mode` `VERIFY_CA` ou `VERIFY_IDENTITY`, spécifiez l'option `--ssl-ca` pointant vers une autorité de certification au format `.pem`. Pour utiliser le `.pem`

fichier, téléchargez toutes les autorités PEMs de certification racine depuis Amazon Trust Services et placez-les dans un seul `.pem` fichier.

Le proxy RDS utilise des certificats à caractères génériques, qui s'appliquent à un domaine et à ses sous-domaines. Si vous utilisez le client `mysql` pour vous connecter avec le mode SSL `VERIFY_IDENTITY`, vous devez actuellement exécuter la commande `mysql` compatible avec MySQL 8.0.

Basculement

Le basculement est une fonctionnalité de haute disponibilité qui remplace une instance de base de données par une autre lorsque l'instance d'origine est indisponible. Un problème lié à une instance de base de données peut entraîner un basculement. Celui-ci peut également faire partie de procédures de maintenance normales, lors d'une mise à niveau de la base de données par exemple. Le basculement s'applique aux clusters de bases de données Aurora dotés d'une ou plusieurs instances de lecteur en plus de l'instance d'enregistreur.

La connexion via un proxy rend vos applications plus résistantes aux basculements de base de données. Lorsque l'instance de base de données d'origine est indisponible, RDS Proxy se connecte à la base de données de secours sans supprimer les connexions d'application inactives. Cela permet d'accélérer et de simplifier le processus de basculement. Ce basculement interrompt moins longtemps votre application par rapport à un problème de redémarrage ou de base de données classique.

Sans RDS Proxy, un basculement provoque une brève interruption de service. Pendant la panne, vous ne pouvez pas effectuer d'opérations d'écriture sur la base de données en cours de basculement. Toutes les connexions de base de données existantes sont interrompues et votre application doit les rouvrir. La base de données est ouverte à de nouvelles connexions et opérations d'écriture lorsqu'une instance de base de données en lecture seule est promue pour remplacer celle qui n'est pas disponible.

Pendant les basculements de base de données, RDS Proxy continue d'accepter les connexions à la même adresse IP et redirige automatiquement les connexions vers la nouvelle instance de base de données principale. Les clients qui se connectent via RDS Proxy ne sont pas sujets aux éléments suivants :

- Délais de propagation du système de noms de domaine (DNS) lors du basculement.
- Mise en cache DNS locale.

- Délai d’expiration de connexion.
- Incertitude concernant l’instance de base de données qui est le rédacteur en cours.
- Attente d’une réponse à la requête d’un ancien rédacteur devenu indisponible sans fermer les connexions.

Pour les applications qui conservent leur propre regroupement de connexions, passer par RDS Proxy implique que la plupart des connexions restent actives pendant des basculements ou d’autres interruptions. Seules les connexions qui se trouvent au milieu d’une transaction ou d’une instruction SQL sont annulées. RDS Proxy accepte immédiatement les nouvelles connexions. Lorsque le rédacteur de base de données n’est pas disponible, RDS Proxy place les demandes entrantes dans la file d’attente.

Pour les applications qui ne conservent pas leurs propres regroupements de connexions, RDS Proxy offre des taux de connexion plus rapides et davantage de connexions ouvertes. Il permet de réduire la surcharge coûteuse des reconnexions fréquentes à la base de données. Il effectue cette opération en réutilisant les connexions de base de données maintenues dans le regroupement de connexions de RDS Proxy. Cette approche est particulièrement importante pour les connexions TLS, où les coûts d’installation sont importants.

Transactions

Toutes les instructions d’une seule transaction utilisent toujours la même connexion à la base de données sous-jacente. La connexion devient disponible pour une session différente lorsque la transaction se termine. Voici les conséquences de l’utilisation de la transaction en tant qu’unité de granularité :

- La connexion peut être réutilisée après chaque instruction individuelle lorsque le paramètre Aurora MySQL `autocommit` est activé.
- Inversement, lorsque le paramètre `autocommit` est désactivé, la première instruction que vous émettez dans une session lance une nouvelle transaction. Par exemple, supposons que vous saisissiez une séquence `SELECT`, `INSERT`, `UPDATE`, ainsi que d’autres instructions en langage de manipulation de données (DML). Dans ce cas, la réutilisation de la connexion ne se produit que lorsque vous émettez `COMMIT`, `ROLLBACK` ou que vous mettez fin à la transaction.
- La saisie d’une instruction en langage de définition de données (DDL) entraîne la fin de la transaction une fois l’instruction terminée.

RDS Proxy détecte lorsqu'une transaction se termine par le protocole réseau utilisé par l'application cliente de base de données. La détection des transactions ne repose pas sur des mots-clés tels que `COMMIT` ou `ROLLBACK` apparaissant dans le texte de l'instruction SQL.

Dans certains cas, RDS Proxy peut détecter une demande de base de données qui rend impossible le déplacement de votre session vers une autre connexion. Dans ce cas, il désactive le multiplexage pour cette connexion pendant le reste de votre session. La même règle s'applique si RDS Proxy ne peut pas s'assurer de la praticité du multiplexage pour la session. Cette opération est appelée épinglage. Pour savoir comment détecter et réduire l'épinglage, consultez Contournement de l'épinglage d'un proxy RDS.

Démarrage avec le proxy RDS

Utilisez les informations des pages suivantes pour configurer, gérer le Proxy Amazon RDS pour Aurora et définir les options de sécurité associées. Les options de sécurité contrôlent les accès à chaque proxy et la connexion de ces derniers aux instances de base de données.

Si vous utilisez le proxy RDS pour la première fois, nous vous recommandons de suivre les pages dans l'ordre dans lequel nous les présentons.

Rubriques

- Configuration des prérequis réseau pour un proxy RDS
- Configuration des informations d'identification de base de données pour le proxy RDS
- Configuration de l'authentification IAM pour RDS Proxy
- Création d'un proxy pour Amazon Aurora
- Affichage d'un proxy
- Connexion à une base de données via RDS Proxy

Configuration des prérequis réseau pour un proxy RDS

L'utilisation de RDS Proxy nécessite un cloud privé virtuel (VPC) commun entre votre cluster de bases de données Aurora et RDS Proxy. Ce VPC doit avoir au moins deux sous-réseaux situés dans des zones de disponibilité différentes. Votre compte peut posséder ces sous-réseaux ou les partager avec d'autres comptes. Pour plus d'informations sur le partage VPC, consultez la section Travailler avec le partage. VPCs

Pour le IPv6 support, une configuration réseau supplémentaire est requise :

- IPv6 types de réseaux de points de terminaison : votre VPC et vos sous-réseaux doivent être configurés pour prendre en charge. IPv6 Cela inclut l'attribution de blocs IPv6 CIDR à votre VPC et à vos sous-réseaux.
- Types de réseaux de points de terminaison à double pile : votre VPC et vos sous-réseaux doivent prendre en charge IPv4 les deux types et l'adressage. IPv6
- IPv6 types de réseaux de connexion cibles : votre base de données doit être configurée pour le mode double pile afin de prendre en charge IPv6 les connexions depuis le proxy.

Les ressources de votre application cliente telles qu'Amazon EC2, Lambda ou Amazon ECS peuvent se trouver dans le même VPC que le proxy. Elles peuvent également se trouver dans un VPC distinct du proxy. Si vous êtes bien connecté à des clusters de base de données Aurora, vous disposez déjà des ressources réseau requises.

Rubriques

Obtention d’informations sur vos sous-réseaux

Si vous commencez tout juste à utiliser RDS ou Aurora, vous pouvez apprendre les bases de la connexion à une base de données en suivant les procédures de la section Configuration de votre environnement pour Amazon Aurora. Vous pouvez également suivre le tutoriel disponible dans Mise en route avec Amazon Aurora.

Pour créer un proxy, vous devez fournir les sous-réseaux et le VPC au sein desquels le proxy fonctionne. L'exemple Linux suivant montre des AWS CLI commandes qui examinent les sous-réseaux VPCs et appartenant à votre Compte AWS. En particulier, vous transmettez le sous-réseau IDs en tant que paramètres lorsque vous créez un proxy à l'aide de la CLI.

```
aws ec2 describe-vpcs
aws ec2 describe-internet-gateways
aws ec2 describe-subnets --query '*[].[VpcId,SubnetId]' --output text | sort
```

L'exemple Linux suivant montre des AWS CLI commandes permettant de déterminer le sous-réseau IDs correspondant à un cluster de base de spécifique.

Pour un cluster Aurora, vous recherchez d'abord l'ID de l'une des instances de base de données associées. Vous pouvez extraire le sous-réseau IDs utilisé par cette instance de base de données. Pour ce faire, examinez les champs imbriqués dans les attributsDBSubnetGroup et `Subnets` dans la sortie de description de l'instance de base de données. Vous spécifiez une partie ou la totalité de ces sous-réseaux IDs lorsque vous configurez un proxy pour ce serveur de base de données.

```
$ # Find the ID of any DB instance in the cluster.
$  aws rds describe-db-clusters --db-cluster-identifier my_cluster_id --query '*[].
[DBClusterMembers]||[0]||[0][*].DBInstanceIdentifier' --output text
```

```
my_instance_id
instance_id_2
instance_id_3
```

Après avoir recherché l'identifiant de l'instance de base de données, examinez le VPC associé pour rechercher ses sous-réseaux. Pour ce faire, examinez l'exemple Linux suivant.

```
$ #From the DB instance, trace through the DBSubnetGroup and Subnets to find the subnet
 IDs.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]||[0]||[0]||[Subnets]||[0]||[*].SubnetIdentifier' --output text
```

```
subnet_id_1
subnet_id_2
subnet_id_3
...
```

```
$ #From the DB instance, find the VPC.
$ aws rds describe-db-instances --db-instance-identifier my_instance_id --query '*[].
[DBSubnetGroup]||[0]||[0].VpcId' --output text
```

```
my_vpc_id
```

```
$ aws ec2 describe-subnets --filters Name=vpc-id,Values=my_vpc_id --query '*[].
[SubnetId]' --output text
```

```
subnet_id_1
subnet_id_2
```

```
subnet_id_3
subnet_id_4
subnet_id_5
subnet_id_6
```

Planification de la capacité des adresses IP

Le proxy RDS ajuste automatiquement sa capacité en fonction de la configuration des instances de base de données enregistrées auprès de celui-ci. Pour les instances provisionnées, cela est déterminé par la taille de l'instance et pour les Aurora Serverless v2 instances, cela est déterminé par la capacité maximale de l'ACU. Certaines opérations peuvent également nécessiter des capacités supplémentaires, telles que l'augmentation de la taille d'une base de données enregistrée ou les opérations de maintenance internes du proxy RDS. Au cours de ces opérations, votre proxy peut avoir besoin de plus d'adresses IP pour fournir de la capacité supplémentaire. Ces adresses supplémentaires permettent à votre proxy d'évoluer sans affecter votre charge de travail. L'absence d'adresses IP libres dans vos sous-réseaux empêche d'augmenter un proxy. Cela peut entraîner des latences de requêtes plus élevées ou des échecs de connexion client. RDS vous avertit par le biais de l'événement `RDS-EVENT-0243` lorsqu'il n'y a pas suffisamment d'adresses IP libres dans vos sous-réseaux. Pour plus d'informations sur cet événement, consultez Utilisation des des événements RDS Proxy.

Réservez le nombre minimal suivant d'adresses IP libres dans vos sous-réseaux pour votre proxy, en fonction des tailles de classes d'instance de base de données.

Classe d'instance de base de données	Nombre minimal d'adresses IP libres
db.*.xlarge ou moins	10
db.*.2xlarge	15
db.*.4xlarge	25
db.*.8xlarge	45
db.*.12xlarge	60
db.*.16xlarge	75
db.*.24xlarge	110

En Aurora Serverless v2 effet, réservez le nombre minimum d'adresses IP libres suivantes dans vos sous-réseaux pour votre proxy, en fonction de la capacité maximale de l'ACU.

Maximum ACU Capacity	Nombre minimal d’adresses IP libres
16 ou moins	10
32	15
64	25
96	30
128	40
160	50
192	55
224	65
256	75

> **Note**
>
> Le proxy RDS ne consomme pas plus de 215 adresses IP pour chaque proxy d'un VPC.

Le proxy RDS nécessite un minimum de 10 adresses IP pour votre base de données Aurora. Ces nombres recommandés d’adresses IP sont des estimations pour un proxy avec uniquement le point de terminaison par défaut. Pour chaque point de terminaison personnalisé supplémentaire, nous vous recommandons de réserver trois adresses IP supplémentaires. Pour chaque instance de lecteur Aurora, nous vous recommandons de réserver des adresses IP supplémentaires, comme indiqué dans le tableau, en fonction de la taille maximale de ce lecteur ACUs pour la Aurora Serverless v2 cible ou de la taille d'instance de base de données pour une cible provisionnée.

Pour estimer les adresses IP requises pour un proxy associé à un cluster de base de données Aurora avec :

- 1 instance d'enregistreur provisionnée de taille `db.r5.8xlarge` et 1 instance de lecteur provisionnée de taille. `db.r5.2xlarge`
- Le proxy attaché à ce cluster possède un point de terminaison par défaut et un point de terminaison personnalisé doté du rôle en lecture seule.

Dans ce cas, le proxy a besoin d'environ 63 adresses IP libres (45 pour l'instance d'enregistreur, 15 pour l'instance de lecture et 3 pour le point de terminaison personnalisé supplémentaire).

Pour estimer les adresses IP requises pour un proxy associé à un cluster de base de données Aurora doté de :

- 1 instance d'Aurora Serverless v2écriture d'une capacité maximale de 256 ACUs et 1 instance de lecteur v2 sans serveur d'une capacité maximale de 192 ACUs.
- Le proxy attaché à ce cluster possède le point de terminaison par défaut et un point de terminaison personnalisé doté du rôle en lecture seule.

Dans ce cas, le proxy a besoin d'environ 133 adresses IP gratuites (75 pour l'instance du rédacteur, 55 pour l'instance du lecteur et 3 pour le point de terminaison personnalisé supplémentaire).

Pour estimer les adresses IP requises pour un proxy associé à un cluster Aurora possédant :

- 1 instance de rédacteur provisionnée avec une taille d'instance de base de données de db.r5.4xlarge et 1 instance de lecteur Serverless v2 d'une capacité maximale de 64. ACUs
- Le proxy attaché à ce cluster possède le point de terminaison par défaut et un point de terminaison personnalisé doté du rôle en lecture seule.

Dans ce cas, le proxy a besoin d'environ 53 adresses IP libres (25 pour l'instance du rédacteur, 25 pour l'instance du lecteur et 3 pour le point de terminaison personnalisé supplémentaire).

Pour estimer les adresses IP requises pour un proxy associé à un cluster de base de données Aurora doté de :

- 1 instance d'enregistreur provisionnée de taille db.r5.24xlarge et 3 instances de lecteur provisionnées de taille db.r5.8xlarge.
- Le proxy associé à ce cluster de bases de données possède le point de terminaison par défaut et 1 point de terminaison personnalisé avec le rôle en lecture seule.

Dans ce cas, le proxy a besoin de 215 adresses IP gratuites. Alors que les calculs suggèrent 248 IPs (110 + (3*45) + 3), le proxy RDS ne consomme pas plus de 215 adresses IP pour chaque proxy d'un VPC.

Configuration des informations d'identification de base de données pour le proxy RDS

Le proxy RDS dans Amazon RDS est utilisé AWS Secrets Manager pour stocker et gérer les informations d'identification de base de données en toute sécurité. Au lieu d'intégrer des informations d'identification dans votre application, vous associez un proxy à un secret Secrets Manager contenant les informations d'authentification nécessaires. Vous créez un secret Secrets Manager distinct pour chaque compte d'utilisateur de base de données auquel le proxy se connecte sur le cluster de bases de données Aurora.

Vous pouvez également configurer le proxy RDS pour utiliser l'authentification end-to-end IAM, ce qui élimine le besoin de stocker les informations d'identification de la base de données dans Secrets Manager. Le proxy RDS utilise l'authentification IAM pour les connexions à la fois client-to-proxy. proxy-to-database Cela fournit une solution d'authentification basée sur IAM entièrement intégrée qui ne nécessite pas de gérer des secrets ou des mots de passe. Pour plus d'informations sur l'ajout d'un nouvel utilisateur de base de données IAM, consultezCréation d'un compte de base de données à l'aide de l'authentification IAM.

Rubriques

- Création de secrets à utiliser avec le proxy RDS

Création de secrets à utiliser avec le proxy RDS

Avant de créer un proxy, vous devez d'abord créer au moins un secret qui stocke les informations d'identification de votre base de données.

Console

Pour créer un secret

1. Ouvrez la console Secrets Manager à l'adresse https://console.aws.amazon.com/secretsmanager/.
2. Choisissez Store a new secret (Stocker un nouveau secret).
3. Choisissez Informations d'identification pour une base de données Amazon RDS.

4. Entrez un nom d'utilisateur et un mot de passe. Les informations d'identification que vous entrez doivent correspondre aux informations d'identification d'un utilisateur de base de données existant dans la base de données RDS associée. Le proxy RDS utilise ces informations d'identification pour authentifier et établir des connexions à la base de données pour le compte des applications.

   En cas d'incompatibilité, vous pouvez mettre à jour le secret pour qu'il corresponde à celui de la base de données. Tant que vous ne mettez pas à jour le secret, les tentatives de connexion via le proxy utilisant ce secret échouent, mais les connexions utilisant d'autres secrets valides fonctionnent toujours.

   > **Note**
   >
   > Pour RDS for SQL Server, le proxy RDS nécessite un secret sensible aux majuscules et minuscules dans Secrets Manager, quels que soient les paramètres de classement des instances de base de données. Si votre application autorise des noms d'utilisateur avec des capitalisations différentes, tels que « Admin » et « admin », vous devez créer des secrets distincts pour chacun. Le proxy RDS ne prend pas en charge l'authentification insensible à la casse entre le client et le proxy.
   > Pour plus d'informations sur le classement SQL Server, consultez la documentation Microsoft SQL Server.

5. Pour Base de données, sélectionnez la base de données Amazon RDS à laquelle le secret doit accéder.
6. Renseignez les autres paramètres pour le secret, puis choisissez Stockage. Pour obtenir des instructions complètes, consultez Création d'un secret AWS Secrets Manager dans le AWS Secrets Manager Guide de l'utilisateur.

AWS CLI

Lorsque vous créez un proxy via le AWS CLI, vous spécifiez les Amazon Resource Names (ARNs) des secrets correspondants. Vous le faites pour tous les comptes utilisateur de base de données auxquels le proxy peut accéder. Dans le AWS Management Console, vous choisissez les secrets par leurs noms descriptifs.

- Pour créer un secret Secrets Manager à utiliser avec le proxy RDS, utilisez la commande create-secret :

```
aws secretsmanager create-secret \
  --name "secret_name" \
  --description "secret_description" \
  --region region_name \
  --secret-string '{"username":"db_user","password":"db_user_password"}'
```

- Vous pouvez également créer une clé personnalisée pour chiffrer votre secret Secrets Manager. La commande suivante crée un exemple de clé.

```
aws kms create-key --description "test-key" --policy '{
  "Id":"kms-policy",
  "Version": "2012-10-17",
  "Statement":
    [
      {
        "Sid":"Enable IAM User Permissions",
        "Effect":"Allow",
        "Principal":{"AWS":"arn:aws:iam::account_id:root"},
        "Action":"kms:*","Resource":"*"
      },
      {
        "Sid":"Allow access for Key Administrators",
        "Effect":"Allow",
        "Principal":
          {
            "AWS":
              ["$USER_ARN","arn:aws:iam:account_id::role/Admin"]
          },
        "Action":
          [
            "kms:Create*",
            "kms:Describe*",
            "kms:Enable*",
            "kms:List*",
            "kms:Put*",
            "kms:Update*",
            "kms:Revoke*",
            "kms:Disable*",
            "kms:Get*",
            "kms:Delete*",
            "kms:TagResource",
            "kms:UntagResource",
```

```
            "kms:ScheduleKeyDeletion",
            "kms:CancelKeyDeletion"
          ],
        "Resource":"*"
      },
      {
        "Sid":"Allow use of the key",
        "Effect":"Allow",
        "Principal":{"AWS":"$ROLE_ARN"},
        "Action":["kms:Decrypt","kms:DescribeKey"],
        "Resource":"*"
      }
    ]
}'
```

Par exemple, les commandes suivantes créent des secrets Secrets Manager pour deux utilisateurs de base de données.

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}'

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}'
```

Pour créer ces secrets chiffrés avec votre AWS KMS clé personnalisée, utilisez les commandes suivantes :

```
aws secretsmanager create-secret \
  --name secret_name_1 --description "db admin user" \
  --secret-string '{"username":"admin","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id

aws secretsmanager create-secret \
  --name secret_name_2 --description "application user" \
  --secret-string '{"username":"app-user","password":"choose_your_own_password"}' \
  --kms-key-id arn:aws:kms:us-east-2:account_id:key/key_id
```

Pour voir les secrets détenus par votre AWS compte, utilisez la commande list-secrets :

```
aws secretsmanager list-secrets
```

Lorsque vous créez un proxy à l'aide de la CLI, vous transmettez les Amazon Resource Names (ARNs) d'un ou de plusieurs secrets au `--auth` paramètre. L'exemple suivant montre comment préparer un rapport avec uniquement le nom et l'ARN de chacun des secrets détenus par votre compte AWS . Cet exemple utilise le `--output table` paramètre disponible dans la AWS CLI version 2. Si vous utilisez AWS CLI la version 1, utilisez `--output text` plutôt.

```
aws secretsmanager list-secrets --query '*[].[Name,ARN]' --output table
```

Pour vérifier que le secret contient les informations d'identification correctes au format approprié, utilisez la get-secret-valuecommande. Remplacez *your_secret_name* par le nom abrégé ou l'ARN du secret.

```
aws secretsmanager get-secret-value --secret-id your_secret_name
```

La sortie contient une ligne avec une valeur codée en JSON semblable à la suivante :

```
...
"SecretString": "{\"username\":\"your_username\",\"password\":\"your_password\"}",
...
```

Configuration de l'authentification IAM pour RDS Proxy

Pour configurer l'authentification Gestion des identités et des accès AWS (IAM) pour le proxy RDS dans Amazon RDS, créez et configurez une politique IAM qui accorde les autorisations nécessaires.

Cette rubrique décrit les étapes de configuration de l'authentification IAM pour RDS Proxy, notamment la création de la politique IAM requise et son association à un rôle IAM.

> **Tip**
>
> Cette procédure n'est nécessaire que si vous souhaitez créer votre propre rôle IAM. Dans le cas contraire, RDS peut créer automatiquement le rôle requis lorsque vous configurez le proxy. Vous pouvez donc ignorer ces étapes.

Conditions préalables

Avant de configurer l’authentification IAM pour RDS Proxy, assurez-vous de disposer des éléments suivants :

- AWS Secrets Manager : au moins un secret stocké contenant les informations d’identification de la base de données. Pour obtenir des instructions sur la création des secrets, consultez the section called “Configuration des informations d’identification de base de données”.

  Cela n'est pas obligatoire si vous utilisez l'authentification end-to-end IAM.
- Autorisations IAM : rôle ou utilisateur IAM autorisé à créer et à gérer des politiques, des rôles et des secrets IAM dans AWS Secrets Manager.

Création d'une politique IAM pour l'authentification end-to-end IAM

Lorsque vous utilisez l'authentification end-to-end IAM, le proxy RDS se connecte à votre base de données à l'aide de l'authentification IAM au lieu de récupérer les informations d'identification auprès de Secrets Manager. Cela nécessite de configurer votre rôle IAM avec des `rds-db:connect` autorisations pour les comptes de base de données que vous souhaitez utiliser avec le proxy.

Pour authentifier votre proxy RDS auprès de la base de données à l'aide d'IAM, créez un rôle IAM avec une politique qui accorde les autorisations de connexion à la base de données nécessaires.

Console

Pour créer un rôle pour l'authentification end-to-end IAM avec votre proxy

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Créez une stratégie d’autorisation pour le rôle. Pour connaître les étapes générales, consultez Créer des politiques IAM (console).

   Collez cette politique dans l’éditeur JSON et effectuez les modifications suivantes :

   - Indiquez votre propre ID de compte.
   - Remplacez `us-east-2` par le lieu où le mandataire doit résider.
   - Remplacez les noms d'utilisateur IDs et de ressource de base de données par ceux que vous souhaitez utiliser. Le format de l'ID de ressource diffère entre les instances RDS etAurora clusters.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds-db:connect",
            "Resource": [
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_1",
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_2"
            ]
        }
    ]
}
```

3. Créez le rôle et associez-le à une stratégie d'autorisations. Pour les étapes générales, voir Créer un rôle pour déléguer des autorisations à un AWS service.

   Choisissez Service AWS pour Type d'entité de confiance. Sous Cas d'utilisation, sélectionnez RDS et choisissez RDS : ajouter un rôle à la base de données pour le cas d'utilisation.

4. Sous Stratégies d'autorisation, choisissez la stratégie que vous avez créée.

5. Pour Sélectionner les entités de confiance, entrez la stratégie d'approbation suivante pour le rôle :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS CLI

Pour créer le rôle à l'aide duAWS CLI, envoyez la demande suivante :

```
aws iam create-role \
  --role-name my_e2e_iam_role_name \

  --assume-role-policy-document '{"Version":"2012-10-17",
     "Statement":[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

Associez ensuite cette stratégie au rôle :

```
aws iam put-role-policy \
  --role-name my_e2e_iam_role_name \
  --policy-name e2e_iam_db_connect_policy \
  --policy-document '{

    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds-db:connect",
            "Resource": [
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_1",
                "arn:aws:rds-db:us-
east-2:account_id:dbuser:db_instance_resource_id/db_user_name_2"
            ]
        }
    ]
}'
```

Le rôle et les autorisations IAM étant configurés pour l'authentification end-to-end IAM, vous pouvez désormais créer un proxy `DefaultAuthScheme` défini sur. `IAM_AUTH` Ce proxy s'authentifie directement auprès de la base de données à l'aide d'IAM sans avoir besoin des secrets de Secrets Manager. Pour obtenir des instructions, veuillez consulter the section called “Création d’un proxy”.

Lorsque vous utilisez l'authentification end-to-end IAM, assurez-vous que les utilisateurs de votre base de données sont configurés pour l'authentification IAM, comme décrit dans. Création d’un compte de base de données à l’aide de l’authentification IAM

Création d’une Politique IAM pour accéder à Secrets Manager

Pour permettre à RDS Proxy de récupérer les informations d’identification de la base de données de Secrets Manager, créez un rôle IAM avec une politique qui accorde les autorisations nécessaires.

Console

Pour créer un rôle permettant d’accéder aux secrets qui seront utilisés avec votre proxy

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Créez une stratégie d’autorisation pour le rôle. Pour connaître les étapes générales, consultez Créer des politiques IAM (console).

   Collez cette politique dans l’éditeur JSON et effectuez les modifications suivantes :

   - Indiquez votre propre ID de compte.
   - Remplacez `us-east-2` par la région où résidera le proxy.
   - Remplacez les noms secrets par ceux que vous avez créés. Pour plus d’informations, consultez Spécification de clés KMS dans les instructions de politique IAM.
   - Remplacez l’ID de clé KMS par celui que vous avez utilisé pour chiffrer les secrets de Secrets Manager, qu’il s’agisse de la clé par défaut ou de votre propre clé.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-
east-2:111122223333:secret:secret_name_1",
                "arn:aws:secretsmanager:us-
east-2:111122223333:secret:secret_name_2"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:111122223333:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-
east-2.amazonaws.com"
                }
            }
        }
    ]
}
```

3. Créez le rôle et associez-le à une stratégie d'autorisations. Pour les étapes générales, voir Créer un rôle pour déléguer des autorisations à un AWS service.

   Choisissez Service AWS pour Type d'entité de confiance. Sous Cas d'utilisation, sélectionnez RDS et choisissez RDS : ajouter un rôle à la base de données pour le cas d'utilisation.

4. Sous Stratégies d'autorisation, choisissez la stratégie que vous avez créée.

5. Pour Sélectionner les entités de confiance, entrez la stratégie d'approbation suivante pour le rôle :

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "rds.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

AWS CLI

Pour créer le rôle à l'aide duAWS CLI, envoyez la demande suivante :

```
aws iam create-role \
  --role-name my_role_name \
  --assume-role-policy-document '{"Version": "2012-10-17",
      "Statement":[{"Effect":"Allow","Principal":{"Service":
["rds.amazonaws.com"]},"Action":"sts:AssumeRole"}]}'
```

Associez ensuite cette stratégie au rôle :

```
aws iam put-role-policy \
  --role-name my_role_name \
  --policy-name secret_reader_policy \
  --policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": [
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_1",
                "arn:aws:secretsmanager:us-east-2:account_id:secret:secret_name_2"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "arn:aws:kms:us-east-2:account_id:key/key_id",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "secretsmanager.us-east-2.amazonaws.com"
                }
            }
        }
    ]
}'
```

Une fois le rôle et les autorisations IAM configurés, vous pouvez créer un proxy et l’associer à ce rôle. Cela permet au proxy de récupérer les informations d'identification de la base de données en toute

sécurité AWS Secrets Manager et d'activer l'authentification IAM pour vos applications. Pour obtenir des instructions, veuillez consulter the section called “Création d’un proxy”.

Création d’un proxy pour Amazon Aurora

Vous pouvez utiliser Proxy Amazon RDS pour améliorer la capacité de mise à l’échelle, la disponibilité et la sécurité de vos applications de base de données en regroupant les connexions et en gérant les basculements de base de données de manière plus efficace. Cette rubrique vous guide dans le processus de création d’un proxy. Avant de commencer, assurez-vous que votre base de données répond aux conditions requises, notamment celles qui concernent les autorisations IAM et la configuration VPC.

Vous pouvez employer un proxy avec un cluster de base de données Aurora MySQL ou Aurora PostgreSQL.

Console

Pour créer un proxy

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, sélectionnez Proxies.
3. Choisissez Création d’un proxy.
4. Configurez les paramètres suivants pour votre proxy.

Paramètre	Description
Famille de moteurs	Le protocole réseau de base de données que le proxy reconnaît lorsqu’il interprète le trafic réseau à destination et en provenance de la base de données. **Note** Pour utiliser Aurora PostgreSQL, assurez-vous de conserver la base de données `postgres` dans votre instance. Consultez Dépannage des problèmes liés à la base de données `postgres`.

Paramètre	Description
Identifiant du proxy	Un nom unique au sein de votre identifiant de AWS compte et de votre AWS région actuelle.
Délai d'inactivité de la connexion client	Le proxy ferme une connexion client si celle-ci reste inactive pendant une période définie. La valeur par défaut est de 1 800 secondes (30 minutes). Une connexion est considéré e comme inactive lorsque l'application ne soumet aucune nouvelle demande dans le délai défini après l'achèvement de la demande précédente. Le proxy maintient la connexion à la base de données sous-jacente ouverte et la renvoie au pool de connexions, la rendant ainsi disponible pour les nouvelles connexions client. Pour supprimer proactivement les connexions obsolètes, réduisez le délai d'expiration de la connexion client inactive. Pour minimiser les coûts de connexion en cas de pics de charge de travail, augmentez le délai d'expiration.
Base de données	Choisissez un cluster de bases de données Aurora pour y accéder via ce proxy. La liste inclut uniquement les instances et les clusters de bases de données dotés de moteurs de base de données, de versions de moteur et d'autres paramètres compatibles. Si la liste est vide, créez une instance ou un cluster de base de données compatible avec RDS Proxy. Pour ce faire, suivez la procédure décrite dans Création d'un cluster de bases de données Amazon Aurora. Puis, réessayez de créer le proxy.
Nombre maximal de connexions du pool de connexions	Une valeur comprise entre 1 et 100 pour définir le pourcenta ge de la limite `max_connections` que RDS Proxy peut utiliser. Si vous ne prévoyez d'utiliser qu'un seul proxy avec cette instance ou ce cluster de bases de données, définissez cette valeur sur 100. Pour plus d'informations sur l'utilisation de ce paramètre par RDS Proxy, consultez the section called "MaxConnectionsPercent".

Paramètre	Description
Filtres d’épinglage de session	Empêche RDS Proxy d’épingler certains états de session détectés, contournant ainsi les mécanismes de sécurité par défaut utilisés pour le multiplexage des connexions. Actuellement, PostgreSQL ne prend pas en charge ce paramètre, et la seule option disponible est `EXCLUDE_VARIABLE_SETS`. Son activation peut entraîner la propagation de variables de session d’une connexion à une autre, ce qui risque de provoquer des erreurs ou des problèmes de cohérence si les requêtes dépendent de variables de session définies en dehors de la transaction en cours. N’utilisez cette option qu’après avoir confirmé que vos applications peuvent partager des connexions de base de données en toute sécurité. Les modèles suivants sont considérés comme sûrs : • Instructions `SET` dans lesquelles aucune modification n’est apportée à la valeur effective de la variable de session. En d’autres termes, aucune modification n’est apportée à la variable de session. • Vous modifiez la valeur de la variable de session et exécutez une instruction dans la même transaction. Pour plus d’informations, consultez Contournement de l’épinglage d’un proxy RDS.
Délai d’expiration de l’emprunt de connexion	Si vous prévoyez que le proxy utilise toutes les connexions de base de données disponibles, définissez le délai d’attente avant qu’il ne renvoie une erreur de temporisation. Vous pouvez spécifier une durée maximale de cinq minutes. Ce paramètre s’applique uniquement lorsque le proxy a atteint le nombre maximal de connexions et que toutes les connexions sont déjà utilisées.

Paramètre	Description
Requête d’initialisation	(Facultatif) Ajoutez une requête d’initialisation ou modifiez la requête actuelle. Vous pouvez spécifier une ou plusieurs instructions SQL que le proxy doit exécuter lors de l’ouverture de chaque nouvelle connexion à la base de données. Ce paramètre est généralement utilisé avec des instructions SET pour s’assurer que chaque connexion a des paramètres identiques. Assurez-vous que la requête que vous ajoutez est valide. Pour inclure plusieurs variables dans une seule instruction SET, utilisez des virgules comme séparateurs. Par exemple : `SET variable1 =value1, variable2 =value2` Pour plusieurs instructions, utilisez des points-virgules comme séparateur. **Important** Comme la requête d’initialisation est accessible dans le cadre de la configuration du groupe cible, elle n’est pas protégée par des mécanismes d’authentification ni de chiffrement. Toute personne ayant accès à l’affichage ou à la gestion de la configuration du groupe cible proxy peut consulter la requête d’initialisation. Vous ne devez pas ajouter de données sensibles, telles que des mots de passe ou des clés de chiffrement à longue durée de vie, à cette option.
Gestion des identités et des accès AWS Rôle (IAM)	Un rôle IAM autorisé à accéder aux secrets de Secrets Manager, qui représentent les informations d’identification des comptes utilisateur de base de données que le proxy peut utiliser. Vous pouvez également créer un rôle IAM à l’aide de la AWS Management Console.

Paramètre	Description
Secrets Manager	Créez ou choisissez des secrets Secrets Manager représentant les informations d'identification des comptes des utilisateurs de base de données autorisés à utiliser le proxy. Lorsque le schéma d'authentification par défaut est défini sur Aucun, ce champ est obligatoire. Lorsque le schéma d'authent ification par défaut est défini sur l'authentification IAM, ce champ devient facultatif et est marqué comme tel dans la console. Vous pouvez choisir un ou plusieurs secrets dans le menu déroulant ou en créer un nouveau à l'aide du lien Créer un nouveau secret.
Type d’authentification client	Le type d’authentification utilisé par le proxy pour les connexion s à partir des clients. Votre choix s’applique à tous les secrets de Secrets Manager que vous associez à ce proxy. Si vous devez spécifier un type d'authentification client différent pour chaque secret, créez votre proxy en utilisant plutôt l'API AWS CLI ou l'API. Spécifiez cette option uniquement lorsque votre connexion client utilise les informations d'identification de base de données pour l'authentification.
Authentification IAM	Spécifiez Obligatoire ou Non autorisé pour l'authentification IAM pour les connexions à votre proxy. Votre choix s’applique à tous les secrets de Secrets Manager que vous associez à ce proxy. Si vous devez spécifier une authentification IAM différente pour chaque secret, créez votre proxy en utilisant plutôt l'API AWS CLI ou l'API.

Paramètre	Description
Schéma d'authentification par défaut	Choisissez le type d'authentification par défaut que le proxy utilise pour les connexions client au proxy et les connexions entre le proxy et la base de données sous-jacente. Vous avez les options suivantes : • Aucun (par défaut) : le proxy récupère les informations d'identification de la base de données à partir des secrets de Secrets Manager. • Authentification IAM : le proxy utilise l'authentification IAM pour se connecter à la base de données, ce qui permet l'authentification end-to-end IAM. Lorsque vous sélectionnez l'authentification IAM, une alerte d'information apparaît pour vous rappeler d'activer l'authent ification de base de données IAM pour les bases de données de la configuration du groupe cible. **Note** Cette option est prise en charge uniquement pour les familles de moteurs MySQL, PostgreSQL et MariaDB.

Paramètre	Description
Comptes de base de données pour l'authentification IAM	Ce champ apparaît uniquement lorsque le schéma d'authentification par défaut est défini sur Authentification IAM et que le rôle de gestion des identités et des accès (IAM) est défini sur Créer un rôle IAM. Nommez les comptes utilisateur de base de données pour le proxy à utiliser avec l'authentification IAM. Ce champ est obligatoire. Spécifiez plusieurs comptes en : • Tapez un nom d'utilisateur de base de données pour l'ajouter en tant que balise • Utilisation de noms d'utilisateur de base de données spécifiques (par exemple`db_user`,,`jane_doe`) • Utilisation de modèles de caractères génériques pour plusieurs utilisateurs (par exemple,`db_test_*`) Chaque compte apparaît sous la forme d'une étiquette amovible que vous pouvez supprimer en cliquant sur l'icône X. La console utilise ces valeurs pour créer les `rds-db:connect` autorisations appropriées dans la politique de rôle IAM.
Exigez la sécurité de la couche de transport	S'applique TLS/SSL à toutes les connexions client. Le proxy utilise le même paramètre de chiffrement pour sa connexion à la base de données sous-jacente, que la connexion du client soit chiffrée ou non.

Paramètre	Description
Type de réseau de connexion cible	Version IP utilisée par le proxy pour se connecter à la base de données cible. Sélectionnez parmi les options suivantes : • IPv4— Le proxy se connecte à la base de données à l'aide d' IPv4 adresses. • IPv6— Le proxy se connecte à la base de données à l'aide d' IPv6 adresses. La valeur par défaut est IPv4. Pour pouvoir être utilisée IPv6, votre base de données doit prendre en charge le mode double pile. Le mode double pile n’est pas disponible pour les connexions cibles.
Type de réseau de point de terminaison	Version IP du point de terminaison de proxy que les clients utilisent pour se connecter au proxy. Sélectionnez parmi les options suivantes : • IPv4— Le point de terminaison du proxy utilise uniquement IPv4 des adresses. • IPv6— Le point de terminaison du proxy utilise uniquement IPv6 des adresses. • Double pile : le point de terminaison du proxy prend en charge à la fois les IPv6 adresses IPv4 et les adresses. La valeur par défaut est IPv4. Pour l'utiliser IPv6 ou le double-st ack, votre VPC et vos sous-réseaux doivent être configurés pour prendre en charge le type de réseau sélectionné.
Sous-réseaux	Ce champ est pré-rempli avec tous les sous-réseaux associés à votre VPC. Vous pouvez supprimer tous les sous-réseaux dont le proxy n’a pas besoin, en veillant toutefois à en conserver au moins deux. Pour les IPv6 types de réseaux de points de terminaison à double pile, assurez-vous que les sous-réseaux sélectionnés prennent en charge le type de réseau choisi.

Paramètre	Description
Groupe de sécurité VPC	Choisissez un groupe de sécurité de VPC existant ou créez-en un à l'aide de la AWS Management Console. Configurez les règles entrantes pour permettre à vos applications d'accéder au proxy, et les règles sortantes pour autoriser le trafic en provenance de vos bases de données cibles. **Note** Le groupe de sécurité doit autoriser les connexions du proxy à la base de données. Il sert à la fois pour le trafic entrant depuis vos applications vers le proxy et pour le trafic sortant du proxy vers la base de données. Par exemple, si vous utilisez le même groupe de sécurité pour la base de données et le proxy, assurez-vous que les ressources de ce groupe de sécurité peuvent communiquer entre elles. Lorsque vous utilisez un VPC partagé, évitez d'utilise r le groupe de sécurité par défaut pour le VPC ou un groupe associé à un autre compte. Choisissez plutôt un groupe de sécurité qui appartient à votre compte. Si aucun n'existe, créez-en un. Pour plus d'informations, voir Travailler avec le partage VPCs. RDS déploie un proxy sur plusieurs zones de disponibilité pour assurer une haute disponibilité. Pour activer la communication entre les zones de disponibilité, la liste de contrôle d'accès (ACL) réseau du sous-réseau de votre proxy doit autoriser le trafic sortant propre aux ports du moteur et autoriser le trafic entrant sur tous les ports. Pour plus d'informations sur le réseau ACLs, voir Contrôler le trafic vers les sous-réseaux à l'aide du réseau ACLs. Si votre proxy et votre cible ont la même liste ACL réseau, vous devez ajouter une règle de trafic entrant de protocole TCP dans laquelle la source est définie sur le CIDR du VPC. Vous devez également ajouter une règle de trafic sortant

Paramètre	Description
	de protocole TCP propre aux ports du moteur dans laquelle la destination est définie sur le CIDR du VPC.
Activation de la journalisation améliorée	Activez ce paramètre pour résoudre les problèmes liés à la compatibilité des proxies ou aux performances. Lorsqu'il est activé, RDS Proxy consigne des informations détaillées sur les performances afin de vous aider à diagnostiquer le comportem ent SQL ou les performances et la capacité de mise à l'échelle des connexions via le proxy. Activez ce paramètre uniquement pour le débogage et assurez-vous que les mesures de sécurité appropriées sont appliquée s pour protéger les informations sensibles contenues dans les journaux. Pour minimiser les frais, RDS Proxy désactive automatiquement ce paramètre 24 heures après son activatio n. Utilisez-le temporairement pour résoudre des problèmes spécifiques.

5. Choisissez Création d'un proxy.

AWS CLI

Pour créer un proxy à l'aide de AWS CLI, appelez la create-db-proxycommande avec les paramètres obligatoires suivants :

- `--db-proxy-name`
- `--engine-family`
- `--role-arn`
- `--vpc-subnet-ids`

La valeur `--engine-family` est sensible à la casse.

Example

Pour Linux, macOS ou Unix :

```
aws rds create-db-proxy \
```

```
    --db-proxy-name proxy_name \
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } \
    --role-arn iam_role \
    --vpc-subnet-ids space_separated_list \
    [--default-auth-scheme { NONE | IAM_AUTH }] \
    [--auth ProxyAuthenticationConfig_JSON_string] \
    [--vpc-security-group-ids space_separated_list] \
    [--require-tls | --no-require-tls] \
    [--idle-client-timeout value] \
    [--debug-logging | --no-debug-logging] \
    [--endpoint-network-type { IPV4 | IPV6 | DUAL }] \
    [--target-connection-network-type { IPV4 | IPV6 }] \
    [--tags comma_separated_list]
```

Pour Windows :

```
aws rds create-db-proxy ^
    --db-proxy-name proxy_name ^
    --engine-family { MYSQL | POSTGRESQL | SQLSERVER } ^
    --role-arn iam_role ^
    --vpc-subnet-ids space_separated_list ^
    [--default-auth-scheme { NONE | IAM_AUTH }] ^
    [--auth ProxyAuthenticationConfig_JSON_string] ^
    [--vpc-security-group-ids space_separated_list] ^
    [--require-tls | --no-require-tls] ^
    [--idle-client-timeout value] ^
    [--debug-logging | --no-debug-logging] ^
    [--endpoint-network-type { IPV4 | IPV6 | DUAL }] ^
    [--target-connection-network-type { IPV4 | IPV6 }] ^
    [--tags comma_separated_list]
```

Voici un exemple de valeur JSON pour l'option `--auth`. Cet exemple applique un type d'authentification client différent à chaque secret.

```
[
  {
    "Description": "proxy description 1",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789123:secret/1234abcd-12ab-34cd-56ef-1234567890ab",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_SCRAM_SHA_256"
  },
```

```
  {
    "Description": "proxy description 2",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122223333:secret/1234abcd-12ab-34cd-56ef-1234567890cd",
    "IAMAuth": "DISABLED",
    "ClientPasswordAuthType": "POSTGRES_MD5"

  },

  {
    "Description": "proxy description 3",
    "AuthScheme": "SECRETS",
    "SecretArn": "arn:aws:secretsmanager:us-
west-2:111122221111:secret/1234abcd-12ab-34cd-56ef-1234567890ef",
    "IAMAuth": "REQUIRED"
  }

]
```

Le paramètre `--endpoint-network-type` indique la version IP du point de terminaison de proxy que les clients utilisent pour se connecter au proxy. Les valeurs valides sont :

- `IPV4`— Le point de terminaison du proxy utilise uniquement IPv4 des adresses (par défaut).
- `IPV6`— Le point de terminaison du proxy utilise uniquement IPv6 des adresses.
- `DUAL`— Le point de terminaison du proxy prend en charge IPv4 les deux IPv6 adresses.

Le paramètre `--target-connection-network-type` indique la version IP utilisée par le proxy pour se connecter à la base de données cible. Les valeurs valides sont :

- `IPV4`— Le proxy se connecte à la base de données à l'aide d' IPv4 adresses (par défaut).
- `IPV6`— Le proxy se connecte à la base de données à l'aide d' IPv6 adresses.

Pour utiliser IPv6 ou doubler les types de réseaux de points de terminaison, votre VPC et vos sous-réseaux doivent être configurés pour prendre en charge le type de réseau sélectionné. Pour utiliser le type de réseau de connexion IPv6 cible, votre base de données doit prendre en charge le mode double pile.

> **Tip**
>
> Si vous ne connaissez pas encore le sous-réseau IDs à utiliser pour le `--vpc-subnet-ids` paramètre, consultez des exemples Configuration des prérequis réseau pour un proxy RDS pour les trouver.

> **Note**
>
> Le groupe de sécurité doit autoriser l'accès à la base de données à laquelle le proxy se connecte. Le même groupe de sécurité est utilisé pour l'entrée de vos applications vers le proxy, et pour la sortie du proxy vers la base de données. Par exemple, supposons que vous utilisiez le même groupe de sécurité pour votre base de données et votre proxy. Dans ce cas, assurez-vous de spécifier que les ressources de ce groupe de sécurité peuvent communiquer avec d'autres ressources du même groupe de sécurité.
> Lorsque vous utilisez un VPC partagé, vous ne pouvez pas utiliser le groupe de sécurité par défaut pour le VPC ni un groupe appartenant à un autre compte. Choisissez un groupe de sécurité qui appartient à votre compte. S'il n'en existe aucun, créez-en un. Pour plus d'informations sur cette limitation, voir Travailler avec le partage VPCs.

Pour créer les associations appropriées pour le proxy, vous devez également utiliser la register-db-proxy-targetscommande. Spécifiez le nom du groupe cible `default`. RDS Proxy crée automatiquement un groupe cible portant ce nom au moment de la création de chaque proxy.

```
aws rds register-db-proxy-targets
    --db-proxy-name value
    [--target-group-name target_group_name]
    [--db-instance-identifiers space_separated_list]  # rds db instances, or
    [--db-cluster-identifiers cluster_id]        # rds db cluster (all instances)
```

API RDS

Pour créer un proxy RDS, appelez l'opération Create de l'API Amazon RDS. DBProxy Vous transmettez un paramètre avec la structure AuthConfigde données.

RDS Proxy crée automatiquement un groupe cible nommé `default` au moment de la création de chaque proxy. Vous associez un cluster de base de données Aurora d'une instance de base au groupe cible en appelant la fonction Register DBProxy Targets.

> **Important**
>
> Lorsque vous sélectionnez l'authentification IAM comme schéma d'authentification par défaut :
>
> - Vous devez activer l'authentification de base de données IAM sur vos instances ou clusters de base de données cibles pour que le proxy puisse se connecter correctement.
> - Si vous choisissez Créer un rôle IAM, le champ Comptes de base de données pour l'authentification IAM est obligatoire.
> - Si vous sélectionnez un rôle IAM existant, la console ne met pas automatiquement à jour le rôle avec les autorisations de connexion à la base de données. Vérifiez que le rôle dispose des `rds-db:connect` autorisations nécessaires.

Affichage d’un proxy

Après avoir créé un ou plusieurs proxys RDS, vous pouvez les afficher et les gérer dans la AWS Management Console, l’AWS CLI ou l’API RDS. Vous pouvez consulter les détails de leur configuration, surveiller les performances et déterminer les proxys à modifier ou à supprimer selon les besoins.

Pour permettre aux applications de base de données d’acheminer le trafic via un proxy, vous devez spécifier le point de terminaison du proxy dans la chaîne de connexion.

Console

Pour afficher un proxy dans la console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, sélectionnez Proxies.
3. Sélectionnez le nom du proxy pour en afficher les détails.
4. Sur la page de détails, la section Groupes cibles montre comment le proxy est lié à un cluster de bases de données Aurora spécifique. Vous pouvez accéder à la page du groupe cible par défaut pour obtenir une vue plus approfondie de cette association, notamment des paramètres de configuration définis lors de la création du proxy. Cela inclut le pourcentage maximal de connexion, le délai d’emprunt de connexion, la famille de moteurs et les filtres d’épinglage de session.

Interface de ligne de commande (CLI)

Pour afficher votre proxy à l'aide de l'interface de ligne de commande, utilisez la commande describe-db-proxies. Par défaut, la demande renvoie tous les proxy appartenant à votre compte AWS. Pour afficher les détails d'un proxy, spécifiez son nom avec le paramètre `--db-proxy-name`.

```
aws rds describe-db-proxies [--db-proxy-name proxy_name]
```

Pour afficher les autres informations associées au proxy, utilisez les commandes suivantes.

```
aws rds describe-db-proxy-target-groups  --db-proxy-name proxy_name

aws rds describe-db-proxy-targets --db-proxy-name proxy_name
```

Utilisez la séquence de commandes suivante pour afficher plus de détails sur les éléments associés au proxy :

1. Pour obtenir une liste des proxies, exécutez describe-db-proxies.
2. Pour afficher les paramètres de connexion tels que le pourcentage maximal de connexions que le proxy peut utiliser, exécutez describe-db-proxy-target-groups `--db-proxy-name`. Utilisez le nom du proxy comme valeur de paramètre.
3. Pour afficher les détails du cluster de bases de données associé au groupe cible renvoyé, exécutez describe-db-proxy-targets.

API RDS

Pour afficher vos proxies à l'aide de l'API RDS, utilisez l'opération DescribeDBProxies. Elle renvoie les valeurs du type de données DBProxy.

Pour afficher les détails des paramètres de connexion du proxy, utilisez les identifiants de proxy à partir de la valeur renvoyée à l'aide de l'opération DescribeDBProxyTargetGroups. Elle renvoie les valeurs du type de données DBProxyTargetGroup.

Pour voir l'instance RDS ou le cluster de bases de données Aurora associé au proxy, utilisez l'opération DescribeDBProxyTargets. Elle renvoie les valeurs du type de données DBProxyTarget.

Connexion à une base de données via RDS Proxy

Votre connexion à un cluster ou à un cluster de base de données Aurora qui utilise Aurora Serverless v2 via un proxy est généralement identique à une connexion directe à la base de données. La principale différence est que vous indiquez le point de terminaison du proxy et non celui du cluster. Par défaut, toutes les connexions proxy sont read/write fonctionnelles et utilisent l'instance Writer. Si vous utilisez généralement le point de terminaison du lecteur pour les connexions en lecture seule, vous pouvez créer un point de terminaison supplémentaire en lecture seule pour le proxy. Vous pouvez utiliser ce point de terminaison de la même manière. Pour de plus amples informations, veuillez consulter Présentation des points de terminaison proxy.

Rubriques

- Connexion à une base de données à l'aide des informations d'identification
- Connexion à une base de données à l'aide de l'authentification IAM
- Considérations relatives à la connexion à PostgreSQL

Connexion à une base de données à l'aide des informations d'identification

Procédez comme suit pour vous connecter à un proxy à l'aide des informations d'identification de base de données :

1. Recherchez le point de terminaison du proxy. Dans le AWS Management Console, vous pouvez trouver le point de terminaison sur la page de détails du proxy correspondant. Avec le AWS CLI, vous pouvez utiliser la describe-db-proxiescommande. L'exemple suivant montre comment procéder.

```
# Add --output text to get output as a simple tab-separated list.
$ aws rds describe-db-proxies --query '*[*].
{DBProxyName:DBProxyName,Endpoint:Endpoint}'
[
    [
        {
            "Endpoint": "the-proxy.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy"
        },
        {
            "Endpoint": "the-proxy-other-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-other-secret"
```

```
        },
        {
            "Endpoint": "the-proxy-rds-secret.proxy-demo.us-
east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-rds-secret"
        },
        {
            "Endpoint": "the-proxy-t3.proxy-demo.us-east-1.rds.amazonaws.com",
            "DBProxyName": "the-proxy-t3"
        }
    ]
]
```

2. Spécifier le point de terminaison comme paramètre hôte dans la chaîne de connexion de votre application cliente. Par exemple, spécifiez le point de terminaison du proxy comme valeur pour l'option `mysql -h` ou `psql -h`.
3. Fournissez le nom d'utilisateur et le mot de passe de base de données que vous utilisez habituellement.

Connexion à une base de données à l'aide de l'authentification IAM

Lorsque vous utilisez l'authentification IAM avec RDS Proxy, vous disposez de deux options d'authentification entre votre client et votre proxy :

- Configurez les utilisateurs de votre base de données pour qu'ils s'authentifient à l'aide de noms d'utilisateur et de mots de passe habituels. RDS Proxy récupère le nom d'utilisateur et les informations d'identification du mot de passe auprès de Secrets Manager. La connexion depuis RDS Proxy à la base de données sous-jacente ne passe pas par IAM.
- Vous pouvez également utiliser l'authentification end-to-end IAM, qui se connecte à votre base de données via le proxy à l'aide d'IAM sans avoir besoin d'informations d'identification de base de données.

Pour vous connecter à RDS Proxy à l'aide de l'authentification IAM, suivez la même procédure de connexion générale que pour vous connecter avec une authentification IAM à un cluster de bases de données Aurora. Pour obtenir des informations générales sur l'utilisation d'IAM, consultez Sécurité dans Amazon Aurora. Si vous utilisez l'authentification end-to-end IAM, fournissez le plug-in d'authentification IAM à votre utilisateur de base de données. Consultez Création d'un compte de base de données à l'aide de l'authentification IAM.

Les principales différences dans l'utilisation d'IAM pour RDS Proxy sont les suivantes :

- Avec l'authentification IAM standard, les utilisateurs de la base de données disposent d'informations d'identification régulières dans la base de données. Vous configurez des secrets Secrets Manager contenant ces noms et mots de passe d'utilisateur, et autorisez RDS Proxy à récupérer les informations d'identification à partir d'Secrets Manager. L'authentification IAM s'applique à la connexion entre votre programme client et le proxy. Le proxy s'authentifie ensuite à la base de données à l'aide des informations d'identification (nom d'utilisateur et mot de passe) extraites via Secrets Manager.
- Avec l'authentification end-to-end IAM, vous n'avez pas besoin de configurer les secrets de Secrets Manager pour les informations d'identification de base de données. L'authentification IAM s'applique à la connexion entre le client et le proxy et la base de données.
- Spécifiez le point de terminaison du proxy plutôt que celui de l'instance, du cluster ou du lecteur. Pour plus d'informations sur le point de terminaison du proxy, consultez Connexion à votre cluster de base de données à l'aide de l'authentification IAM..
- Veillez à utiliser le protocole TLS (Transport Layer Security)/SSL (Secure Sockets Layer) lorsque vous vous connectez à un proxy avec l'authentification IAM.

Vous pouvez accorder l'accès au proxy à un utilisateur spécifique en modifiant la politique IAM. Un exemple suit.

```
"Resource": "arn:aws:rds-db:us-east-2:1234567890:dbuser:prx-ABCDEFGHIJKL01234/db_user"
```

> **Tip**
>
> Lorsque vous configurez l'authentification IAM pour les connexions au proxy RDS, suivez ces directives importantes pour éviter les problèmes de connexion :
>
> - N'accordez pas le `rds_iam` rôle tout en conservant l'authentification générale par mot de passe pour le même utilisateur ou le même rôle de base de données.
> - N'oubliez pas que lorsque les clients se connectent au proxy RDS à l'aide de l'authentification IAM, le proxy RDS se connecte toujours à la base de données à l'aide de l'authentification par mot de passe via Secrets Manager.
> - Si vous rencontrez fréquemment des interruptions et des reconnexions de connexion, supprimez toutes les autorisations existantes `rds_iam` accordées à l'utilisateur ou au rôle et utilisez uniquement l'authentification par mot de passe.

- Assurez-vous que votre politique de mot de passe répond aux exigences du protocole SCRAM-SHA-256 en matière de caractères sûrs.

La combinaison de méthodes d'authentification IAM et par mot de passe pour le même utilisateur de base de données peut entraîner une instabilité de connexion.

Considérations relatives à la connexion à PostgreSQL

Si vous créez un nouvel utilisateur de base de données PostgreSQL pour vous connecter au proxy RDS, veillez à lui accorder `CONNECT` sur la base de données. À défaut de ce privilège, l'utilisateur ne sera pas en mesure de se connecter. Pour plus d'informations, consultez the section called "Ajout d'un nouvel utilisateur de base de données à une base de données PostgreSQL lors de l'utilisation du proxy RDS".

Lorsqu'un client démarre une connexion à une base de données PostgreSQL, il envoie un message de démarrage. Ce message inclut des paires de chaînes de noms de paramètres et de valeurs. Pour plus de détails, consultez `StartupMessage` dans la section relative aux formats de message PostgreSQL de la documentation PostgreSQL.

Lors de la connexion via un proxy RDS, le message de démarrage peut inclure les paramètres actuellement reconnus suivants :

- `user`
- `database`

Le message de démarrage peut également inclure les paramètres d'exécution supplémentaires suivants :

- `application_name`
- `client_encoding`
- `DateStyle`
- `TimeZone`
- `extra_float_digits`
- `search_path`

Pour plus d'informations sur la messagerie PostgreSQL, consultez protocole frontend/backend dans la documentation PostgreSQL.

Pour PostgreSQL, si vous utilisez JDBC, nous vous recommandons ce qui suit pour éviter l'épinglage :

- Définissez le paramètre de connexion JDBC `assumeMinServerVersion` sur `9.0` au minimum afin d'éviter l'épinglage. Cela empêche le pilote JDBC d'effectuer un aller-retour supplémentaire au démarrage de la connexion lorsqu'il exécute `SET extra_float_digits = 3`.
- Définissez le paramètre de connexion JDBC `ApplicationName` sur *`any/your-application-name`* afin d'éviter l'épinglage. Cela empêche le pilote JDBC d'effectuer un aller-retour supplémentaire au démarrage de la connexion lorsqu'il exécute `SET application_name = "PostgreSQL JDBC Driver"`. Notez que le paramètre JDBC est `ApplicationName`, mais que le paramètre PostgreSQL `StartupMessage` est `application_name`.

Pour plus d'informations, consultez Contournement de l'épinglage d'un proxy RDS. Pour plus d'informations sur la connexion à l'aide de JDBC, consultez Connexion à la base de données dans la documentation PostgreSQL.

Gestion d'un RDS Proxy

Cette section fournit des informations sur la gestion du fonctionnement et de la configuration de RDS Proxy. Ces procédures aident votre application à utiliser de manière optimale les connexions de base de données et à obtenir une réutilisation maximale des connexions. Plus vous tirez profit de la réutilisation des connexions, plus vous évitez une surcharge de l'UC et de la mémoire. Cela réduit la latence de votre application et permet à la base de données de dédier une plus grande partie de ses ressources au traitement des requêtes d'application.

Rubriques

- Modification d'un RDS Proxy
- Ajout d'un nouvel utilisateur de base de données lors de l'utilisation du proxy RDS
- Passage de l'authentification IAM standard à l'authentification end-to-end IAM pour le proxy RDS
- Considérations relatives à la connexion RDS Proxy
- Contournement de l'épinglage d'un proxy RDS
- Suppression d'un RDS Proxy

Modification d’un RDS Proxy

Vous pouvez modifier des paramètres spécifiques associés à un proxy après sa création. Pour ce faire, modifiez le proxy lui-même, son groupe cible associé, ou les deux. Chaque proxy dispose d’un groupe cible associé.

AWS Management Console

> **Important**
>
> Les valeurs des champs Client authentication type (Type d’authentification client) et IAM authentication (Authentification IAM) s’appliquent à tous les secrets de Secrets Manager associés à ce proxy. Pour spécifier des valeurs différentes pour chaque secret, modifiez votre proxy en utilisant plutôt l'API AWS CLI ou l'API.

Modifications des paramètres d’un proxy

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, sélectionnez Proxies.
3. Dans la liste de proxy, choisissez celui dont vous souhaitez modifier les paramètres ou accédez à sa page de détails.
4. Pour Actions, choisissez Modifier.
5. Saisissez ou sélectionnez les propriétés à modifier. Vous pouvez modifier les valeurs suivantes :

   - Identifiant du proxy : renommez le proxy en saisissant un nouvel identifiant.
   - Délai d’inactivité de la connexion client – Saisissez une période pour le délai d’inactivité de la connexion client.
   - Rôle IAM – Modifiez le rôle IAM utilisé pour récupérer les secrets de Secrets Manager.

     > **Note**
     >
     > Vous ne pouvez pas créer de nouveau rôle IAM si vous définissez le schéma d'authentification par défaut sur Authentification IAM.

   - Secrets de Secrets Manager – Ajoutez ou supprimez des secrets Secrets Manager. Ces secrets correspondent aux noms d’utilisateur et mots de passe de la base de données.

- Type d'authentification client : modifiez le type d'authentification pour les connexions client au proxy.
- IAM authentication (Authentification IAM) : exigez ou désactivez l’authentification IAM pour les connexions au proxy.
- Schéma d'authentification par défaut : modifiez le schéma d'authentification par défaut utilisé par le proxy pour les connexions client au proxy et les connexions entre le proxy et la base de données sous-jacente.
- Exiger la Sécurité de la couche transport – Activez ou désactivez l’exigence du protocole TLS (Transport Layer Security).
- Groupe de sécurité de VPC – Ajoutez ou supprimez des groupes de sécurité de VPC que le proxy doit utiliser.
- Activation de la journalisation améliorée – Activez ou désactivez la journalisation améliorée.

6. Sélectionnez Modify.

Si vous n’avez pas trouvé les paramètres répertoriés que vous souhaitez modifier, procédez comme suit pour mettre à jour le groupe cible du proxy. Le groupe cible associé à un proxy contrôle les paramètres liés aux connexions à la base de données physique. Chaque proxy dispose d’un groupe cible associé, nommé `default`, qui est créé automatiquement avec le proxy. Vous ne pouvez pas renommer le groupe cible par défaut.

Vous pouvez uniquement modifier le groupe cible à partir de la page de détails du proxy, et non depuis la liste de la page Proxies.

Modification des paramètres d’un groupe cible proxy

1. À partir de la page Proxies, accédez à la page des détails d’un proxy.
2. Pour les Groupes cibles, choisissez le lien `default`. Actuellement, tous les proxy ont un groupe cible unique nommé `default`.
3. Sur la page de détails du groupe cible par défaut, sélectionnez Modifier.
4. Définissez de nouveaux paramètres pour les propriétés que vous pouvez modifier :

   - Base de données : choisissez un autre cluster Aurora.
   - Nombre maximal de connexions dans le groupe de connexions – Ajustez le pourcentage du nombre de connexions maximum disponibles que le proxy peut utiliser.

- Filtre d'épinglage de session – (Facultatif) Choisissez un filtre d'épinglage de session. Cela permet de contourner les mesures de sécurité par défaut pour le multiplexage des connexions de base de données entre les connexions client. Actuellement, le paramètre n'est pas pris en charge pour PostgreSQL. Le seul choix est `EXCLUDE_VARIABLE_SETS`.

  Avec l'activation de ce paramètre, les variables de session d'une connexion peuvent avoir un impact sur d'autres connexions. Cela peut entraîner des erreurs ou des problèmes d'exactitude si vos requêtes dépendent de valeurs de variables de session définies en dehors de la transaction en cours. Vous pouvez utiliser cette option après avoir vérifié que vos applications peuvent partager des connexions de base de données en toute sécurité entre les connexions client.

  Les modèles suivants peuvent être considérés comme sûrs :

  - Instructions `SET` dans lesquelles aucune modification n'est apportée à la valeur effective de la variable de session, c'est-à-dire qu'aucune modification n'est apportée à la variable de session.
  - Vous modifiez la valeur de la variable de session et exécutez une instruction dans la même transaction.

  Pour plus d'informations, consultez Contournement de l'épinglage d'un proxy RDS.
- Délai d'expiration d'emprunt de connexion – Ajustez l'intervalle du délai d'attente d'emprunt de connexion. Ce paramètre s'applique lorsque le nombre maximal de connexions est déjà utilisé pour le proxy. Ce paramètre permet de définir combien de temps le proxy doit attendre la disponibilité d'une connexion avant de renvoyer une erreur de dépassement de délai d'attente.
- Requête d'initialisation. (Facultatif) Ajoutez une requête d'initialisation ou modifiez la requête actuelle. Vous pouvez spécifier une ou plusieurs instructions SQL que le proxy doit exécuter lors de l'ouverture de chaque nouvelle connexion à la base de données. Ce paramètre est généralement utilisé avec des instructions `SET` pour s'assurer que chaque connexion a des paramètres identiques. Assurez-vous que la requête que vous ajoutez est valide. Pour inclure plusieurs variables dans une seule instruction `SET`, utilisez des virgules comme séparateurs. Par exemple :

  ```
  SET variable1=value1, variable2=value2
  ```

  Pour plusieurs instructions, utilisez des points-virgules comme séparateur.

Certaines propriétés, telles que l'identifiant du groupe cible et le moteur de base de données, sont corrigées.

5. Sélectionnez Modification du groupe cible.

AWS CLI

Pour modifier un proxy à l'aide de AWS CLI, utilisez les commandes modify-db-proxymodify-db-proxy-target-group deregister-db-proxy-targetset register-db-proxy-targets.

Avec la commande `modify-db-proxy`, vous pouvez modifier des propriétés, par exemple :

- Ensemble des secrets Secrets Manager utilisés par le proxy.
- TLS requis ou non.
- Délai d'expiration de la connexion client inactive.
- Nécessité ou non de consigner des informations supplémentaires des instructions SQL pour le débogage.
- Rôle IAM utilisé pour récupérer les secrets Secrets Manager.
- Groupes de sécurité utilisés par le proxy.
- Schéma d'authentification par défaut associé au proxy.

L'exemple suivant montre comment renommer un proxy existant.

```
aws rds modify-db-proxy --db-proxy-name the-proxy --new-db-proxy-name the_new_name
```

Pour modifier les paramètres liés à la connexion ou renommer le groupe cible, utilisez la commande `modify-db-proxy-target-group`. Actuellement, tous les proxy ont un groupe cible unique nommé `default`. Lorsque vous travaillez avec ce groupe cible, vous indiquez le nom du proxy et `default` pour le nom du groupe cible. Vous ne pouvez pas renommer le groupe cible par défaut.

L'exemple suivant montre comment vérifier le paramètre `MaxIdleConnectionsPercent` d'un proxy, puis le modifier à l'aide du groupe cible.

```
aws rds describe-db-proxy-target-groups --db-proxy-name the-proxy

{
    "TargetGroups": [
```

```
        {
            "Status": "available",
            "UpdatedDate": "2019-11-30T16:49:30.342Z",
            "ConnectionPoolConfig": {
                "MaxIdleConnectionsPercent": 50,
                "ConnectionBorrowTimeout": 120,
                "MaxConnectionsPercent": 100,
                "SessionPinningFilters": []
            },
            "TargetGroupName": "default",
            "CreatedDate": "2019-11-30T16:49:27.940Z",
            "DBProxyName": "the-proxy",
            "IsDefault": true
        }
    ]
}

aws rds modify-db-proxy-target-group --db-proxy-name the-proxy --target-group-name
 default --connection-pool-config '
{ "MaxIdleConnectionsPercent": 75 }'

{
    "DBProxyTargetGroup": {
        "Status": "available",
        "UpdatedDate": "2019-12-02T04:09:50.420Z",
        "ConnectionPoolConfig": {
            "MaxIdleConnectionsPercent": 75,
            "ConnectionBorrowTimeout": 120,
            "MaxConnectionsPercent": 100,
            "SessionPinningFilters": []
        },
        "TargetGroupName": "default",
        "CreatedDate": "2019-11-30T16:49:27.940Z",
        "DBProxyName": "the-proxy",
        "IsDefault": true
    }
}
```

Grâce aux commandes `deregister-db-proxy-targets` et `register-db-proxy-targets`, vous modifiez les clusters de base de données Aurora auxquels le proxy est associé via son groupe cible. Actuellement, chaque proxy peut se connecter à un cluster de bases de données Aurora. Le groupe cible suit les détails de connexion de toutes les instances de base de données dans un cluster Aurora.

L'exemple suivant commence par un proxy associé à un cluster Aurora MySQL nommé `cluster-56-2020-02-25-1399`. L'exemple vous explique comment modifier le proxy afin qu'il puisse se connecter à un autre cluster nommé `provisioned-cluster`.

Lorsque vous travaillez avec un cluster de bases de données Aurora, sélectionnez l'option `--db-cluster-identifier`.

L'exemple suivant modifie un proxy Aurora MySQL. Un proxy PostgreSQL Aurora dispose du port 5432.

```
aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": [
        {
            "Endpoint": "instance-9814.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-9814"
        },
        {
            "Endpoint": "instance-8898.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-8898"
        },
        {
            "Endpoint": "instance-1018.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-1018"
        },
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "cluster-56-2020-02-25-1399"
        },
        {
            "Endpoint": "instance-4330.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "instance-4330"
        }
```

```
    ]
}

aws rds deregister-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 cluster-56-2020-02-25-1399

aws rds describe-db-proxy-targets --db-proxy-name the-proxy

{
    "Targets": []
}

aws rds register-db-proxy-targets --db-proxy-name the-proxy --db-cluster-identifier
 provisioned-cluster

{
    "DBProxyTargets": [
        {
            "Type": "TRACKED_CLUSTER",
            "Port": 0,
            "RdsResourceId": "provisioned-cluster"
        },
        {
            "Endpoint": "gkldje.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "gkldje"
        },
        {
            "Endpoint": "provisioned-1.demo.us-east-1.rds.amazonaws.com",
            "Type": "RDS_INSTANCE",
            "Port": 3306,
            "RdsResourceId": "provisioned-1"
        }
    ]
}
```

API RDS

Pour modifier un proxy à l'aide de l'API RDS, vous devez utiliser les opérations ModifierDBProxy, Modifier DBProxy TargetGroup, Désenregistrer DBProxy les cibles et Enregistrer DBProxy les cibles.

Avec `ModifyDBProxy`, vous pouvez modifier des propriétés, par exemple :

- Ensemble des secrets Secrets Manager utilisés par le proxy.
- TLS requis ou non.
- Délai d’expiration de la connexion client inactive.
- Nécessité ou non de consigner des informations supplémentaires des instructions SQL pour le débogage.
- Rôle IAM utilisé pour récupérer les secrets Secrets Manager.
- Groupes de sécurité utilisés par le proxy.

Avec `ModifyDBProxyTargetGroup`, vous pouvez modifier les paramètres liés à la connexion. Actuellement, tous les proxy ont un groupe cible unique nommé `default`. Lorsque vous travaillez avec ce groupe cible, vous indiquez le nom du proxy et `default` pour le nom du groupe cible. Vous ne pouvez pas renommer le groupe cible par défaut.

Avec `DeregisterDBProxyTargets` et `RegisterDBProxyTargets`, vous pouvez modifier le cluster Aurora à laquelle/auquel le proxy est associé via son groupe cible. Actuellement, chaque proxy peut se connecter à un cluster Aurora. Le groupe cible suit les détails de connexion des instances de base de données dans un cluster Aurora.

Ajout d’un nouvel utilisateur de base de données lors de l’utilisation du proxy RDS

Dans certains cas, vous pouvez ajouter un nouvel utilisateur de base de données à un cluster Aurora qui est associé à un proxy. Procédez selon que vous utilisez l'authentification standard avec les secrets de Secrets Manager ou l'authentification end-to-end IAM.

Si vous utilisez l'authentification IAM standard, suivez ces instructions :

1. Créez un nouveau secret Secrets Manager en suivant les instructions décrites dans la section Configuration des informations d'identification de base de données pour le proxy RDS.
2. Mettez à jour le rôle IAM pour permettre au proxy RDS d’accéder au nouveau secret Secrets Manager. Pour ce faire, mettez à jour la section des ressources de la politique de rôle IAM.
3. Modifiez le proxy RDS pour ajouter le nouveau secret de Secrets Manager sous Secrets de Secrets Manager.
4. Si le nouvel utilisateur remplace un utilisateur existant, mettez à jour les informations d’identification stockées dans le secret Secrets Manager du proxy pour l’utilisateur existant.

Si vous utilisez l'authentification end-to-end IAM, vous devez créer l'utilisateur de base de données et configurer les autorisations IAM. Pour ce faire, suivez les étapes suivantes.

1. Créez un nouvel utilisateur de base de données dans votre base de données qui correspond au nom d'utilisateur ou de rôle IAM que vous souhaitez utiliser pour l'authentification.
2. Assurez-vous que l'utilisateur de la base de données est configuré avec le plug-in d'authentification IAM dans la base de données. Consultez Création d’un compte de base de données à l’aide de l’authentification IAM.
3. Mettez à jour la politique IAM pour accorder l'`rds-db:connect`autorisation à l'utilisateur ou au rôle IAM, comme décrit dans. Création d'une politique IAM pour l'authentification end-to-end IAM
4. Assurez-vous que votre proxy est configuré pour utiliser l'authentification IAM comme schéma d'authentification par défaut.

Avec l'authentification end-to-end IAM, vous n'avez pas besoin de gérer les informations d'identification de la base de données dans les secrets de Secrets Manager, car les informations d'identification IAM sont utilisées pour l'authentification du client vers le proxy et du proxy vers la base de données.

Ajout d’un nouvel utilisateur de base de données à une base de données PostgreSQL lors de l’utilisation du proxy RDS

Lorsque vous ajoutez un nouvel utilisateur à votre base de données PostgreSQL, si vous avez exécuté la commande suivante :

```
REVOKE CONNECT ON DATABASE postgres FROM PUBLIC;
```

Accordez à l’utilisateur `rdsproxyadmin` le privilège `CONNECT` afin qu’il puisse surveiller les connexions sur la base de données cible.

```
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

Vous pouvez également autoriser d’autres utilisateurs de la base de données cible à effectuer des surveillances de l’état en modifiant `rdsproxyadmin` pour l’utilisateur de la base de données dans la commande ci-dessus.

Modification du mot de passe d’un utilisateur de base de données lors de l’utilisation du proxy RDS

Dans certains cas, vous pouvez modifier le mot de passe d’un utilisateur de base de données d’un cluster Aurora associé à un proxy. Le cas échéant, mettez à jour le secret Secrets Manager correspondant avec le nouveau mot de passe.

Si vous utilisez l'authentification end-to-end IAM, vous n'avez pas besoin de mettre à jour les mots de passe dans les secrets de Secrets Manager.

Passage de l'authentification IAM standard à l'authentification end-to-end IAM pour le proxy RDS

Si vous utilisez actuellement l'authentification IAM standard pour le proxy RDS, dans laquelle les clients s'authentifient auprès du proxy via IAM mais le proxy se connecte à la base de données à l'aide de secrets, vous pouvez passer à l'authentification IAM dans laquelle les connexions client-to-proxy et proxy-to-database les connexions utilisent l'authentification end-to-end IAM.

Pour passer à l' end-to-endauthentification IAM

1. Mettre à jour les autorisations du rôle IAM du proxy RDS

   Créez une politique d'autorisation de proxy mise à jour qui inclut à la fois Secrets Manager et `rds:db-connect` les autorisations :

   ```
   # Create updated proxy permission policy
   cat > updated-proxy-policy.json # EOF
   ```

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Sid": "GetSecretsValue",
         "Action": [
           "secretsmanager:GetSecretValue"
         ],
         "Effect": "Allow",
         "Resource": [
           "arn:aws:secretsmanager:us-east-1:123456789012:secret:secretName-1234f"
         ]
   ```

```
    },
    {
      "Sid": "RdsDBConnect",
      "Action": [
        "rds-db:connect"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:rds-db:us-east-1:123456789012:dbuser:cluster-ABCDEFGHIJKL01234/
jane_doe"
      ]
    }
  ]
}
```

Mettez à jour votre politique de rôle en tant que proxy :

```
aws iam put-role-policy \
            --role-name RDSProxyRole \
            --policy-name UpdatedProxyPermissions \
            --policy-document file://updated-proxy-policy.json
```

2. Modifiez votre proxy RDS pour activer l'authentification end-to-end IAM

```
aws rds modify-db-proxy \
  --db-proxy-name my-database-proxy \
  --default-auth-scheme IAM_AUTH \
  --region us-east-1
```

Vérifiez que l'état du proxy RDS est disponible et qu'il l'`DefaultAuthScheme`est `IAM_AUTH` avant de poursuivre afin de garantir l'absence de temps d'arrêt pendant la migration.

```
aws rds describe-db-proxies --db-proxy-name my-database-proxy --region us-east-1
```

Sortie attendue :

```
{
  "DBProxies": [
    {
      "DBProxyName": "my-database-proxy",
```

```
      "DBProxyArn": "arn:aws:rds:us-east-1:123456789012:db-
proxy:prx-0123456789abcdef",
      "Status": "available",
      ...
      "DefaultAuthScheme": "IAM_AUTH"
    }
  ]
}
```

3. Activer l'authentification IAM sur la base de données

```
aws rds modify-db-cluster \
  --db-cluster-identifier my-database-cluster \
  --enable-iam-database-authentication \
  --region us-east-1
```

4. Configuration de l'utilisateur de base de données pour l'authentification IAM

   Pour Aurora PostgreSQL :

```
GRANT rds_iam TO jane_doe;
```

   Pour Aurora MySQL :

```
ALTER USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
ALTER USER 'jane_doe'@'%' REQUIRE SSL;
```

5. Le code de votre application client n'a pas besoin de changer. Le processus de connexion reste le même :

   Pour Aurora PostgreSQL :

```
# Generate authentication token
export PGPASSWORD=$(aws rds generate-db-auth-token \
  --hostname my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
 \
  --port 5432 \
  --username jane_doe \
  --region us-east-1)

# Connect to database through proxy
```

```
psql "host=my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
 port=5432 user=jane_doe dbname=postgres password=$PGPASSWORD sslmode=require
 sslrootcert=us-east-1-bundle.pem"
```

Pour Aurora MySQL :

```
# Generate authentication token
export MYSQL_PWD=$(aws rds generate-db-auth-token \
  --hostname my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com
 \
  --port 3306 \
  --username jane_doe \
  --region us-east-1)

# Connect to database through proxy
mysql -h my-database-proxy.proxy-ABCDEFGHIJKL01234.us-east-1.rds.amazonaws.com \
  -P 3306 \
  -u jane_doe \
  --ssl-ca=us-east-1-bundle.pem \
  --enable-cleartext-plugin
```

Considérations relatives à la connexion RDS Proxy

Configuration des paramètres de connexion

Pour ajuster le regroupement de connexion RDS Proxy, vous pouvez modifier les paramètres suivants :

- IdleClientTimeout
- MaxConnectionsPercent
- MaxIdleConnectionsPercent
- ConnectionBorrowTimeout

IdleClientTimeout

Vous pouvez spécifier la durée d'inactivité d'une connexion client avant que le proxy puisse la fermer. La valeur par défaut est de 1 800 secondes (30 minutes).

Une connexion client est considérée comme inactive lorsque l'application ne soumet aucune nouvelle demande dans le délai défini après l'achèvement de la demande précédente. La connexion à la base de données sous-jacente reste ouverte et est renvoyée au regroupement de connexions. Ainsi, elle peut être réutilisée pour de nouvelles connexions client. Réduisez le délai d'expiration de la connexion client inactive si vous souhaitez que le proxy supprime proactivement les connexions obsolètes. Si votre charge de travail se connecte régulièrement au proxy, augmentez le délai d'expiration de la connexion client inactive pour économiser le coût d'établissement de connexions.

Ce paramètre est représenté par le champ Idle client connection timeout (Délai d'inactivité de la connexion client) dans la console RDS et le paramètre `IdleClientTimeout` dans l'AWS CLI et l'API. Pour savoir comment modifier la valeur du champ Idle client connection timeout (Délai d'inactivité de la connexion client) dans la console RDS, consultez AWS Management Console. Pour apprendre à modifier la valeur du paramètre `IdleClientTimeout`, utilisez la commande de la CLI modify-db-proxy ou l'opération d'API ModifyDBProxy.

MaxConnectionsPercent

Vous pouvez limiter le nombre de connexions qu'un RDS Proxy peut établir avec la base de données cible. Vous indiquez la limite, sous forme de pourcentage, des connexions maximales disponibles pour votre base de données. Ce paramètre est représenté par le champ Connection pool maximum connections (Connexions maximales du groupe de connexion) dans la console RDS et le paramètre `MaxConnectionsPercent` dans l'AWS CLI et l'API.

La valeur `MaxConnectionsPercent` est exprimée en pourcentage du paramètre `max_connections` pour le cluster de bases de données Aurora utilisé par le groupe cible. Le proxy ne crée pas toutes ces connexions à l'avance. Ce paramètre permet au proxy d'établir ces connexions, car la charge de travail en a besoin.

Par exemple, pour une cible de base de données enregistrée avec `max_connections` définies sur 1 000 et `MaxConnectionsPercent` défini sur 95, RDS Proxy définit 950 connexions comme la limite supérieure pour les connexions simultanées à cette cible de base de données.

Le fait que votre charge de travail atteigne le nombre maximum de connexions à la base de données autorisées a souvent pour effet secondaire d'augmenter la latence globale des requêtes, ainsi que d'augmenter la métrique `DatabaseConnectionsBorrowLatency`. Vous pouvez surveiller les connexions à la base de données actuellement utilisées et le nombre total de connexions autorisées en comparant les métriques `DatabaseConnections` et `MaxDatabaseConnectionsAllowed`.

Pour définir ce paramètre, tenez compte des bonnes pratiques suivantes :

- Prévoyez une marge de connexion suffisante pour les modifications du modèle de la charge de travail. Il est recommandé de définir le paramètre afin qu'il soit au moins 30 % supérieur à votre utilisation surveillée maximale récente. Comme RDS Proxy redistribue les quotas de connexion à la base de données entre plusieurs nœuds, les modifications de la capacité interne peuvent nécessiter une marge d'au moins 30 % pour les connexions supplémentaires afin d'éviter des latences d'emprunt plus importantes.
- RDS Proxy réserve un certain nombre de connexions pour une surveillance active afin de permettre un basculement rapide, le routage du trafic et les opérations internes. La métrique `MaxDatabaseConnectionsAllowed` n'inclut pas ces connexions réservées. Elle représente le nombre de connexions disponibles pour répondre à la charge de travail et peut être inférieure à la valeur dérivée du paramètre `MaxConnectionsPercent`.

  Les valeurs `MaxConnectionsPercent` minimales recommandées sont les suivantes :

  - db.t3.small : 100
  - db.t3.medium : 55
  - db.t3.large : 35
  - db.r3.large ou supérieur : 20

  Si plusieurs instances cibles sont enregistrées avec RDS Proxy, comme un cluster Aurora avec des nœuds de lecteur, définissez la valeur minimale en fonction de la plus petite instance enregistrée.

Pour savoir comment modifier la valeur du champ Connection pool maximum connections (Connexions maximales au groupe de connexion) dans la console RDS, consultez AWS Management Console. Pour apprendre à modifier la valeur du paramètre `MaxConnectionsPercent`, utilisez la commande de la CLI modify-db-proxy-target-group ou l'opération d'API ModifyDBProxyTargetGroup.

> **Important**
>
> Si le cluster de bases de données fait partie d'une base de données globale où le transfert d'écriture est activé, réduisez la valeur `MaxConnectionsPercent` de votre proxy du quota alloué au transfert d'écriture. Le quota de transfert d'écriture est défini dans le paramètre de cluster de bases de données `aurora_fwd_writer_max_connections_pct`. Pour plus d'informations sur le transfert d'écriture, consultez Utilisation du transfert d'écriture dans une base de données globale Amazon Aurora.

Pour en savoir plus sur les limites de connexion aux bases de données, consultez Nombre maximal de connexions à une instance de base de données Aurora MySQL et Nombre maximal de connexions à une instance de base de données Aurora PostgreSQL.

MaxIdleConnectionsPercent

Vous pouvez contrôler le nombre de connexions aux bases de données inactives que RDS Proxy peut conserver dans le groupe de connexion. Par défaut, RDS Proxy considère qu'une connexion à une base de données dans son groupe est inactive lorsqu'il n'y a pas eu d'activité sur la connexion pendant cinq minutes.

La valeur `MaxIdleConnectionsPercent` est exprimée en pourcentage du paramètre `max_connections` du groupe de cibles de l'instance de base de données RDS. La valeur par défaut est de 50 % de `MaxConnectionsPercent` et la limite supérieure est la valeur de `MaxConnectionsPercent`. Par exemple, si `MaxConnectionsPercent` est défini sur 80, la valeur par défaut de `MaxIdleConnectionsPercent` est 40.

Une valeur élevée permet au proxy de laisser ouvert un pourcentage élevé de connexions inactives à la base de données. Avec une valeur faible, le proxy ferme un pourcentage élevé de connexions de base de données inactives. Si vos charges de travail sont imprévisibles, pensez à définir une valeur élevée pour `MaxIdleConnectionsPercent`. Cela signifie que RDS Proxy peut prendre en charge les vagues d'activité sans ouvrir de nombreuses nouvelles connexions aux bases de données.

Ce paramètre est représenté par le paramètre `MaxIdleConnectionsPercent` du `DBProxyTargetGroup` dans l'AWS CLI et l'API. Pour apprendre à modifier la valeur du paramètre `MaxIdleConnectionsPercent`, utilisez la commande de la CLI modify-db-proxy-target-group ou l'opération d'API ModifyDBProxyTargetGroup.

Pour en savoir plus sur les limites de connexion aux bases de données, consultez Nombre maximal de connexions à une instance de base de données Aurora MySQL et Nombre maximal de connexions à une instance de base de données Aurora PostgreSQL.

ConnectionBorrowTimeout

Vous pouvez choisir combien de temps le RDS Proxy doit attendre la disponibilité d'utilisation d'une connexion à une base de données dans le groupe de connexion avant de renvoyer une erreur de dépassement de délai d'attente. La durée par défaut est de 120 secondes. Ce paramètre s'applique lorsque le nombre maximal de connexions est atteint et qu'aucune connexion n'est disponible dans le groupe de connexion. Il s'applique également si aucune instance de base de données appropriée n'est disponible pour traiter la requête en raison, par exemple, d'une opération de basculement

en cours. À l'aide de ce paramètre, vous pouvez définir la meilleure période d'attente pour votre application sans avoir à modifier le délai d'attente de requête dans votre code d'application.

Ce paramètre est représenté par le champ Connection borrow timeout (Délai d'expiration d'emprunt de connexion) dans la console RDS ou le paramètre `ConnectionBorrowTimeout` du `DBProxyTargetGroup` dans l'AWS CLI ou l'API. Pour savoir comment modifier la valeur du champ Connection borrow timeout (Délai d'expiration d'emprunt de connexion) dans la console RDS, consultez AWS Management Console. Pour apprendre à modifier la valeur du paramètre `ConnectionBorrowTimeout`, utilisez la commande de la CLI modify-db-proxy-target-group ou l'opération d'API ModifyDBProxyTargetGroup.

Connexions client et connexions aux bases de données

Les connexions entre votre application et RDS Proxy sont appelées connexions client. Les connexions d'un proxy à la base de données sont appelées connexions à la base de données. Lorsque vous utilisez RDS Proxy, les connexions client s'arrêtent au niveau du proxy tandis que les connexions à la base de données sont gérées au sein de RDS Proxy.

Le regroupement des connexions côté application peut offrir l'avantage de réduire l'établissement de connexions récurrentes entre votre application et RDS Proxy.

Tenez compte des aspects de configuration suivants avant d'implémenter un regroupement de connexions côté application :

- Durée de vie maximale d'une connexion client : RDS Proxy impose une durée de vie maximale de 24 heures aux connexions client. Cette valeur n'est pas configurable. Configurez votre regroupement avec une durée de vie maximale inférieure à 24 heures afin d'éviter des interruptions inattendues de la connexion client.
- Délai d'inactivité de la connexion client : RDS Proxy impose une durée d'inactivité maximale pour les connexions client. Configurez votre regroupement avec un délai d'inactivité inférieur au délai d'inactivité de votre connexion client pour RDS Proxy afin d'éviter les interruptions de connexion inattendues.

Le nombre maximal de connexions client configurées dans votre regroupement de connexions côté application ne doit pas nécessairement être limité au paramètre max_connections pour RDS Proxy.

Le regroupement des connexions client prolonge leur durée de vie. Si vos connexions sont épinglées, le regroupement des connexions client peut réduire l'efficacité du multiplexage. Les connexions client épinglées mais inactives dans le regroupement de connexions côté application continuent de

conserver une connexion à la base de données et empêchent la réutilisation de cette connexion par d'autres connexions client. Consultez les journaux de votre proxy pour vérifier si vos connexions sont épinglées.

> **Note**
>
> RDS Proxy ferme les connexions à la base de données après 24 heures lorsqu'elles ne sont plus utilisées. Le proxy effectue cette action indépendamment de la valeur du paramètre de connexions inactives maximum.

Contournement de l'épinglage d'un proxy RDS

Le multiplexage est plus efficace lorsque les demandes de base de données ne dépendent pas des informations d'état issues de demandes précédentes. Dans ce cas, le proxy RDS peut réutiliser une connexion à la fin de chaque transaction. Ces informations d'état incluent la plupart des variables et des paramètres de configuration que vous pouvez modifier à l'aide des instructions `SET` ou `SELECT`. Les transactions SQL sur une connexion client peuvent se multiplexer entre les connexions de base de données sous-jacentes par défaut.

Vos connexions au proxy peuvent entrer dans un état appelé épinglage. Lorsqu'une connexion est épinglée, chaque transaction ultérieure utilise la même connexion de base de données sous-jacente jusqu'à la fin de la session. De même, les autres connexions client ne peuvent pas réutiliser cette connexion à la base de données tant que la session n'est pas terminée. La session se termine lorsque la connexion client est supprimée.

Le proxy RDS épingle automatiquement une connexion client à une connexion de base de données spécifique lorsqu'il détecte un changement d'état de session qui n'est pas approprié pour d'autres sessions. L'épinglage réduit l'efficacité de la réutilisation des connexions. Si la totalité, ou presque, de vos connexions font l'objet d'un épinglage, pensez à modifier le code de votre application ou votre charge de travail afin de réduire les conditions à l'origine de l'épinglage.

Par exemple, votre application modifie une variable de session ou un paramètre de configuration. Dans ce cas, les instructions ultérieures peuvent reposer sur la nouvelle variable ou le nouveau paramètre pour entrer en vigueur. Ainsi, lorsque le proxy RDS traite des demandes de modification des variables ou des paramètres de configuration de session, il épingle cette session à la connexion de base de données. De cette manière, l'état de session reste en vigueur pour toutes les transactions ultérieures de la même session.

Pour les moteurs de bases de données, cette règle ne s'applique pas à tous les paramètres que vous pouvez définir. Le proxy RDS suit certaines instructions et variables. Ainsi, le proxy RDS n'épingle pas la session lorsque vous les modifiez. Dans ce cas, le proxy RDS réutilise la connexion uniquement pour les autres sessions dont les valeurs de ces paramètres sont identiques. Pour les listes des instructions et des variables suivies pour Aurora MySQL, consultez Ce que le proxy RDS suit pour les bases de données Aurora MySQL.

Ce que le proxy RDS suit pour les bases de données Aurora MySQL

Le proxy RDS suit les instructions MySQL suivantes :

- DROP DATABASE
- DROP SCHEMA
- USE

Le proxy RDS suit les variables MySQL suivantes :

- `AUTOCOMMIT`
- `AUTO_INCREMENT_INCREMENT`
- `CHARACTER SET (or CHAR SET)`
- `CHARACTER_SET_CLIENT`
- `CHARACTER_SET_DATABASE`
- `CHARACTER_SET_FILESYSTEM`
- `CHARACTER_SET_CONNECTION`
- `CHARACTER_SET_RESULTS`
- `CHARACTER_SET_SERVER`
- `COLLATION_CONNECTION`
- `COLLATION_DATABASE`
- `COLLATION_SERVER`
- `INTERACTIVE_TIMEOUT`
- `NAMES`
- `NET_WRITE_TIMEOUT`
- `QUERY_CACHE_TYPE`

- `SESSION_TRACK_SCHEMA`
- `SQL_MODE`
- `TIME_ZONE`
- `TRANSACTION_ISOLATION (or TX_ISOLATION)`
- `TRANSACTION_READ_ONLY (or TX_READ_ONLY)`
- `WAIT_TIMEOUT`

> **Note**
>
> Le proxy RDS suit les modifications apportées aux variables `TRANSACTION_ISOLATION` et `TRANSACTION_READ_ONLY` lorsque vous les définissez au cours dans le champ de la session. Toutefois, si vous les définissez lors du prochain champ de transaction, le proxy RDS épingle les connexions. Ce comportement s'applique que vous utilisiez une instruction `SET` ou `SET TRANSACTION` pour configurer ces valeurs.

Minimiser l'épinglage

Le réglage des performances du proxy RDS entraîne une tentative d'optimisation de la réutilisation des connexions au niveau de la transaction (multiplexage) en réduisant l'épinglage.

Vous pouvez minimiser l'épinglage en procédant comme suit :

- Évitez les requêtes de base de données inutiles qui pourraient provoquer l'épinglage.
- Définissez les variables et les paramètres de configuration de manière cohérente sur toutes les connexions. De cette façon, les sessions ultérieures sont plus susceptibles de réutiliser les connexions qui ont ces paramètres particuliers.

  En revanche, pour PostgreSQL, la définition d'une variable entraîne l'épinglage de la session.
- Pour une base de données de la famille de moteur MySQL, appliquez un filtre d'épinglage de session au proxy. Vous pouvez configurer certains types d'opérations pour qu'elles n'épinglent pas la session si vous savez que cela n'affecte pas le bon fonctionnement de votre application.
- Consultez la fréquence d'épinglage en surveillant la métrique Amazon CloudWatch `DatabaseConnectionsCurrentlySessionPinned`. Pour en savoir plus sur ce sujet et sur d'autres métriques CloudWatch, consultez Surveillance des métriques RDS Proxy avec Amazon CloudWatch.

- Si vous utilisez des instructions `SET` pour exécuter une initialisation identique pour chaque connexion client, vous pouvez conserver le multiplexage au niveau de la transaction. Dans ce cas, vous déplacez les instructions qui définissent l'état initial de la session vers la requête d'initialisation utilisée par un proxy. Cette propriété est une chaîne contenant une ou plusieurs instructions SQL, séparées par des points-virgules.

  Par exemple, vous pouvez définir une requête d'initialisation pour un proxy qui établit certains paramètres de configuration. Le proxy RDS applique ensuite ces paramètres dès qu'il configure une nouvelle connexion pour ce proxy. Vous pouvez supprimer les instructions `SET` correspondantes de votre code d'application, afin qu'elles n'interfèrent pas avec le multiplexage au niveau de la transaction.

  Pour les métriques relatives à la fréquence d'épinglage d'un proxy, consultez Surveillance des métriques RDS Proxy avec Amazon CloudWatch.

Conditions qui entraînent l'épinglage pour toutes les familles de moteurs

Le proxy épingle la session à la connexion en cours dans les situations suivantes où le multiplexage peut entraîner un comportement inattendu :

- Le proxy épingle la session si la taille de texte de l'instruction est supérieure à 16 Ko.

Conditions qui entraînent l'épinglage pour Aurora MySQL

Pour MySQL, les interactions suivantes entraînent également l'épinglage :

- Le proxy épingle la session en cas d'instructions de verrouillage de table `LOCK TABLE`, `LOCK TABLES` ou `FLUSH TABLES WITH READ LOCK` explicites.
- La création de verrous nommés à l'aide de `GET_LOCK` entraîne le proxy à épingler la session.
- La définition d'une variable utilisateur ou système (sauf exceptions) épingle la session au proxy. Si cela limite considérablement la réutilisation des connexions, vous pouvez configurer des opérations `SET` pour éviter l'épinglage. Pour ce faire, ajustez la propriété des filtres d'épinglage de session. Pour plus d'informations, consultez Création d'un proxy pour Amazon Aurora et Modification d'un RDS Proxy.
- Le proxy épingle la session lors de la création d'une table temporaire. De cette façon, le contenu de la table temporaire est conservé tout au long de la session, sans tenir compte des limites de transaction.

- L’appel des fonctions `ROW_COUNT` et `FOUND_ROWS` entraîne parfois un épinglage.

  Les circonstances exactes dans lesquelles ces fonctions provoquent un épinglage peuvent différer selon les versions d’Aurora MySQL compatibles avec MySQL 5.7.
- Le proxy épingle la session en cas d’instructions préparées. Cette règle s’applique si l’instruction préparée utilise du texte SQL ou le protocole binaire.
- Le proxy RDS n’épingle pas les connexions lorsque vous utilisez SET LOCAL.
- L’appel de procédures et de fonctions stockées ne provoque pas d’épinglage. Le proxy RDS ne détecte aucun changement d’état de session résultant de tels appels. Assurez-vous que votre application ne modifie pas l’état de session dans les routines stockées si vous reposez sur cet état de session pour persister entre les transactions. Par exemple, le proxy RDS n’est actuellement pas compatible avec une procédure stockée qui crée une table temporaire qui est conservée dans toutes les transactions.

Si vous avez des connaissances avancées sur le comportement de votre application, vous pouvez ignorer le comportement d’épinglage de certaines instructions d’application. Pour ce faire, sélectionnez l’option Filtres d’épinglage de session lors de la création du proxy. Actuellement, vous pouvez désactiver l’épinglage de session pour définir des variables de session et des paramètres de configuration.

Conditions qui entraînent l’épinglage pour Aurora PostgreSQL

Pour PostgreSQL, les interactions suivantes provoquent également l’épinglage :

- Utilisation des commandes `SET`.
- Utilisation des commandes `PREPARE`, `DISCARD`, `DEALLOCATE` ou `EXECUTE` pour gérer les instructions préparées.
- Création de séquences, de tables ou de vues temporaires.
- Déclaration de curseurs.
- Suppression de l’état de la session.
- Écoute d’un canal de notification.
- Chargement d’un module de bibliothèque comme `auto_explain`.
- Manipulation de séquences à l’aide de fonctions comme `nextval` et `setval`.
- Interaction avec des verrous à l’aide de fonctions comme `pg_advisory_lock` et `pg_try_advisory_lock`.

> **Note**
>
> Le proxy RDS ne s'épingle pas aux verrous consultatifs au niveau des transactions, en particulier `pg_advisory_xact_lock`, `pg_advisory_xact_lock_shared`, `pg_try_advisory_xact_lock` et `pg_try_advisory_xact_lock_shared`.

- Définition d'un paramètre ou rétablissement de sa valeur par défaut. Plus précisément, l'utilisation des commandes `SET` et `set_config` pour attribuer des valeurs par défaut aux variables de session.
- L'appel de procédures et de fonctions stockées ne provoque pas d'épinglage. Le proxy RDS ne détecte aucun changement d'état de session résultant de tels appels. Assurez-vous que votre application ne modifie pas l'état de session dans les routines stockées si vous reposez sur cet état de session pour persister entre les transactions. Par exemple, le proxy RDS n'est actuellement pas compatible avec une procédure stockée qui crée une table temporaire qui est conservée dans toutes les transactions.
- Suppression de l'état de la session. Si vous utilisez des bibliothèques de regroupement de connexions avec une requête `DISCARD ALL` configurée comme une requête de réinitialisation, le proxy RDS épingle votre connexion client lors de la publication. Cela réduit l'efficacité du multiplexage du proxy et peut entraîner des résultats inattendus, car la commande `DISCARD ALL` peut interférer avec la gestion de l'état de session.

Suppression d'un RDS Proxy

Vous pouvez supprimer un proxy si vous n'en avez plus besoin. Vous pouvez également supprimer un proxy si vous mettez hors service l'instance ou le cluster de bases de données qui lui est associé.

AWS Management Console

Suppression d'un proxy

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, sélectionnez Proxies.
3. Choisissez le proxy à supprimer de la liste.
4. Sélectionnez Suppression du proxy.

AWS CLI

Pour supprimer un proxy de base de données, utilisez la commande de l'AWS CLI delete-db-proxy. Pour supprimer les associations connexes, utilisez également la commande deregister-db-proxy-targets.

```
aws rds delete-db-proxy --name proxy_name
```

```
aws rds deregister-db-proxy-targets
    --db-proxy-name proxy_name
    [--target-group-name target_group_name]
    [--target-ids comma_separated_list]       # or
    [--db-instance-identifiers instance_id]       # or
    [--db-cluster-identifiers cluster_id]
```

API RDS

Pour supprimer un proxy de base de données, appelez la fonction d'API Amazon RDS DeleteDBProxy. Pour supprimer les éléments et associations connexes, vous appelez également les fonctions DeleteDBProxyTargetGroup et DeregisterDBProxyTargets.

Utilisation des points de terminaison du proxy Amazon RDS

Les points de terminaison RDS Proxy permettent une gestion flexible et performante des connexions aux bases de données, ce qui améliore la capacité de mise à l'échelle, la disponibilité et la sécurité. Avec les points de terminaison proxy, vous pouvez :

- Simplifier la surveillance et la résolution de problèmes : utilisez plusieurs points de terminaison pour suivre et gérer de façon indépendante les connexions provenant de différentes applications.
- Améliorer la capacité de mise à l'échelle de la lecture : tirez parti des points de terminaison des lecteurs avec les clusters de bases de données Aurora pour répartir efficacement le trafic de lecture, en vue d'optimiser les performances pour les charges de travail gourmandes en requêtes.
- Activez l'accès inter-VPC : connectez les bases de données à VPCs l'aide de points de terminaison inter-VPC, ce qui permet à des ressources telles que les instances Amazon d'un VPC d'accéder EC2 aux bases de données d'un autre.

Rubriques

- Présentation des points de terminaison proxy

- Limites pour les points de terminaison proxy
- Utilisation des points de terminaison de lecteur avec les clusters Aurora
- Accès aux bases de données Aurora via VPCs
- Création d'un point de terminaison proxy
- Affichage des points de terminaison proxy
- Modification d'un point de terminaison proxy
- Suppression d'un point de terminaison proxy

Présentation des points de terminaison proxy

Travailler avec des points de terminaison RDS Proxy implique les mêmes types de procédures qu'avec les points de terminaison de clusters de bases de données et de lecteur Aurora. Pour vous familiariser avec les points de terminaison Aurora, consultez les informations dans Connexions de point de terminaison Amazon Aurora.

Lorsque vous utilisez RDS Proxy avec un cluster Aurora, le point de terminaison par défaut prend en charge les opérations de lecture et d'écriture. Cela signifie qu'il achemine toutes les demandes vers l'instance d'enregistreur du cluster, ce qui peut l'amener à atteindre sa limite `max_connections`. Pour mieux répartir le trafic, vous pouvez créer des points de terminaison supplémentaires read/write ou en lecture seule pour votre proxy, ce qui permet une gestion plus efficace de la charge de travail et une meilleure évolutivité.

Utilisez un point de terminaison en lecture seule avec votre proxy pour gérer les requêtes de lecture, comme vous le feriez avec un point de terminaison de lecteur pour un cluster provisionné Aurora. Cette approche maximise la capacité de mise à l'échelle de lecture d'Aurora en répartissant les requêtes sur une ou plusieurs instances de base de données de lecteur. En utilisant un point de terminaison en lecture seule et en ajoutant d'autres instances de lecteur selon vos besoins, vous pouvez augmenter le nombre de requêtes et de connexions simultanées que votre cluster est capable de gérer.

> **Tip**
>
> Lorsque vous créez un proxy pour un cluster Aurora à l'aide de AWS Management Console, le proxy RDS peut créer automatiquement un point de terminaison de lecteur. Pour plus d'informations sur les avantages d'un point de terminaison de lecteur, consultez Utilisation des points de terminaison de lecteur avec les clusters Aurora.

Lorsque vous créez un point de terminaison proxy, vous pouvez l'associer à un cloud privé virtuel (VPC) différent de celui utilisé par le VPC du proxy. Cela vous permet de vous connecter au proxy à partir d'un autre VPC, par exemple celui d'une autre application au sein de votre organisation.

Pour plus d'informations sur les limites associées aux points de terminaison proxy, consultez Limites pour les points de terminaison proxy.

Les journaux RDS Proxy préfixent chaque entrée avec le nom du point de terminaison proxy associé. Il peut s'agir du nom que vous avez spécifié pour un point de terminaison défini par l'utilisateur ou du nom spécial du point `default` de read/write terminaison par défaut du proxy.

Chaque point de terminaison du proxy possède son propre ensemble de CloudWatch mesures. Surveillez les métriques de tous les points de terminaison du proxy, d'un point de terminaison spécifique ou de tous les points de terminaison d'un proxy read/write ou de tous les points de terminaison en lecture seule d'un proxy. Pour de plus amples informations, veuillez consulter Surveillance des métriques RDS Proxy avec Amazon CloudWatch.

Un point de terminaison de proxy utilise le même mécanisme d'authentification que le proxy auquel il est associé. RDS Proxy configure automatiquement des permissions et des autorisations pour le point de terminaison défini par l'utilisateur, conformément aux propriétés du proxy associé.

Pour découvrir comment les points de terminaison proxy fonctionnent pour les clusters de base de données dans une base de données globale Aurora, consultez Fonctionnement des points de terminaison du proxy RDS avec les bases de données globales.

Limites pour les points de terminaison proxy

Les points de terminaison RDS Proxy présentent les limites suivantes :

- Le point de terminaison par défaut du proxy RDS ne peut pas être modifié.
- Le nombre maximal de points de terminaison définis par l'utilisateur pour un proxy est de 20. Un proxy peut ainsi avoir jusqu'à 21 points de terminaison : le point de terminaison par défaut, plus 20 que vous créez.
- Lorsque vous associez des points de terminaison supplémentaires à un proxy, RDS Proxy détermine automatiquement les instances de base de données à utiliser dans votre cluster pour chaque point de terminaison. Vous ne pouvez pas sélectionner d'instances spécifiques comme c'est le cas pour les points de terminaison personnalisés Aurora.
- Pour les IPv6 types de réseaux de points de terminaison à double pile, votre VPC et vos sous-réseaux doivent être configurés pour prendre en charge le type de réseau sélectionné.

Lorsque vous créez un proxy, RDS crée automatiquement un point de terminaison d'un VPC pour garantir une communication sécurisée entre les applications et la base de données. Le point de terminaison d'un VPC est visible et est accessible à partir de la console Amazon VPC.

L'ajout d'un nouveau point de terminaison proxy provisionne un point de terminaison d' AWS PrivateLink interface. Si vous ajoutez un ou plusieurs points de terminaison à votre proxy, des frais supplémentaires vous seront facturés. Pour plus d'informations, consultez Tarification de RDS Proxy.

Utilisation des points de terminaison de lecteur avec les clusters Aurora

Vous pouvez créer et vous connecter à des points de terminaison en lecture seule appelés points de terminaison de lecteur lorsque vous utilisez RDS Proxy avec les clusters Aurora. Ces points de terminaison du lecteur peuvent améliorer la capacité de mise à l'échelle de lecture de vos applications exigeantes en requêtes. Les points de terminaison du lecteur peuvent également améliorer la disponibilité de vos connexions si une instance de base de données de lecteur dans votre cluster devient indisponible.

> **Note**
>
> Lorsque vous spécifiez qu'un nouveau point de terminaison est en lecture seule, RDS Proxy exige que le cluster Aurora contienne une ou plusieurs instances de base de données de lecteur. Dans certains cas, vous pouvez modifier la cible du proxy pour un cluster Aurora ne contenant qu'un seul cluster rédacteur. Dans ce cas, toutes les demandes au point de terminaison de lecteur échouent avec une erreur. Les demandent échouent également si la cible du proxy est une instance RDS au lieu d'un cluster Aurora.
> Si un cluster Aurora possède des instances de lecteur mais que ces instances ne sont pas disponibles, RDS Proxy attend d'envoyer la demande au lieu de retourner une erreur immédiatement. Si aucune instance de lecteur n'est disponible pendant la période de délai d'expiration de l'emprunt de connexion, la demande retourne une erreur.

Amélioration de la disponibilité des applications par les points de terminaison du lecteur

Il peut arriver qu'une ou plusieurs instances de lecteur de votre cluster deviennent indisponibles. Dans ce cas, les connexions qui utilisent un point de terminaison de lecteur d'un proxy de base de données peuvent récupérer plus rapidement que celles qui utilisent le point de terminaison du lecteur Aurora. RDS Proxy achemine les connexions uniquement vers les instances de lecteur disponibles

dans le cluster. La mise en cache DNS ne provoque pas de retard lorsqu'une instance devient indisponible.

Si la connexion est multiplexée, RDS Proxy dirige les requêtes suivantes vers une autre instance de base de données de lecteur sans aucune interruption de votre application. Pendant la bascule automatique vers une nouvelle instance de lecteur, RDS Proxy vérifie le retard de réplication entre les anciennes et les nouvelles instances de lecteur. RDS Proxy s'assure que la nouvelle instance de lecteur est à jour avec les mêmes modifications que l'instance précédente. De cette façon, votre application ne voit jamais de données obsolètes lorsque RDS Proxy passe d'une instance de base de données de lecteur à une autre.

Si la connexion est épinglée, la requête suivante sur la connexion retourne une erreur. Toutefois, votre application peut se reconnecter immédiatement au même point de terminaison. RDS Proxy achemine la connexion vers une autre instance de base de données du lecteur qui se trouve dans l'état `available`. Lorsque vous vous reconnectez manuellement, RDS Proxy ne vérifie pas le décalage de réplication entre l'ancienne instance de lecteur et la nouvelle.

Si votre cluster Aurora n'a pas d'instances de lecteur disponibles, RDS Proxy vérifie si cette condition est temporaire ou permanente. Le comportement respectif dans chaque cas est le suivant :

- Supposons que votre cluster contienne une ou plusieurs instances de base de données de lecteur, mais qu'aucune d'entre elles ne se trouve dans l'état `Available`. Par exemple, toutes les instances de lecteur peuvent se relancer ou rencontrer des problèmes. Dans ce cas, les tentatives de connexion à un point de terminaison de lecteur attendent qu'une instance de lecteur devienne disponible. Si aucune instance de lecteur n'est disponible pendant la période de délai d'expiration de l'emprunt de connexion, la tentative de connexion échoue. Si une instance de lecteur devient disponible, la tentative de connexion aboutit.
- Supposons que votre cluster ne contienne pas d'instances de base de données de lecteur. Dans ce cas, RDS Proxy retourne une erreur immédiatement si vous essayez de vous connecter à un point de terminaison de lecteur. Pour corriger ce problème, ajoutez une ou plusieurs instances de lecteur à votre cluster avant de vous connecter au point de terminaison du lecteur.

Amélioration de la capacité de mise à l'échelle des requêtes par les points de terminaison du lecteur

Les points de terminaison de lecteur d'un proxy peuvent améliorer l'évolutivité des requêtes Aurora de la manière suivante :

- Lorsque vous ajoutez des instances de lecteur à votre cluster Aurora, RDS Proxy peut acheminer de nouvelles connexions à l'un des points de terminaison de lecteur vers les différentes instances de lecteur. De cette façon, les requêtes effectuées à l'aide d'une connexion de point de terminaison de lecteur ne ralentissent pas les demandes effectuées à l'aide d'une autre connexion de point de terminaison de lecteur. Les requêtes s'exécutent sur des instances de base de données distinctes. Chaque instance de base de données possède ses propres ressources de calcul, de cache tampon, etc.
- Dans la mesure du possible, RDS Proxy utilise la même instance de base de données de lecteur pour tous les problèmes de requêtes utilisant une connexion de point de terminaison de lecteur particulière. De cette façon, un ensemble de requêtes associées sur les mêmes tables peut tirer avantage de la mise en cache, de l'optimisation du plan, etc., sur une instance de base de données particulière.
- Si une instance de base de données de lecteur devient indisponible, l'effet sur votre application sera différent selon que la séance est multiplexée ou épinglée. Si la séance est multiplexée, RDS Proxy achemine toutes les requêtes ultérieures vers une autre instance de base de données de lecteur sans que vous ayez à intervenir. Si la séance est épinglée, votre application obtient une erreur et doit se reconnecter. Vous pouvez vous reconnecter au point de terminaison du lecteur immédiatement et RDS Proxy achemine la connexion vers une instance de base de données de lecteur disponible. Pour plus d'informations sur le multiplexage et l'épinglage des séances proxy, consultez Présentation des concepts RDS Proxy.
- Plus le cluster contient d'instances de base de données de lecteur, plus vous pouvez établir de connexions simultanées à l'aide de points de terminaison de lecteur. Par exemple, supposons que votre cluster contienne quatre instances de base de données de lecteur, qui sont chacune configurées pour prendre en charge 200 connexions simultanées. Supposons que votre proxy soit configuré pour utiliser 50 % du nombre maximal de connexions. Ici, le nombre maximal de connexions que vous pouvez effectuer via les points de terminaison de lecteur dans le proxy est de 100 (50 % de 200) pour le lecteur 1. Le nombre est également de 100 pour le lecteur 2, et ainsi de suite jusqu'à un total de 400. Si vous doublez le nombre d'instances de base de données de lecteur dans le cluster à huit, le nombre maximal de connexions via les points de terminaison de lecteur double également pour atteindre 800.

Exemples d'utilisation de points de terminaison de lecteur

L'exemple Linux suivant montre comment vous pouvez confirmer que vous êtes connecté à un cluster Aurora MySQL via un point de terminaison de lecteur. Le paramètre de configuration `innodb_read_only` est activé. Les tentatives d'effectuer des opérations en écriture telles que

des instructions `CREATE DATABASE` retournent une erreur. Vous pouvez confirmer que vous êtes connecté à une instance de base de données de lecteur en vérifiant le nom de l'instance de base de données à l'aide de la variable `aurora_server_id`.

> **Tip**
>
> Ne vous fiez pas uniquement à la vérification du nom de l'instance de base de données pour déterminer si la connexion est en lecture seule read/write ou en lecture seule. N'oubliez pas que les instances de base de données dans un cluster Aurora peuvent modifier les rôles entre le rédacteur et le lecteur lorsque des basculements se produisent.

```
$ mysql -h endpoint-demo-reader.endpoint.proxy-demo.us-east-1.rds.amazonaws.com -u
 admin -p
...
mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+
mysql> create database shouldnt_work;
ERROR 1290 (HY000): The MySQL server is running with the --read-only option so it
 cannot execute this statement

mysql> select @@aurora_server_id;
+---------------------------------------+
| @@aurora_server_id                    |
+---------------------------------------+
| proxy-reader-endpoint-demo-instance-3 |
+---------------------------------------+
```

L'exemple suivant montre comment votre connexion à un point de terminaison de lecteur proxy peut continuer à fonctionner même lorsque l'instance de base de données de lecteur est supprimée. Dans cet exemple, le cluster Aurora contient deux instances de lecteur, `instance-5507` et `instance-7448`. La connexion au point de terminaison du lecteur commence en utilisant l'une des instances du lecteur. Pendant l'exemple, cette instance de lecteur est supprimée par une commande `delete-db-instance`. RDS Proxy passe à une autre instance de lecteur pour les requêtes suivantes.

```
$ mysql -h reader-demo.endpoint.proxy-demo.us-east-1.rds.amazonaws.com
  -u my_user -p
...
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-5507      |
+--------------------+

mysql> select @@innodb_read_only;
+--------------------+
| @@innodb_read_only |
+--------------------+
|                  1 |
+--------------------+

mysql> select count(*) from information_schema.tables;
+----------+
| count(*) |
+----------+
|      328 |
+----------+
```

Pendant que la séance `mysql` continue de s'exécuter, la commande suivante supprime l'instance de lecteur à laquelle le point de terminaison du lecteur est connecté.

```
aws rds delete-db-instance --db-instance-identifier instance-5507 --skip-final-snapshot
```

Les requêtes dans la session `mysql` continuent à opérer sans avoir besoin de se reconnecter. RDS Proxy bascule automatiquement vers une autre instance de base de données de lecteur.

```
mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-7448      |
+--------------------+

mysql> select count(*) from information_schema.TABLES;
+----------+
| count(*) |
```

```
+----------+
|      328 |
+----------+
```

Accès aux bases de données Aurora via VPCs

Par défaut, les composants de votre pile technologique Aurora sont tous dans le même VPC Amazon. Supposons, par exemple, qu'une application exécutée sur une EC2 instance Amazon se connecte à une instance de base de données de données Aurora. Dans ce cas, le serveur d'applications et la base de données doivent se trouver tous les deux dans le même VPC.

Avec RDS Proxy, vous pouvez configurer l'accès à une de données Aurora dans un VPC à partir des ressources d'un autre VPC, telles que des instances. EC2 Par exemple, votre organisation peut avoir plusieurs applications qui accèdent aux mêmes ressources de base de données. Chaque application peut se trouver dans son propre VPC.

Pour activer l'accès entre VPC, vous créez un nouveau point de terminaison pour le proxy. Le proxy lui-même réside dans le même VPC que le cluster de bases de données Aurora. Cependant, le point de terminaison inter-VPC réside dans l'autre VPC, avec les autres ressources telles que les instances. EC2 Le point de terminaison inter-VPC est associé à des sous-réseaux et à des groupes de sécurité provenant du même VPC que les autres ressources. EC2 Ces associations vous permettent de vous connecter au point de terminaison à partir des applications qui, autrement, ne peuvent pas accéder à la base de données en raison des restrictions de VPC.

Les étapes suivantes vous expliquent comment créer et accéder à un point de terminaison entre VPC via RDS Proxy :

1. Créez-en deux VPCs ou choisissez-en deux VPCs que vous utilisez déjà pour le travail avec Aurora Chaque VPC doit disposer de ses propres ressources réseau associées, par exemple une passerelle Internet, des tables de routage, des sous-réseaux et des groupes de sécurité. Si vous n'avez qu'un seul VPC, vous pouvez consulter Mise en route avec Amazon Aurora à propos des étapes de configuration d'un autre VPC à suivre pour utiliser Aurora avec succès. Vous pouvez également examiner votre VPC existant dans la EC2 console Amazon pour voir les types de ressources à connecter entre elles.
2. Créez un proxy de base de données associé au cluster de bases de données Aurora DB auquel vous voulez vous connecter. Suivez la procédure décrite dans Création d'un proxy pour Amazon Aurora.

3. Sur la page Details (Détails) de votre proxy dans la console RDS, dans la section Proxy endpoints (Points de terminaison proxy), cliquez sur Create endpoint (Créer un point de terminaison). Suivez la procédure décrite dans Création d’un point de terminaison proxy.
4. Choisissez de rendre le point de terminaison inter-VPC read/write ou de le rendre en lecture seule.
5. Au lieu d’accepter la valeur par défaut du même VPC que le cluster de bases de données Aurora, sélectionnez un autre VPC. Ce VPC doit se trouver dans la même AWS région que le VPC où réside le proxy.
6. Maintenant, au lieu d’accepter les valeurs par défaut pour les sous-réseaux et les groupes de sécurité du même VPC que le cluster de bases de données Aurora, effectuez de nouvelles sélections. Basez vos sélections sur les sous-réseaux et des groupes de sécurité du VPC que vous avez sélectionné.
7. Vous n’avez pas besoin de modifier les paramètres des secrets Secrets Manager. Les mêmes informations d’identification fonctionnent pour tous les points de terminaison de votre proxy, indépendamment du VPC dans lequel réside chaque point de terminaison. De même, lorsque vous utilisez l'authentification IAM, votre configuration et vos autorisations IAM fonctionnent de manière cohérente sur tous les points de terminaison du proxy, même lorsque les points de terminaison sont différents. VPCs Aucune configuration IAM supplémentaire n'est requise par point de terminaison.
8. Attendez que le nouveau point de terminaison atteigne l’état Available (Disponible).
9. Notez le nom complet du point de terminaison. Il s’agit de la valeur se terminant par *`Region_name`*`.rds.amazonaws.com` que vous fournissez dans le cadre de la chaîne de connexions pour votre application de base de données.
10. Accédez au nouveau point de terminaison à partir d’une ressource située dans le même VPC que le point de terminaison. Un moyen simple de tester ce processus consiste à créer une nouvelle EC2 instance dans ce VPC. Connectez-vous ensuite à l' EC2 instance et exécutez les `psql` commandes `mysql` or pour vous connecter en utilisant la valeur du point de terminaison dans votre chaîne de connexion.

Création d’un point de terminaison proxy

Suivez ces instructions pour créer un point de terminaison de proxy :

Console

Pour créer un point de terminaison proxy

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, sélectionnez Proxies.
3. Cliquez sur le nom du proxy pour lequel vous voulez créer un nouveau point de terminaison.

   La page de détails concernant ce proxy apparaît.
4. Dans la section Proxy endpoints (Points de terminaison proxy), cliquez sur Create proxy endpoint (Créer un point de terminaison proxy).

   La fenêtre Create proxy endpoint (Créer un point de terminaison proxy) apparaît.
5. Pour Proxy endpoint name (Nom du point de terminaison proxy), saisissez le nom descriptif de votre choix.
6. Pour le rôle cible, choisissez de définir le point de terminaison read/write ou de le rendre en lecture seule.

   Les connexions qui utilisent des read/write points de terminaison peuvent effectuer toutes sortes d'opérations, telles que des instructions en langage de définition des données (DDL), des instructions en langage de manipulation des données (DML) et des requêtes. Ces points de terminaison se connectent toujours à l'instance principale du cluster Aurora. Vous pouvez utiliser des read/write points de terminaison pour les opérations générales de base de données lorsque vous n'utilisez qu'un seul point de terminaison dans votre application. Vous pouvez également utiliser des read/write points de terminaison pour les opérations administratives, les applications de traitement des transactions en ligne (OLTP) et les tâches extract-transform-load (ETL).

   Les connexions qui utilisent un point de terminaison en lecture seule ne peuvent effectuer que des requêtes. Lorsque le cluster Aurora contient plusieurs instances de lecteur, RDS Proxy peut utiliser une instance de lecteur différente pour chaque connexion au point de terminaison. De cette façon, une application exigeante en requêtes peut tirer avantage de la capacité de clustering d'Aurora. Vous pouvez augmenter la capacité de requêtes du cluster en ajoutant d'autres instances de base de données de lecteur. Ces connexions en lecture seule n'imposent aucune surcharge à l'instance principale du cluster. Vos requêtes de reporting et d'analyse ne ralentissent donc pas les opérations d'écriture de vos applications OLTP.

7. Pour Virtual Private Cloud (VPC), choisissez la valeur par défaut pour accéder au point de terminaison à partir des mêmes EC2 instances ou d'autres ressources que celles utilisées normalement pour accéder au proxy ou à sa base de données associée. Pour configurer l'accès entre VPC pour ce proxy, choisissez un VPC autre que le VPC par défaut. Pour plus d'informations sur l'accès entre VPC, consultez Accès aux bases de données Aurora via VPCs.
8. Pour le Type de réseau de point de terminaison, choisissez la version IP du point de terminaison de proxy. Les options disponibles sont les suivantes :

   - IPv4— Le point de terminaison du proxy utilise uniquement IPv4 des adresses (par défaut).
   - IPv6— Le point de terminaison du proxy utilise uniquement IPv6 des adresses.
   - Double pile : le point de terminaison du proxy prend en charge à la fois les IPv6 adresses IPv4 et les adresses.

   Pour l'utiliser IPv6 ou le double-stack, votre VPC et vos sous-réseaux doivent être configurés pour prendre en charge le type de réseau sélectionné.
9. Pour Subnets (Sous-réseaux), RDS Proxy renseigne par défaut les mêmes sous-réseaux que le proxy associé. Pour restreindre l'accès au point de terminaison de sorte que seule une partie de la plage d'adresses du VPC puisse s'y connecter, supprimez un ou plusieurs sous-réseaux.
10. Pour Groupes de sécurité VPC, vous pouvez sélectionner un groupe de sécurité existant ou en créer un. RDS Proxy renseigne par défaut le ou les mêmes groupes de sécurité que le proxy associé. Si les règles entrantes et sortantes du proxy conviennent pour ce point de terminaison, conservez le choix par défaut.

    Si vous choisissez de créer un nouveau groupe de sécurité, donnez-lui un nom sur cette page. Modifiez ensuite les paramètres du groupe de sécurité depuis la EC2 console.
11. Cliquez sur Create proxy endpoint (Créer un point de terminaison proxy).

AWS CLI

Pour créer un point de terminaison proxy, utilisez la AWS CLI create-db-proxy-endpointcommande.

Incluez les paramètres requis suivants :

- `--db-proxy-name` *`value`*
- `--db-proxy-endpoint-name` *`value`*

- `--vpc-subnet-ids` *`list_of_ids`*. Séparez le sous-réseau IDs par des espaces. Vous n'avez pas à spécifier l'ID du VPC lui-même.

Vous pouvez également ajouter les paramètres facultatifs suivants :

- `--target-role { READ_WRITE | READ_ONLY }`. Ce paramètre a pour valeur par défaut `READ_WRITE`. La valeur `READ_ONLY` n'influe que sur les clusters provisionnés Aurora qui contiennent une ou plusieurs instances de base de données de lecteur. Lorsque le proxy est associé à un cluster Aurora contenant une seule instance de base de données d'enregistreur, vous ne pouvez pas spécifier `READ_ONLY`. Pour plus d'informations sur l'utilisation prévue des points de terminaison en lecture seule avec des clusters Aurora, consultez Utilisation des points de terminaison de lecteur avec les clusters Aurora .
- `--vpc-security-group-ids` *`value`*. Séparez le groupe de sécurité IDs par des espaces. Si vous omettez ce paramètre, RDS Proxy utilise le groupe de sécurité par défaut pour le VPC. Le proxy RDS détermine le VPC en fonction du IDs sous-réseau que vous spécifiez pour le paramètre. `--vpc-subnet-ids`
- `--endpoint-network-type { IPV4 | IPV6 | DUAL }`. Ce paramètre spécifie la version IP du point de terminaison du proxy. La valeur par défaut est `IPV4`. Pour utiliser `IPV6` ou `DUAL`, votre VPC et vos sous-réseaux doivent être configurés pour prendre en charge le type de réseau sélectionné.

Example

L'exemple suivant crée un point de terminaison proxy nommé `my-endpoint`.

Pour Linux, macOS ou Unix :

```
aws rds create-db-proxy-endpoint \
  --db-proxy-name my-proxy \
  --db-proxy-endpoint-name my-endpoint \
  --vpc-subnet-ids subnet_id subnet_id subnet_id ... \
  --target-role READ_ONLY \
  --vpc-security-group-ids security_group_id \
  --endpoint-network-type DUAL
```

Pour Windows :

```
aws rds create-db-proxy-endpoint ^
```

```
    --db-proxy-name my-proxy ^
    --db-proxy-endpoint-name my-endpoint ^
    --vpc-subnet-ids subnet_id_1 subnet_id_2 subnet_id_3 ... ^
    --target-role READ_ONLY ^
    --vpc-security-group-ids security_group_id ^
    --endpoint-network-type DUAL
```

API RDS

Pour créer un point de terminaison proxy, utilisez l'action Créer un point de DBProxy terminaison de l'API RDS.

Affichage des points de terminaison proxy

Pour afficher les points de terminaison proxy existants, procédez comme suit :

Console

Pour afficher les détails d’un point de terminaison proxy

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, sélectionnez Proxies.
3. Dans la liste, sélectionnez le proxy dont vous souhaitez afficher le point de terminaison. Cliquez sur le nom du proxy pour afficher la page contenant ses détails.
4. Dans la section Proxy endpoints (Points de terminaison proxy), sélectionnez le point de terminaison que vous voulez afficher. Cliquez sur son nom pour afficher la page contenant ses détails.
5. Examinez les paramètres dont les valeurs vous intéressent. Vous pouvez vérifier les propriétés suivantes :

    - Indique si le point de terminaison est en lecture/écriture ou en lecture seule.
    - Adresse de point de terminaison que vous utilisez dans une chaîne de connexions à la base de données.
    - Le VPC, les sous-réseaux et les groupes de sécurité associés au point de terminaison.

AWS CLI

Pour afficher un ou plusieurs points de terminaison proxy, utilisez la commande AWS CLIdescribe-db-proxy-endpoints.

Vous pouvez ajouter les paramètres facultatifs suivants :

- `--db-proxy-endpoint-name`
- `--db-proxy-name`

L'exemple suivant décrit le point de terminaison proxy `my-endpoint`.

Example

Pour Linux, macOS ou Unix :

```
aws rds describe-db-proxy-endpoints \
  --db-proxy-endpoint-name my-endpoint
```

Pour Windows :

```
aws rds describe-db-proxy-endpoints ^
  --db-proxy-endpoint-name my-endpoint
```

API RDS

Pour décrire un ou plusieurs points de terminaison proxy, exécutez l'opération de l'API RDS DescribeDBProxyEndpoints.

Modification d'un point de terminaison proxy

Pour modifier les points de terminaison de votre proxy, procédez comme suit :

Console

Pour modifier un ou plusieurs points de terminaison proxy

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, sélectionnez Proxies.

3. Dans la liste, sélectionnez le proxy dont vous voulez modifier le point de terminaison. Cliquez sur le nom du proxy pour afficher la page contenant ses détails.
4. Dans la section Proxy endpoints (Points de terminaison proxy), sélectionnez le point de terminaison que vous voulez modifier. Vous pouvez le sélectionner dans la liste ou cliquer sur son nom pour afficher la page contenant ses détails.
5. Sur la page de détails du proxy, dans la section Proxy endpoints (Points de terminaison proxy), cliquez sur Edit (Modifier). Ou bien, sur la page de détails du point de terminaison de proxy, choisissez Supprimer dans Actions.
6. Modifiez les valeurs des paramètres que vous souhaitez remplacer.
7. Sélectionnez Enregistrer les modifications.

AWS CLI

Pour modifier un point de terminaison proxy, utilisez la commande AWS CLI modify-db-proxy-endpoint avec les paramètres requis suivants :

- `--db-proxy-endpoint-name`

Précisez les modifications apportées aux propriétés du point de terminaison à l'aide d'un ou plusieurs des paramètres suivants :

- `--new-db-proxy-endpoint-name`
- `--vpc-security-group-ids`. Séparez les ID de groupe de sécurité par des espaces.

L'exemple suivant renomme le point de terminaison proxy `my-endpoint` à `new-endpoint-name`.

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint \
  --new-db-proxy-endpoint-name new-endpoint-name
```

Pour Windows :

```
aws rds modify-db-proxy-endpoint ^
```

```
  --db-proxy-endpoint-name my-endpoint ^
  --new-db-proxy-endpoint-name new-endpoint-name
```

API RDS

Pour modifier un point de terminaison proxy, exécutez l'opération de l'API RDS ModifyDBProxyEndPoint.

Suppression d'un point de terminaison proxy

Pour supprimer un point de terminaison de votre proxy, procédez comme suit :

> **Note**
>
> Vous ne pouvez pas supprimer le point de terminaison de proxy par défaut que RDS Proxy crée automatiquement pour chaque proxy.
> Lorsque vous supprimez un proxy, RDS Proxy supprime automatiquement tous les points de terminaison qui lui sont associés.

Console

Pour supprimer un point de terminaison proxy en utilisant AWS Management Console

1. Dans le panneau de navigation, sélectionnez Proxies.
2. Dans la liste, sélectionnez le proxy dont vous voulez supprimer un point de terminaison. Cliquez sur le nom du proxy pour afficher la page contenant ses détails.
3. Dans la section Proxy endpoints (Points de terminaison proxy), sélectionnez le point de terminaison que vous voulez supprimer. Vous pouvez sélectionner un ou plusieurs points de terminaison dans la liste ou cliquer sur le nom d'un seul point de terminaison pour afficher sa page de détails.
4. Sur la page de détails du proxy, dans la section Proxy endpoints (Points de terminaison proxy), cliquez sur Delete (Supprimer). Ou bien, sur la page de détails du point de terminaison de proxy, choisissez Supprimer dans Actions.

AWS CLI

Pour supprimer un point de terminaison proxy, exécutez la commande delete-db-proxy-endpoint avec les paramètres requis suivants :

- `--db-proxy-endpoint-name`

La commande suivante supprime le point de terminaison proxy nommé `my-endpoint`.

Pour Linux, macOS ou Unix :

```
aws rds delete-db-proxy-endpoint \
  --db-proxy-endpoint-name my-endpoint
```

Pour Windows :

```
aws rds delete-db-proxy-endpoint ^
  --db-proxy-endpoint-name my-endpoint
```

API RDS

Pour supprimer un point de terminaison proxy avec l'API RDS, exécutez l'opération DeleteDBProxyEndPoint. Spécifiez le nom du point de terminaison proxy pour le paramètre `DBProxyEndpointName`.

Surveillance des métriques RDS Proxy avec Amazon CloudWatch

Vous pouvez surveiller le proxy RDS à l'aide d'Amazon CloudWatch. CloudWatch collecte et convertit des données brutes de proxys en métriques lisibles en quasi-temps réel. Pour trouver ces mesures dans la console CloudWatch, sélectionnez Métriques, puis RDS et enfin Métriques par proxy. Pour plus d'informations, consultez Utilisation des métriques Amazon CloudWatch dans la section Guide de l'utilisateur Amazon CloudWatch.

> **Note**
>
> RDS publie ces métriques pour chaque instance Amazon EC2 sous-jacente associée au proxy. Un proxy unique peut être servi par plusieurs instances EC2. Utilisez les statistiques CloudWatch pour agréger les valeurs d'un proxy sur l'ensemble des instances associées. Certaines de ces métriques peuvent ne pas être visibles avant la première connexion réussie par un proxy.

Dans les journaux RDS Proxy, chaque entrée est préfixée avec le nom du point de terminaison proxy associé. Ce nom peut être celui que vous avez spécifié pour un point de terminaison défini

par l'utilisateur, ou le nom spécial `default` pour le point de terminaison par défaut d'un proxy qui effectue des demandes de lecture/écriture.

Toutes les métriques du proxy RDS sont dans le groupe `proxy`.

Chaque point de terminaison proxy possède ses propres métriques CloudWatch. Vous pouvez surveiller l'utilisation de chaque point de terminaison proxy individuellement. Pour plus d'informations sur les points de terminaison proxy, consultez Utilisation des points de terminaison du proxy Amazon RDS.

Vous pouvez agréger les valeurs de chaque métrique à l'aide de l'un des jeux de dimensions suivants. Par exemple, en utilisant le jeu de dimensions `ProxyName`, vous pouvez analyser l'ensemble du trafic d'un proxy particulier. En utilisant les autres jeux de dimensions, vous pouvez fractionner les métriques de différentes manières. Vous pouvez fractionner les métriques en fonction des différents points de terminaison ou bases de données cibles de chaque proxy, ou du trafic en lecture/écriture et en lecture seule vers chaque base de données.

- Ensemble de dimensions 1 : `ProxyName`
- Ensemble de dimensions 2 : `ProxyName`, `EndpointName`
- Ensemble de dimensions 3 : `ProxyName`, `TargetGroup`, `Target`
- Ensemble de dimensions 4 : `ProxyName`, `TargetGroup`, `TargetRole`

Métrique	Description	Période de validité	Jeu de dimensions CloudWatch
`Availabil ityPercentage`	Pourcentage de temps pendant lequel le groupe cible était disponible dans le rôle indiqué par la dimension. Cette métrique est donnée toutes les minutes. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute	Dimension set 4

Métrique	Description	Période de validité	Jeu de dimensions CloudWatch
`ClientConnections`	Le nombre actuel de connexions client. Cette métrique est donnée toutes les minutes. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute	Dimension set 1, Dimension set 2
`ClientConnectionsClosed`	Le nombre de connexions client fermées. La statistiq ue la plus utile pour cette métrique est `Sum`.	1 minute et plus	Dimension set 1, Dimension set 2
`ClientConnectionsInSetup`	Nombre actuel de connexions client ouvertes, mais qui n'ont pas terminé la configuration. Cette métrique est donnée toutes les minutes. Sum est la statistique la plus utile pour cette métrique.	1 minute	Dimension set 1, Dimension set 2

Métrique	Description	Période de validité	Jeu de dimensions CloudWatch
`ClientConnectionsNoTLS`	Nombre actuel de connexions client sans TLS (Transport Layer Security). Cette métrique est donnée toutes les minutes. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute	Dimension set 1, Dimension set 2
`ClientConnectionsReceived`	Le nombre de demandes de connexion client reçues. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 2
`ClientConnectionsSetupFailedAuth`	Nombre de tentatives de connexion client ayant échoué en raison d'une mauvaise configuration de l'authentification ou du protocole TLS. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 2

Métrique	Description	Période de validité	Jeu de dimensions CloudWatch
`ClientConnectionsSetupSucceeded`	Le nombre de connexions client établies avec succès via n'importe quel mécanisme d'authentification avec ou sans protocole TLS. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 2
`ClientConnectionsTLS`	Nombre actuel de connexions client avec TLS. Cette métrique est donnée toutes les minutes. La statistique la plus utile pour cette métrique est `Sum`.	1 minute	Dimension set 1, Dimension set 2
`DatabaseConnectionRequests`	Le nombre de demandes de création d'une connexion à une base de données. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 3, Dimension set 4

Métrique	Description	Période de validité	Jeu de dimensions CloudWatch
`DatabaseConnectionRequestsWithTLS`	Nombre de demandes de création d'une connexion à une base de données avec TLS. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnections`	Le nombre actuel de connexions à une base de données. Cette métrique est donnée toutes les minutes. La statistique la plus utile pour cette métrique est `Sum`.	1 minute	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsBorrowLatency`	Temps, en microsecondes, nécessaire au proxy surveillé pour obtenir une connexion à la base de données. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 2

Métrique	Description	Période de validité	Jeu de dimensions CloudWatch
`DatabaseConnectionsCurrentlyBorrowed`	Le nombre actuel de connexions à une base de données en état d'emprunt. Cette métrique est donnée toutes les minutes. La statistique la plus utile pour cette métrique est `Sum`.	1 minute	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsCurrentlyInTransaction`	Nombre actuel de connexions à la base de données dans une transaction. Cette métrique est donnée toutes les minutes. La statistique la plus utile pour cette métrique est `Sum`.	1 minute	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsCurrentlySessionPinned`	Nombre de connexions à la base de données actuellement épinglées en raison d'opérations dans les demandes client qui modifient l'état de session. Cette métrique est donnée toutes les minutes. La statistique la plus utile pour cette métrique est `Sum`.	1 minute	Dimension set 1, Dimension set 3, Dimension set 4

Métrique	Description	Période de validité	Jeu de dimensions CloudWatch
`DatabaseConnectionsSetupFailed`	Le nombre de demandes de connexion à une base de données qui ont échoué. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsSetupSucceeded`	Le nombre de connexions à une base de données correctement établies avec ou sans protocole TLS. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 3, Dimension set 4
`DatabaseConnectionsWithTLS`	Nombre actuel de connexions à une base de données avec TLS. Cette métrique est donnée toutes les minutes. La statistique la plus utile pour cette métrique est `Sum`.	1 minute	Dimension set 1, Dimension set 3, Dimension set 4

Métrique	Description	Période de validité	Jeu de dimensions CloudWatch
`MaxDatabaseConnectionsAllowed`	Le nombre maximal de connexions à une base de données autorisées. Cette métrique est donnée toutes les minutes. La statistique la plus utile pour cette métrique est `Sum`.	1 minute	Dimension set 1, Dimension set 3, Dimension set 4
`QueryDatabaseResponseLatency`	Temps, en microsecondes, pris par la base de données pour répondre à la requête. La statistique la plus utile pour cette métrique est `Average`.	1 minute et plus	Dimension set 1, Dimension set 2, Dimension set 3, Dimension set 4
`QueryRequests`	Le nombre de requêtes reçues. Une requête comprenant plusieurs instructions est comptée comme étant une seule et même requête. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 2

Métrique	Description	Période de validité	Jeu de dimensions CloudWatch
`QueryRequestsNoTLS`	Nombre de requêtes reçues de connexions non TLS. Une requête comprenant plusieurs instructions est comptée comme étant une seule et même requête. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 2
`QueryRequestsTLS`	Nombre de requêtes reçues des connexions TLS. Une requête comprenant plusieurs instructions est comptée comme étant une seule et même requête. est la statistique la plus utile pour cette métriqu `Sum`.	1 minute et plus	Dimension set 1, Dimension set 2
`QueryResponseLatency`	Temps, en microsecondes, entre l'obtention d'une demande de requête et le proxy qui y répond. La statistique la plus utile pour cette métrique est `Average`.	1 minute et plus	Dimension set 1, Dimension set 2

Vous pouvez trouver les journaux d'activité d proxy RDS sous CloudWatch dans la AWS Management Console. Chaque proxy dispose d'une entrée dans la page Groupes de journaux.

> **Important**
>
> Ces journaux sont destinés à la consommation humaine à des fins de dépannage et non à un accès par programmation. Le format et le contenu des journaux sont susceptibles d'être modifiés.
>
> En particulier, les journaux plus anciens ne contiennent pas de préfixes indiquant le point de terminaison pour chaque requête. Dans les journaux plus récents, chaque entrée est préfixée avec le nom du point de terminaison proxy associé. Ce nom peut être celui que vous avez spécifié pour un point de terminaison défini par l'utilisateur, ou le nom spécial `default` pour les demandes utilisant le point de terminaison par défaut d'un proxy.

Utilisation des des événements RDS Proxy

Un événement indique un changement dans un environment, comme un environnement AWS, un service ou une application d'un partenaire SaaS. Il peut s'agir aussi de l'un de vos propres applications ou services personnalisés. Par exemple, Amazon Aurora génère un événement lorsque vous créez ou modifiez un proxy RDS. Amazon Aurora fournit des événements à Amazon EventBridge en temps quasi-réel. Vous trouverez ci-dessous une liste des événements RDS Proxy auxquels vous pouvez vous abonner et un exemple d'événement RDS Proxy.

Pour plus d'informations sur l'utilisation des événements, consultez les éléments suivants :

- Pour obtenir des instructions sur l'affichage des événements à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API RDS, consultez Affichage d'événements Amazon RDS.
- Pour savoir comment configurer Amazon Aurora pour l'envoi d'événements à EventBridge, consultez Création d'une règle qui se déclenche sur un événement Amazon Aurora.

Événements RDS Proxy

Le tableau suivant recense la catégorie d'événement et la liste des événements lorsqu'un proxy RDS Proxy est le type source.

Catégorie	ID d'événement RDS	Message	Remarques
modification de configuration	RDS-EVENT-0204	RDS a modifié le proxy de base de données *nom*.	Aucun
modification de configuration	RDS-EVENT-0207	RDS a modifié le point de terminaison du proxy de base de données *nom*.	Aucun
modification de configuration	RDS-EVENT-0213	RDS a détecté l'ajout de l'instance de base de données et l'a automatiquement ajoutée au groupe cible du proxy de base de données *nom*.	Aucun
modification de configuration	RDS-EVENT-0214	RDS a détecté la suppression de l'instance de base de données *nom* et l'a automatiquement supprimée du groupe cible *nom* du proxy de base de données *nom*.	Aucun
modification de configuration	RDS-EVENT-0215	RDS a détecté la suppression du cluster de bases de données *nom* et l'a automatiquement supprimée du groupe cible *nom* du proxy de base de données *nom*.	Aucun
création	RDS-EVENT-0203	RDS a créé le proxy de base de données *nom*.	Aucun

Catégorie	ID d'événement RDS	Message	Remarques
création	RDS-EVENT-0206	RDS a créé le point de terminaison *nom* pour le proxy de base de données *nom*.	Aucun
suppression	RDS-EVENT-0205	RDS a supprimé le proxy de base de données *nom*.	Aucun
suppression	RDS-EVENT-0208	RDS a supprimé le point de terminaison *nom* pour le proxy de base de données *nom*.	Aucun
échec	RDS-EVENT-0243	RDS n'a pas pu allouer la capacité pour le proxy *nom*, car il n'y a pas suffisamment d'adresses IP disponibles dans vos sous-réseaux : *nom*. Pour résoudre ce problème, veillez à ce que vos sous-réseaux aient le nombre minimum d'adresses IP inutilisées, comme recommandé dans la documentation Proxy RDS.	Pour déterminer le nombre recommandé pour votre classe d'instances, consultez Planification de la capacité des adresses IP.
échec	RDS-EVENT-0275	RDS a limité certaines connexions au proxy de base de données *nom*. Le nombre de demandes de connexion simultanée du client au proxy a dépassé la limite.	Aucun

Voici un exemple d'événement RDS Proxy au format JSON. L'événement montre que RDS a modifié le point de terminaison nommé `my-endpoint` du proxy RDS nommé `my-rds-proxy`. L'ID de l'événement est RDS-EVENT-0207.

```
{
  "version": "0",
  "id": "68f6e973-1a0c-d37b-f2f2-94a7f62ffd4e",
  "detail-type": "RDS DB Proxy Event",
  "source": "aws.rds",
  "account": "123456789012",
  "time": "2018-09-27T22:36:43Z",
  "region": "us-east-1",
  "resources": [
     "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy"
  ],
  "detail": {
    "EventCategories": [
      "configuration change"
    ],
    "SourceType": "DB_PROXY",
    "SourceArn": "arn:aws:rds:us-east-1:123456789012:db-proxy:my-rds-proxy",
    "Date": "2018-09-27T22:36:43.292Z",
    "Message": "RDS modified endpoint my-endpoint of DB Proxy my-rds-proxy.",
    "SourceIdentifier": "my-endpoint",
    "EventID": "RDS-EVENT-0207"
  }
}
```

Exemples de ligne de commande pour le proxy RDS

Pour voir comment les combinaisons de commandes de connexion et d'instructions SQL interagissent avec RDS Proxy, observez les exemples suivants.

Exemples

- Preserving Connections to a MySQL Database Across a Failover
- Adjusting the max_connections Setting for an Aurora DB Cluster

Example Conservation des connexions à une base de données MySQL lors d'un basculement

Cet exemple MySQL montre comment les connexions ouvertes continuent de fonctionner pendant un basculement. Par exemple, lorsque vous redémarrez une base de données ou qu'un problème la rend indisponible. Cet exemple utilise un proxy nommé `the-proxy` et un cluster de base de données Aurora avec des instances de base de données `instance-8898` et `instance-9814`. Lorsque vous exécutez la commande `failover-db-cluster` à partir de la ligne de commande Linux, l'instance de rédacteur à laquelle le proxy est connecté change d'instance de base de données. Vous pouvez voir que l'instance de base de données associée au proxy change pendant que la connexion est ouverte.

```
$ mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com -u admin_user -p
Enter password:
...

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-9814      |
+--------------------+
1 row in set (0.01 sec)

mysql>
[1]+  Stopped                 mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com
 -u admin_user -p
$ # Initially, instance-9814 is the writer.
$ aws rds failover-db-cluster --db-cluster-identifier cluster-56-2019-11-14-1399
JSON output
$ # After a short time, the console shows that the failover operation is complete.
$ # Now instance-8898 is the writer.
$ fg
mysql -h the-proxy.proxy-demo.us.us-east-1.rds.amazonaws.com -u admin_user -p

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-8898      |
+--------------------+
1 row in set (0.01 sec)
```

```
mysql>
[1]+  Stopped                 mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com
 -u admin_user -p
$ aws rds failover-db-cluster --db-cluster-identifier cluster-56-2019-11-14-1399
JSON output
$ # After a short time, the console shows that the failover operation is complete.
$ # Now instance-9814 is the writer again.
$ fg
mysql -h the-proxy.proxy-demo.us-east-1.rds.amazonaws.com -u admin_user -p

mysql> select @@aurora_server_id;
+--------------------+
| @@aurora_server_id |
+--------------------+
| instance-9814      |
+--------------------+
1 row in set (0.01 sec)
+---------------+---------------+
| Variable_name | Value         |
+---------------+---------------+
| hostname      | ip-10-1-3-178 |
+---------------+---------------+
1 row in set (0.02 sec)
```

Example Réglage du paramètre max_connections pour un cluster de bases de données Aurora

Cet exemple montre comment ajuster le paramètre `max_connections` pour un cluster de base de données Aurora MySQL. Pour ce faire, vous créez votre propre groupe de paramètres de cluster de base de données en fonction des paramètres par défaut des clusters compatibles avec MySQL 5.7. Vous indiquez une valeur pour le paramètre `max_connections`, en remplaçant la formule qui définit la valeur par défaut. Vous associez le groupe de paramètres de cluster de base de données à votre cluster de base de données.

```
export REGION=us-east-1
export CLUSTER_PARAM_GROUP=rds-proxy-mysql-57-max-connections-demo
export CLUSTER_NAME=rds-proxy-mysql-57

aws rds create-db-parameter-group --region $REGION \
  --db-parameter-group-family aurora-mysql5.7 \
  --db-parameter-group-name $CLUSTER_PARAM_GROUP \
  --description "Aurora MySQL 5.7 cluster parameter group for RDS Proxy demo."

aws rds modify-db-cluster --region $REGION \
```

```
  --db-cluster-identifier $CLUSTER_NAME \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP

echo "New cluster param group is assigned to cluster:"
aws rds describe-db-clusters --region $REGION \
  --db-cluster-identifier $CLUSTER_NAME \
  --query '*[*].{DBClusterParameterGroup:DBClusterParameterGroup}'

echo "Current value for max_connections:"
aws rds describe-db-cluster-parameters --region $REGION \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP \
  --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
  --output text | grep "^max_connections"

echo -n "Enter number for max_connections setting: "
read answer

aws rds modify-db-cluster-parameter-group --region $REGION --db-cluster-parameter-
group-name $CLUSTER_PARAM_GROUP \
  --parameters "ParameterName=max_connections,ParameterValue=$
$answer,ApplyMethod=immediate"

echo "Updated value for max_connections:"
aws rds describe-db-cluster-parameters --region $REGION \
  --db-cluster-parameter-group-name $CLUSTER_PARAM_GROUP \
  --query '*[*].{ParameterName:ParameterName,ParameterValue:ParameterValue}' \
  --output text | grep "^max_connections"
```

Résolution des problèmes liés au proxy RDS

Vous trouverez ci-dessous des idées pour résoudre certains problèmes courants liés au proxy RDS et des informations sur les CloudWatch journaux du proxy RDS.

Dans les journaux RDS Proxy, chaque entrée est préfixée avec le nom du point de terminaison proxy associé. Ce nom peut être celui que vous avez spécifié pour un point de terminaison défini par l'utilisateur. Il peut également s'agir du nom `default` spécial du point de terminaison par défaut d'un proxy qui exécute les read/write demandes. Pour plus d'informations sur les points de terminaison proxy, consultez Utilisation des points de terminaison du proxy Amazon RDS.

Rubriques

- Vérification de la connectivité pour un proxy

- Problèmes courants et solutions correspondantes
- Dépannage des problèmes liés au proxy RDS avec RDS for MySQL
- Dépannage des problèmes liés au proxy RDS pour PostgreSQL

Vérification de la connectivité pour un proxy

Vous pouvez utiliser les commandes suivantes pour vérifier que tous les composants tels que le proxy, la base de données et les instances de calcul de la connexion peuvent communiquer entre eux.

Examinez le proxy lui-même à l'aide de la describe-db-proxiescommande. Examinez également le groupe cible associé à l'aide de la commande describe-db-proxy-target-groups. Vérifiez que les détails des cibles correspondent au cluster de bases de données Aurora que vous prévoyez d'associer au proxy. Utilisez des commandes telles que les suivantes.

```
aws rds describe-db-proxies --db-proxy-name $DB_PROXY_NAME
aws rds describe-db-proxy-target-groups --db-proxy-name $DB_PROXY_NAME
```

Pour vérifier que le proxy peut se connecter à la base de données sous-jacente, examinez les cibles spécifiées dans les groupes cibles à l'aide de la describe-db-proxy-targetscommande. Utilisez une commande telle que la suivante.

```
aws rds describe-db-proxy-targets --db-proxy-name $DB_PROXY_NAME
```

La sortie de la describe-db-proxy-targetscommande inclut un `TargetHealth` champ. Vous pouvez examiner les champs `State`, `Reason` et `Description` dans `TargetHealth` pour vérifier si le proxy peut communiquer avec l'instance de base de données sous-jacente.

- Si la valeur `State` est `AVAILABLE`, cela indique que le proxy peut se connecter à l'instance de base de données.
- Si la valeur `State` est `UNAVAILABLE`, cela signale un problème de connexion temporaire ou permanent. Dans ce cas, examinez les champs `Reason` et `Description`. Par exemple, si `Reason` a une valeur `PENDING_PROXY_CAPACITY`, essayez de vous connecter à nouveau une fois que le proxy a terminé son opération de mise à l'échelle. Si `Reason` a une valeur `UNREACHABLE`, `CONNECTION_FAILED` ou `AUTH_FAILURE`, utilisez l'explication du champ `Description` pour vous aider à diagnostiquer le problème.

- Le champ `State` peut avoir une valeur `REGISTERING` pendant une courte période avant de passer à `AVAILABLE` ou `UNAVAILABLE`.

Si la commande Netcat (`nc`) suivante aboutit, vous pouvez accéder au point de terminaison du proxy depuis l' EC2 instance ou un autre système sur lequel vous êtes connecté. Cette commande signale un échec si vous n'êtes pas dans le même VPC que le proxy et la base de données associée. Vous pouvez peut-être vous connecter directement à la base de données sans vous trouver dans le même VPC. Cependant, vous ne pouvez pas vous connecter au proxy sauf si vous êtes dans le même VPC.

```
nc -zx MySQL_proxy_endpoint 3306

nc -zx PostgreSQL_proxy_endpoint 5432
```

Vous pouvez utiliser les commandes suivantes pour vous assurer que votre EC2 instance possède les propriétés requises. En particulier, le VPC de l' EC2 instance doit être le même que le VPC du cluster Aurora de l'instance de base de données RDS auquel le proxy se connecte.

```
aws ec2 describe-instances --instance-ids your_ec2_instance_id
```

Examinez les secrets Secrets Manager utilisés pour le proxy.

```
aws secretsmanager list-secrets
aws secretsmanager get-secret-value --secret-id your_secret_id
```

Assurez-vous que le champ `SecretString` affiché par `get-secret-value` est codé au format d'une chaîne JSON qui inclut les champs `username` et `password`. L'exemple suivant illustre le format du champ `SecretString`.

```
{
  "ARN": "some_arn",
  "Name": "some_name",
  "VersionId": "some_version_id",
  "SecretString": '{"username":"some_username","password":"some_password"}',
  "VersionStages": [ "some_stage" ],
  "CreatedDate": some_timestamp
}
```

Lorsque vous résolvez des problèmes d'authentification IAM, vérifiez les points suivants :

- L'authentification IAM est activée dans la base de données.
- Le proxy est configuré avec le schéma d'authentification approprié.
- Les politiques IAM du rôle IAM fourni au proxy accordent les `rds-db:connect` autorisations nécessaires à la base de données appropriée et à son nom d'utilisateur.
- Pour l'authentification end-to-end IAM, il existe des utilisateurs de base de données qui correspondent aux noms d'utilisateur ou de rôle IAM.
- Le protocole SSL/TLS est activé pour la connexion.

Problèmes courants et solutions correspondantes

Cette section décrit certains problèmes courants et les solutions potentielles lors de l’utilisation du proxy RDS.

Après avoir exécuté la commande CLI`aws rds describe-db-proxy-targets`, si la description `TargetHealth` indique `Proxy does not have any registered credentials`, vérifiez les points suivants :

- Des informations d’identification sont enregistrées pour permettre à l’utilisateur d’accéder au proxy.
- Le rôle IAM permettant d’accéder au secret Secrets Manager utilisé par le proxy est valide.

Vous pouvez rencontrer les événements RDS suivants lors de la connexion à un proxy de base de données ou de sa création.

Catégorie	ID d’événement RDS	Description
échec	RDS-EVENT-0243	RDS n’a pas pu allouer la capacité pour le proxy, car il n’y a pas suffisamment d’adresses IP disponibles dans vos sous-réseaux. Pour résoudre ce problème, veillez à ce que vos sous-réseaux aient le nombre minimum d’adresses IP inutilisées. Pour déterminer le nombre recommandé pour votre

Catégorie	ID d'événement RDS	Description
		classe d'instances, consultez Planification de la capacité des adresses IP.
échec	RDS-EVENT-0275	RDS a limité certaines connexions au proxy de base de données. *name* Le nombre de demandes de connexion simultanée du client au proxy a dépassé la limite.

Vous pouvez rencontrer les problèmes suivants lors de la création d'un proxy ou de la connexion à un proxy.

Erreur	Causes ou solutions de contournement
`403: The security token included in the request is invalid`	Sélectionnez un rôle IAM existant au lieu d'en créer un nouveau.

Dépannage des problèmes liés au proxy RDS avec RDS for MySQL

Vous pouvez rencontrer les problèmes suivants lors de la connexion à un proxy MySQL.

Erreur	Causes ou solutions de contournement
`ERROR 1040 (HY000): Connections rate limit exceeded (`*`limit_value`*` )`	Le taux de demandes de connexion du client au proxy a dépassé la limite.

Erreur	Causes ou solutions de contournement
`ERROR 1040 (HY000): IAM authentication rate limit exceeded`	Le nombre de demandes de connexion simultanée avec authentification IAM du client au proxy a dépassé la limite.
`ERROR 1040 (HY000): Number simultaneous connections exceeded (`*`limit_value`*` )`	Le nombre de demandes de connexion simultanée du client au proxy a dépassé la limite.
`ERROR 1045 (28000): Access denied for user '`*`DB_USER`*`'@'%' (usi password: YES)`	Le secret Secrets Manager utilisé par le proxy ne correspond pas au nom d'utilisateur et au mot de passe d'un utilisateur de base de données existant. Mettez à jour les informations d'identification dans le secret Secrets Manager ou assurez-vous que l'utilisateur de base de données existe et possède le même mot de passe que celui du secret.
`ERROR 1105 (HY000): Unknown error`	Une erreur inconnue s'est produite.
`ERROR 1231 (42000): Variable ''character_set_client'' can't be set to the value of `*`value`*	La valeur définie pour le paramètre `character_set_client` n'est pas valide. Par exemple, la valeur `ucs2` n'est pas valide, car elle peut bloquer le serveur MySQL.

Erreur	Causes ou solutions de contournement
`ERROR 3159 (HY000): This RDS Proxy requires TLS connections.`	Vous avez activé le paramètre Exiger la sécurité de la couche de transport dans le proxy, mais votre connexion a inclus le paramètre `ssl-mode=DISABLED` dans le client MySQL. Effectuez l'une des actions suivantes : • Désactivez le paramètre Exiger la sécurité de la couche de transport pour le proxy. • Connectez-vous à la base de données en utilisant le paramètre minimal de `ssl-mode=REQUIRED` dans le client MySQL.
`ERROR 2026 (HY000): SSL connection error: Internal Server` *`Error`*	La négociation TLS avec le proxy a échoué. Les causes possibles sont notamment les suivantes : • SSL est requis, mais le serveur ne le prend pas en charge. • Une erreur interne du serveur s'est produite. • Une mauvaise négociation s'est produite.
`ERROR 9501 (HY000): Timed-out waiting to acquire database connection`	Le proxy a expiré en attendant l'obtention d'une connexion à la base de données. Les causes possibles sont notamment les suivantes : • Le proxy n'est pas en mesure d'établir une connexion à la base de données, car le nombre maximal de connexions a été atteint. • Le proxy n'est pas en mesure d'établir une connexion à la base de données, car la base de données n'est pas disponible.

Dépannage des problèmes liés au proxy RDS pour PostgreSQL

Vous pouvez rencontrer les problèmes suivants lors de la connexion à un proxy PostgreSQL.

Erreur	Cause	Solution
`ERROR 28000: IAM authentication is allowed only with SSL connections.`	L'utilisateur a essayé de se connecter à la base de données à l'aide de l'authentification IAM en utilisant	L'utilisateur doit se connecter à la base de données en utilisant le paramètre minimal `sslmode=require` du

Erreur	Cause	Solution
	le paramètre `sslmode=disable` du client PostgreSQL.	client PostgreSQL. Pour plus d'informations, consultez la documentation PostgreSQL SSL Support.
`ERROR 28000: This RDS proxy has no credentials for the role` *`role_name`* `. Check the credentials for this role and try again.`	Il n'y a pas de secret Secrets Manager pour ce rôle.	Ajoutez un secret Secrets Manager pour ce rôle. Pour plus d'informations, consultez Configuration de l'authentification IAM pour RDS Proxy.
`ERROR 28000: RDS supports only IAM, MD5, or SCRAM authentication.`	Le client de base de données utilisé pour se connecter au proxy utilise un mécanisme d'authentification qui n'est actuellement pas pris en charge par le proxy.	Si vous n'utilisez pas l'authentification IAM, utilisez l'authentification par mot de passe MD5 ou SCRAM.
`ERROR 28000: A user name is missing from the connection startup packet. Provide a user name for this connection.`	Le client de base de données utilisé pour la connexion au proxy n'envoie pas de nom d'utilisateur lorsqu'il tente d'établir une connexion.	Veillez à définir un nom d'utilisateur lors de la configuration d'une connexion au proxy à l'aide du client PostgreSQL de votre choix.
`ERROR 28000: IAM is allowed only with SSL connections.`	Un client a essayé de se connecter à l'aide de l'authentification IAM, mais le protocole SSL n'était pas activé.	Activez SSL sur le client PostgreSQL.

Erreur	Cause	Solution
`ERROR 28000: This RDS Proxy requires TLS connections.`	L'utilisateur a activé l'option Exiger la sécurité de la couche de transport mais a essayé de se connecter en utilisant `sslmode=disable` dans le client PostgreSQL.	Pour corriger cette erreur, effectuez l'une des opérations suivantes : • Désactivez l'option Exiger la sécurité de la couche de transport du proxy. • Connectez-vous à la base de données en utilisant le paramètre minimal `sslmode=allow` du client PostgreSQL.
`ERROR 28P01: IAM authentication failed for user` *`user_name`* `. Check the IAM token for this user and try again.`	Cette erreur peut être due aux raisons suivantes : • Le client a indiqué un nom d'utilisateur IAM incorrect. • Le client a fourni un jeton d'autorisation IAM incorrect pour l'utilisateur • Le client utilise une politique IAM qui ne dispose pas des autorisations nécessaires. • Le client a fourni un jeton d'autorisation IAM expiré pour l'utilisateur.	Pour corriger cette erreur, procédez comme suit : 1. Vérifiez que l'utilisateur IAM indiqué existe. 2. Vérifiez que le jeton d'autorisation IAM appartient à l'utilisateur IAM indiqué. 3. Vérifiez que la politique IAM dispose des autorisations adéquates pour RDS. 4. Vérifiez la validité du jeton d'autorisation IAM utilisé.
`ERROR 28P01: The password that was provided for the role` *`role_name`* `is wrong.`	Le mot de passe de ce rôle ne correspond pas au secret Secrets Manager.	Vérifiez le secret de ce rôle dans Secrets Manager pour voir si le mot de passe est le même que celui utilisé sur votre client PostgreSQL.

Erreur	Cause	Solution
`ERROR 28P01: The IAM authentication failed for the role` *`role_name`* `. Check the IAM token for this role and try again.`	Il y a un problème avec le jeton IAM utilisé pour l'authentification IAM.	Générez un nouveau jeton d'authentification et utilisez-le dans une nouvelle connexion.
`ERROR 0A000: Feature not supported: RDS Proxy supports only version 3.0 of the PostgreSQL messaging protocol.`	Le client PostgreSQL utilisé pour la connexion au proxy utilise un protocole antérieur à la version 3.0.	Utilisez un client PostgreSQL plus récent qui prend en charge le protocole de messagerie 3.0. Si vous utilisez l'interface de ligne de commande `psql` de PostgreSQL, utilisez une version supérieure ou égale à 7.4.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support streaming replication mode.`	Le client PostgreSQL utilisé pour la connexion au proxy essaie d'utiliser le mode de réplication de streaming, lequel n'est actuellement pas pris en charge par le proxy RDS.	Désactivez le mode de réplication de streaming sur le client PostgreSQL utilisé pour la connexion.
`ERROR 0A000: Feature not supported: RDS Proxy currently doesn't support the option` *`option_name`* `.`	Par le biais du message de démarrage, le client PostgreSQL utilisé pour la connexion au proxy demande une option qui n'est pas actuellement prise en charge par le proxy RDS.	Désactivez l'option indiquée comme non prise en charge dans le message ci-dessus sur le client PostgreSQL utilisé pour la connexion.

Erreur	Cause	Solution
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	Le nombre de demandes de connexion simultanée avec authentification IAM du client au proxy a dépassé la limite.	Réduisez le taux d'établis sement de connexions à l'aide de l'authentification IAM à partir d'un client PostgreSQL.
`ERROR 53300: The maximum number of client connections to the proxy exceeded` *`number_value`* `.`	Le nombre de demandes de connexion simultanée du client au proxy a dépassé la limite.	Réduisez le nombre de connexions actives des clients PostgreSQL à ce proxy RDS.
`ERROR 53300: Rate of connection to proxy exceeded` *`number_va lue`* `.`	Le taux de demandes de connexion du client au proxy a dépassé la limite.	Réduisez le taux d'établis sement de connexions à partir d'un client PostgreSQL.
`ERROR XX000: Unknown error.`	Une erreur inconnue s'est produite.	Contactez le AWS Support pour étudier le problème.

Erreur	Cause	Solution
`ERROR 08000: Timed-out waiting to acquire database connection.`	Le proxy a expiré en attendant l'obtention d'une connexion à la base de données pendant la durée spécifiée par le paramètre `ConnectionBorrowTimeout` . Les causes possibles sont notamment les suivantes : • Le proxy ne peut pas établir une connexion à la base de données, car le nombre maximal de connexions a été atteint. • Le proxy ne peut pas établir une connexion à la base de données, car la base de données n'est pas disponible ou l'établissement de la connexion à la base de données prend plus de temps que le délai configuré par le paramètre `ConnectionBorrowTimeout` .	Les solutions possibles sont les suivantes : • Évitez d'épingler les connexions proxy. Consultez Contournement de l'épinglage d'un proxy RDS. • Passez en revue les paramètres `ConnectionBorrowTimeout` et `MaxConnectionsPercent` . Consultez Considérations relatives à la connexion RDS Proxy. • Vérifiez la disponibilité des cibles. Consultez `AvailabilityPercentage` dans Surveillance des métriques RDS Proxy avec Amazon CloudWatch.

Erreur	Cause	Solution
`ERROR XX000: Request returned an error: database_error .`	La connexion à la base de données établie à partir du proxy a renvoyé une erreur.	La solution dépend de l'erreur de base de données spécifique. Par exemple : `Request returned an error: database "your-database-name" does not exist`. Cela signifie que le nom de base de données spécifié n'existe pas sur le serveur de base de données. Ou cela signifie que le nom d'utilisateur utilisé comme nom de base de données (si un nom de base de données n'est pas spécifié) n'existe pas sur le serveur.
`ERROR 53300: The IAM authentication failed because of too many competing requests.`	Le nombre de demandes de connexion simultanée avec authentification IAM du client au proxy a dépassé la limite.	Réduisez le taux d'établissement de connexions à l'aide de l'authentification IAM à partir d'un client PostgreSQL.
`ERROR 28000: Enable IAM authentication for the client connection to the proxy and try again.`	Le proxy RDS ne peut pas se connecter à la base de données car l'authentification IAM n'est pas activée pour la connexion du client au proxy. Cela se produit lorsque le `DefaultAuthScheme` paramètre du proxy est défini sur un utilisateur enregistré, mais que le client utilise l'authentification par mot de passe au lieu de l'authentification IAM. `IAM_AUTH`	Activez l'authentification IAM pour la connexion du client au proxy et réessayez.

Erreur	Cause	Solution
`ERROR 28000: Configure IAM authentication as the DefaultAuthScheme in your proxy and try again.`	Le proxy RDS ne peut pas se connecter à la base de données car elle `DefaultAuthScheme` n'est pas définie sur. `IAM_AUTH` Le `DefaultAuthScheme` paramètre du proxy est défini sur`NONE`, mais le client tente d'utiliser l'authentification IAM.	Définissez `DefaultAuthScheme` ce `IAM_AUTH` paramètre pour votre proxy et réessayez.

Dépannage des problèmes liés à la base de données **`postgres`**

Si vous supprimez par erreur la base de données `postgres` de votre instance, vous devez la restaurer pour rétablir la connectivité à votre instance. Exécutez les commandes suivantes dans votre instance de base de données :

```
CREATE DATABASE postgres;
GRANT CONNECT ON DATABASE postgres TO rdsproxyadmin;
```

Utilisation de RDS Proxy avec AWS CloudFormation

Vous pouvez utiliser RDS Proxy avec AWS CloudFormation. Vous pouvez ainsi créer des groupes de ressources connexes. Un tel groupe peut inclure un proxy qui peut se connecter à un cluster de base de données Aurora que vous venez de créer. La prise en charge de RDS Proxy dans CloudFormation implique deux nouveaux types de registres : `DBProxy` et `DBProxyTargetGroup`.

La liste suivante présente un exemple de modèle CloudFormation pour RDS Proxy.

```
Resources:
 DBProxy:
   Type: AWS::RDS::DBProxy
   Properties:
     DBProxyName: CanaryProxy
     EngineFamily: MYSQL
     RoleArn:
      Fn::ImportValue: SecretReaderRoleArn
```

```
    Auth:
      - {AuthScheme: SECRETS, SecretArn: !ImportValue ProxySecret, IAMAuth: DISABLED}
    VpcSubnetIds:
      Fn::Split: [",", "Fn::ImportValue": SubnetIds]

ProxyTargetGroup:
  Type: AWS::RDS::DBProxyTargetGroup
  Properties:
    DBProxyName: CanaryProxy
    TargetGroupName: default
    DBInstanceIdentifiers:
      - Fn::ImportValue: DBInstanceName
  DependsOn: DBProxy
```

Pour plus d'informations sur les ressources de cet exemple, consultez `DBProxy` et `DBProxyTargetGroup`.

Pour plus d'informations sur les ressources que vous pouvez créer avec CloudFormation, consultez Référence de type de ressource RDS.

Utilisation du proxy RDS avec les bases de données globales Aurora

Une base de données globale Aurora est une base de données unique couvrant plusieurs Régions AWS, ce qui permet des lectures globales à faible latence et une reprise après sinistre en cas de panne dans une région quelconque. Elle fournit une tolérance de panne intégrée pour votre déploiement, car l'instance de base de données ne repose pas sur une seule Région AWS, mais sur plusieurs régions et différentes zones de disponibilité. Pour de plus amples informations, consultez Utilisation d'Amazon Aurora Global Database.

Vous pouvez utiliser le proxy RDS avec n'importe quel cluster de bases de données dans une base de données globale Aurora. Avant de commencer à utiliser ces fonctionnalités ensemble, veillez à vous familiariser avec les informations suivantes.

> **Important**
>
> Si le cluster de bases de données fait partie d'une base de données globale où le transfert d'écriture est activé, réduisez la valeur `MaxConnectionsPercent` de votre proxy du quota alloué au transfert d'écriture. Le quota de transfert d'écriture est défini dans le paramètre

de cluster de base de données `aurora_fwd_writer_max_connections_pct`. Pour de plus amples informations sur le transfert d'écriture, veuillez consulter Utilisation du transfert d'écriture dans une base de données globale Amazon Aurora.

Limites pour le proxy RDS avec les bases de données globales

Lorsque le transfert d'écriture est activé dans le cluster de bases de données Aurora, le proxy RDS ne prend pas en charge la valeur `SESSION` de la variable `aurora_replica_read_consistency`. La définition de cette valeur peut entraîner un comportement inattendu.

Fonctionnement des points de terminaison du proxy RDS avec les bases de données globales

Lorsque vous comprenez le fonctionnement des points de terminaison du proxy RDS avec les bases de données globales, vous pouvez mieux gérer vos applications qui utilisent des bases de données Aurora avec ces deux fonctionnalités.

Pour un proxy dont le cluster principal d'une base de données globale est la cible enregistrée, les points de terminaison du proxy fonctionnent de la même manière qu'avec n'importe quel cluster de bases de données Aurora. Les points de terminaison de lecture/écriture du proxy envoient toutes les demandes à l'instance d'écriture du cluster. Les points de terminaison en lecture seule du proxy envoient toutes les demandes aux instances de lecture. Si un lecteur devient indisponible alors qu'une connexion est ouverte, le proxy RDS redirige les requêtes suivantes sur la connexion vers une autre instance de lecture. Pour un proxy doté d'un cluster secondaire comme cible enregistrée, les demandes envoyées aux points de terminaison en lecture seule du proxy sont également envoyées aux instances de lecture. Comme le cluster ne possède aucune instance d'écriture, les demandes envoyées aux points de terminaison de lecture/écriture échouent avec l'erreur « `The target group doesn't have any associated read/write instances` ».

Les opérations globales de basculement et de commutation des bases de données impliquent toutes deux un changement de rôle entre le cluster de bases de données principal et l'un des clusters de bases de données secondaires. Lorsque le cluster secondaire sélectionné devient le nouveau cluster principal, l'une de ses instances de lecture est promue processus d'écriture. Cette instance de base de données est désormais la nouvelle instance d'écriture pour le cluster global. Veillez à rediriger les opérations d'écriture de votre application vers le point de terminaison de lecture/écriture approprié du proxy qui est associé au nouveau cluster principal. Ce point de terminaison du proxy peut être le point de terminaison par défaut ou un point de terminaison de lecture/écriture personnalisé.

Le proxy RDS met en file d'attente toutes les demandes via les points de terminaison de lecture/écriture et les envoie à l'instance d'écriture du nouveau cluster principal dès qu'il est disponible. Il le fait indépendamment du fait que l'opération de basculement ou de commutation soit terminée ou non. Pendant le basculement ou la commutation, le point de terminaison par défaut du proxy de l'ancien cluster principal continue d'accepter les opérations d'écriture. Toutefois, dès que ce cluster devient un cluster secondaire, toutes les opérations d'écriture échouent. Pour savoir comment et quand effectuer des tâches spécifiques de basculement ou de commutation globale, consultez les rubriques suivantes :

- Commutation globale des bases de données – Réalisation de bascules pour les bases de données Amazon Aurora Global Database
- Basculement global de la base de données – Reprise d’une base de données Amazon Aurora globale à partir d’une panne non planifiée

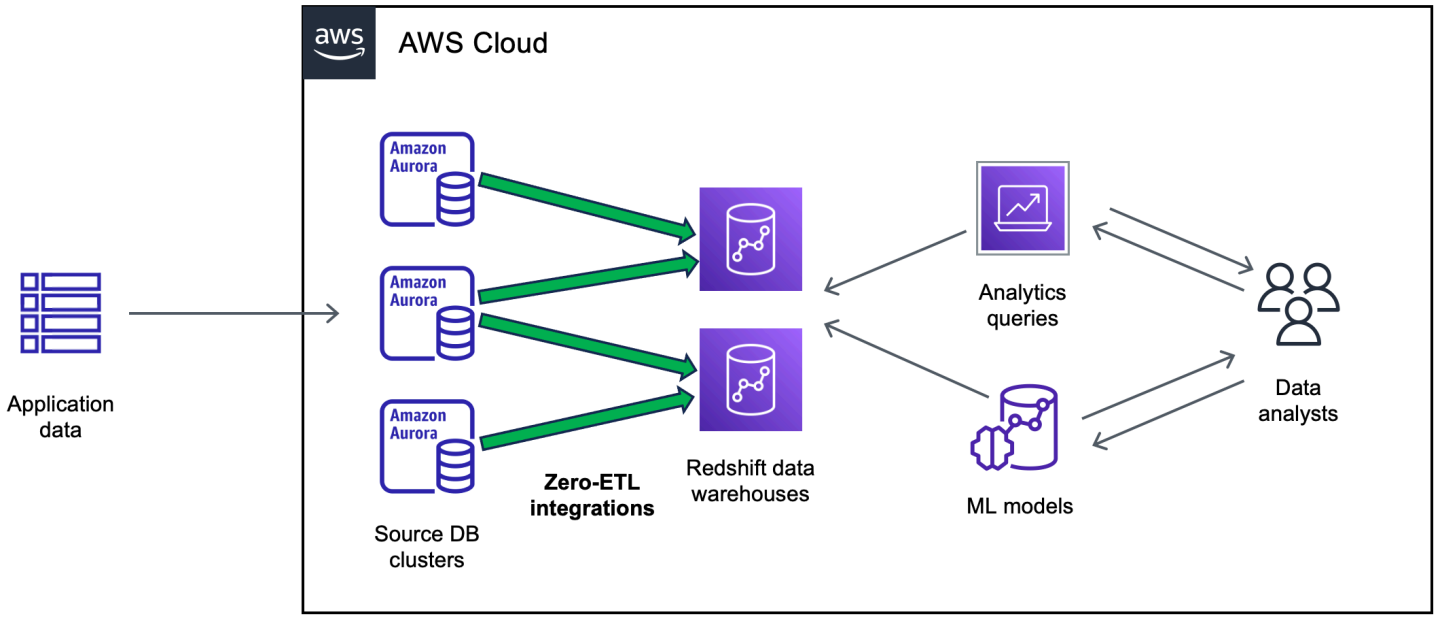
Intégrations zéro ETL Aurora

Il s’agit d’une solution entièrement gérée qui permet de rendre les données transactionnelles disponibles dans votre destination d’analytique après leur écriture dans un cluster de bases de données Aurora. Le processus d’extraction, transformation et chargement (ETL) consiste à combiner des données provenant de plusieurs sources dans un grand entrepôt de données central.

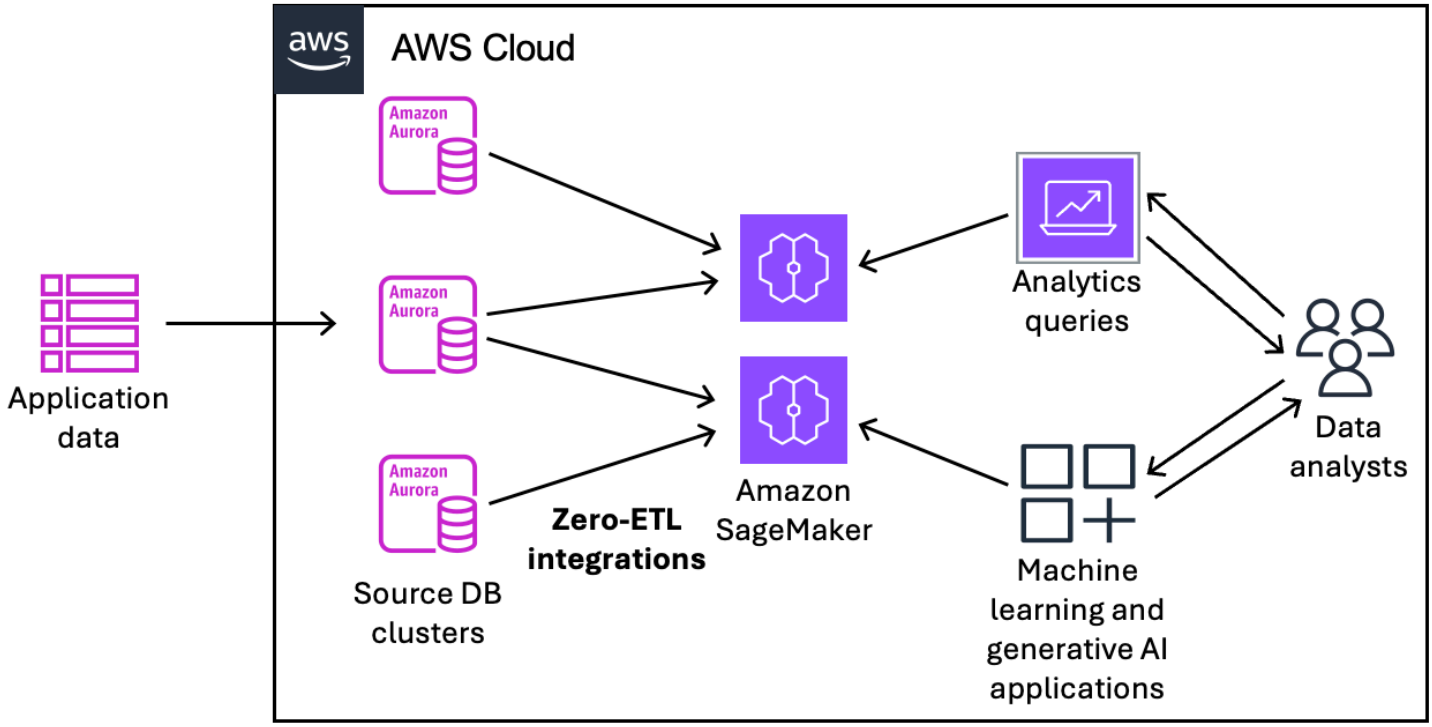
Une intégration zéro ETL rend les données de votre cluster de bases de données Aurora disponibles dans Amazon Redshift ou un Amazon SageMaker AI Lakehouse en temps quasi réel. Une fois que ces données se trouvent dans l'entrepôt de données ou le lac de données cible, vous pouvez optimiser vos charges de travail d'analyse, de machine learning et d'IA à l'aide des fonctionnalités intégrées, telles que l'apprentissage automatique, les vues matérialisées, le partage de données, l'accès fédéré à plusieurs magasins de données et lacs de données, et les intégrations avec SageMaker Amazon AI, Quick Suite, etc. Services AWS

Pour créer une intégration zéro ETL, vous devez spécifier un cluster de bases de données Aurora comme source et un entrepôt de données ou un lakehouse pris en charge comme cible. L’intégration réplique les données de la base de données source vers l’entrepôt de données ou le lakehouse cible.

Le schéma suivant illustre cette fonctionnalité d’intégration zéro ETL à Amazon Redshift :



Le schéma suivant illustre cette fonctionnalité pour l’intégration zéro ETL avec un Amazon SageMaker AI Lakehouse :



L'intégration surveille l'état du pipeline de données et effectue la récupération en cas de problèmes, lorsque cela est possible. Vous pouvez créer des intégrations à partir de plusieurs clusters de bases de données Aurora dans un seul entrepôt de données ou lakehouse cible, ce qui vous permet de dériver des informations entre plusieurs applications.

Pour plus d'informations sur la tarification des intégrations zéro ETL, consultez Tarification d'Amazon Aurora et Tarification d'Amazon Redshift.

Rubriques

- Ajouter des données à un cluster de bases de données Aurora source et les interroger
- Affichage et surveillance des intégrations zéro ETL Aurora
- Modification des intégrations zéro ETL Aurora
- Suppression d’intégrations zéro ETL Aurora
- Résolution des problèmes liés aux intégrations zéro ETL Aurora

Avantages

Les intégrations zéro ETL Aurora présentent les avantages suivants :

- Elles vous aident à dériver des informations holistiques de plusieurs sources de données.
- Elles éliminent la nécessité de créer et de gérer des pipelines de données complexes qui effectuent des opérations d'extraction, de transformation et de chargement (ETL). Les intégrations zéro ETL suppriment les défis liés à la création et à la gestion de pipelines en les provisionnant et en les gérant pour vous.
- Elles réduisent la charge opérationnelle et les coûts, et vous permettent de vous concentrer sur l'amélioration de vos applications.
- Elles vous permettent de tirer parti des fonctionnalités d’analytique et de machine learning de la destination cible pour dériver des informations à partir de données transactionnelles et autres, afin de répondre efficacement aux événements critiques et urgents.

Concepts clés

Lorsque vous commencez à utiliser des intégrations zéro ETL, tenez compte des concepts suivants :

Integration

: Pipeline de données entièrement géré qui réplique automatiquement les données transactionnelles et les schémas d’un cluster de bases de données Aurora vers un entrepôt de données ou un catalogue.

Base de données source

: Cluster de bases de données Aurora à partir d’où les données sont répliquées. Vous pouvez spécifier un cluster de base de données qui utilise des instances de base de données provisionnées ou des Aurora Serverless v2 instances de base de données comme source.

Cible

L’entrepôt de données ou le lakehouse vers lequel les données sont répliquées. Il existe deux types d'entrepôts de données : l'entrepôt de données en cluster provisionné et l'entrepôt de données sans serveur. Un entrepôt de données en cluster provisionné est une collection de ressources informatiques appelées nœuds, qui sont organisées en un groupe appelé cluster. Un entrepôt de données sans serveur est composé d'un groupe de travail qui stocke les ressources de calcul et d'un espace de noms qui héberge les utilisateurs et les objets de base de données. Les deux entrepôts de données exécutent un moteur d’analytique et contiennent une ou plusieurs bases de données.

Un lakehouse cible se compose de catalogues, de bases de données, de tables et de vues. Pour plus d’informations sur l’architecture de lakehouse, consultez la section SageMaker Lakehouse components dans le Guide de l’utilisateur Amazon SageMaker AI Unified Studio.

Plusieurs clusters de bases de données sources peuvent écrire sur la même cible.

Pour plus d'informations, consultez Architecture système de l'entrepôt de données dans le Guide du développeur de base de données Amazon Redshift.

Limitations

Les limitations suivantes s’appliquent aux intégrations zéro ETL Aurora.

Rubriques

- Limitations générales
- Limitations propres à Aurora MySQL
- Limitations relatives à Aurora PostgreSQL
- Limitations propres à Amazon Redshift
- Amazon SageMaker AI limites du lakehouse

Limitations générales

- Le cluster de bases de données source doit se trouver dans la même région que la cible.
- Vous ne pouvez pas renommer un cluster de bases de données ni aucune de ses instances possédant des intégrations existantes.

- Vous ne pouvez pas créer plusieurs intégrations entre les mêmes bases de données source et cible.
- Vous ne pouvez pas supprimer un cluster de bases de données qui possède des intégrations existantes. Vous devez d'abord supprimer toutes les intégrations associées.
- Si vous arrêtez le cluster de bases de données source, les dernières transactions risquent de ne pas être répliquées vers l'entrepôt de données cible tant que vous ne reprenez pas l'exécution du cluster.
- Si votre cluster est la source d'un déploiement bleu/vert, les environnements bleu et vert ne peuvent pas comporter d'intégrations zéro ETL existantes lors de la bascule. Vous devez d'abord supprimer l'intégration et basculer, puis la recréer.
- Un cluster de bases de données doit contenir au moins une instance de base de données pour être la source d'une intégration.
- Vous ne pouvez pas créer d'intégration pour un cluster de base de données source qui est un clone entre comptes, tel que ceux partagés à l'aide de AWS Resource Access Manager (AWS RAM).
- Si votre cluster source est le cluster de bases de données principal d'une base de données globale Aurora et qu'il bascule sur l'un de ses clusters secondaires, l'intégration devient inactive. Vous devez supprimer et recréer l'intégration.
- Vous ne pouvez pas créer d'intégration pour une base de données source dont une autre intégration est activement créée.
- Lors de la création initiale d'une intégration ou lors de la resynchronisation d'une table, l'ensemencement des données de la source vers la cible peut prendre 20 à 25 minutes, voire plus, selon la taille de la base de données source. Ce délai peut entraîner une augmentation du retard de réplica.
- Certains types de données ne sont pas pris en charge. Pour de plus amples informations, veuillez consulter the section called “Différences de type de données”.
- Les tables système, les tables temporaires et les vues ne sont pas répliquées vers les entrepôts cibles.
- L'exécution de commandes DDL (par exemple`ALTER TABLE`) sur une table source peut déclencher une resynchronisation de la table, rendant la table indisponible pour les requêtes pendant la resynchronisation. Pour de plus amples informations, veuillez consulter the section called “Une ou plusieurs de mes tables Amazon Redshift nécessitent une resynchronisation”.

Limitations propres à Aurora MySQL

- Votre cluster de bases de données source doit exécuter une version prise en charge d'Aurora MySQL. Pour une liste de versions prises en charge, consultez the section called "Intégrations sans ETL ".
- Les intégrations zéro ETL s'appuient sur la journalisation binaire MySQL (binlog) pour capturer les modifications continues des données. N'utilisez par le filtrage de données basé sur binlog, car cela peut entraîner des incohérences de données entre les bases de données source et cible.
- Les intégrations zéro ETL sont prises en charge uniquement pour les bases de données configurées pour utiliser le moteur de stockage InnoDB.
- Les références de clé étrangère avec des mises à jour de table prédéfinies ne sont pas prises en charge. Plus précisément, les règles `ON DELETE` et `ON UPDATE` ne sont pas prises en charge avec les actions `CASCADE`, `SET NULL` et `SET DEFAULT`. Toute tentative de création ou de mise à jour d'une table contenant de telles références à une autre table entraînera l'échec de la table.
- Les transactions XA effectuées sur le cluster de bases de données source font passer l'intégration à l'état `Syncing`.

Limitations relatives à Aurora PostgreSQL

- Votre cluster de bases de données source doit exécuter une version prise en charge d'Aurora PostgreSQL. Pour une liste de versions prises en charge, consultez the section called "Intégrations sans ETL ".
- Si vous sélectionnez un cluster de bases de données Aurora PostgreSQL source, vous devez spécifier au moins un modèle de filtre de données. Le modèle doit au minimum inclure une seule base de données (*`database-name`*`.*.*`) pour la réplication vers l'entrepôt cible. Pour de plus amples informations, veuillez consulter the section called "Filtrage des données pour les intégrations zéro ETL".
- Toutes les bases de données créées dans le cluster de bases de données Aurora PostgreSQL source doivent utiliser l'encodage UTF-8.
- Lorsque le partitionnement déclaratif est utilisé, les partitions de table sont répliquées sur Amazon Redshift. Cependant, la table partitionnée elle-même n'est pas répliquée sur Amazon Redshift.
- Les transactions en deux phases ne sont pas prises en charge.

- Si vous supprimez toutes les instances de base de données d'un cluster de bases de données à l'origine d'une intégration, puis que vous ajoutez à nouveau une instance de base de données, la réplication est interrompue entre les clusters source et cible.
- Le cluster de bases de données source ne peut pas utiliser la base de données Aurora Limitless.
- Les clés primaires sont requises sur toutes les tables présentes dans le filtre de données. Toutes les tables dépourvues de clé primaire seront mises en état d'échec.

Limitations propres à Amazon Redshift

Pour une liste des limitations Amazon Redshift liées aux intégrations zéro ETL, consultez la section Considérations relatives au moment d'utiliser les intégrations zéro ETL avec Amazon Redshift dans le Guide de gestion Amazon Redshift.

Amazon SageMaker AI limites du lakehouse

Voici une limite pour les intégrations Amazon SageMaker AI Lakehouse Zero-ETL.

- Les noms de catalogue sont limités à 19 caractères.

Quotas

Votre compte possède les quotas suivants relatifs aux intégrations zéro ETL Aurora. Chaque quota s'applique par région, sauf indication contraire.

Nom	Par défaut	Description
Intégrations	100	Nombre total d'intégrations au sein d'un Compte AWS.
Intégrations par cible	50	Nombre d'intégrations envoyant des données à un entrepôt de données ou un lakehouse cible unique.
Intégrations par cluster source	5	Nombre d'intégrations envoyant des données à partir d'un cluster de bases de données source unique.

En outre, l'entrepôt cible impose certaines limites au nombre de tables autorisées dans chaque instance de base de données ou nœud de cluster. Pour plus d'informations sur les quotas et limites

relatifs à Amazon Redshift, consultez la section Quotas et limites dans Amazon Redshift dans le Guide de gestion Amazon Redshift.

Régions prises en charge

Les intégrations Aurora Zero-ETL sont disponibles dans un sous-ensemble de. Régions AWS Pour obtenir une liste des régions prises en charge, consultez the section called “Intégrations sans ETL ”.

Bien démarrer avec les intégrations zéro ETL Aurora

Avant de créer une intégration zéro ETL, configurez votre cluster de bases de données Aurora et votre entrepôt de données avec les paramètres et les autorisations nécessaires. Au cours de la configuration, vous allez suivre les étapes suivantes :

1. Création d’un groupe personnalisé de paramètres de cluster de bases de données.
2. Créez un cluster de bases de données source.
3. Créez un entrepôt de données cible pour Amazon Redshift ou créez un lakehouse Amazon SageMaker AI cible.

Une fois ces tâches terminées, reportez-vous à the section called “Création d’intégrations zéro ETL avec Amazon Redshift” ou the section called “Création d’Intégrations zéro ETL avec un Amazon SageMaker Lakehouse”.

Vous pouvez utiliser le AWS SDKs pour automatiser le processus de configuration pour vous. Pour de plus amples informations, veuillez consulter the section called “Configurez une intégration à l'aide du AWS SDKs”.

> **Tip**
>
> Vous pouvez demander à RDS d’effectuer ces étapes de configuration pour vous pendant que vous créez l’intégration, plutôt que de les exécuter manuellement. Pour commencer immédiatement la création d'une intégration, consultez the section called “Création d’intégrations zéro ETL avec Amazon Redshift”.

Pour l’étape 3, vous pouvez choisir de créer un entrepôt de données cible (étape 3a) ou un lakehouse cible (étape 3b) en fonction de vos besoins :

- Choisissez un entrepôt de données si vous avez besoin de fonctionnalités d'entreposage de données traditionnelles avec des opérations d'analytique basées sur SQL.
- Choisissez un Amazon SageMaker AI Lakehouse si vous avez besoin de fonctionnalités d'apprentissage automatique et souhaitez utiliser les fonctionnalités de Lakehouse pour la science des données et les flux de travail de machine learning.

Étape 1 : Création d'un groupe de paramètres de cluster de bases de données personnalisé

Les intégrations zéro ETL Aurora nécessitent des valeurs spécifiques pour les paramètres de cluster de bases de données qui contrôlent la réplication. Plus précisément, Aurora MySQL nécessite un binlog (*aurora_enhanced_binlog*) amélioré, et Aurora PostgreSQL nécessiteune réplication logique améliorée (`aurora.enhanced_logical_replication`).

Pour configurer la journalisation binaire ou la réplication logique, vous devez créer un groupe de paramètres de cluster de bases de données personnalisés, puis l'associer au cluster de bases de données source.

Aurora MySQL (famille aurora-mysql8.0) :

- `aurora_enhanced_binlog=1`
- `binlog_backup=0`
- `binlog_format=ROW`
- `binlog_replication_globaldb=0`
- `binlog_row_image=full`
- `binlog_row_metadata=full`

Assurez-vous également que le paramètre `binlog_transaction_compression` n'est pas défini sur `ON` et que le paramètre `binlog_row_value_options` n'est pas défini sur `PARTIAL_JSON`.

Pour plus d'informations sur le binlog amélioré Aurora MySQL, consultez the section called "Configuration du binlog amélioré".

Aurora PostgreSQL (famille aurora-postgresql16) :

- `rds.logical_replication=1`

- `aurora.enhanced_logical_replication=1`
- `aurora.logical_replication_backup=0`
- `aurora.logical_replication_globaldb=0`

L'activation de la réplication logique améliorée (`aurora.enhanced_logical_replication`) écrira toujours toutes les valeurs des colonnes dans le journal d'écriture anticipée (WAL), même si `REPLICA IDENTITY FULL` n'est pas activée. Cela peut augmenter les IOPS pour votre cluster de bases de données source.

> **Important**
>
> Si vous activez ou désactivez le paramètre de cluster de bases de données `aurora.enhanced_logical_replication`, l'instance de base de données principale invalide tous les emplacements de réplication logique. Cela arrête la réplication de la source vers la cible, et vous devez recréer des emplacements de réplication sur l'instance de base de données principale. Pour éviter les interruptions, veillez à ce que l'état des paramètres reste cohérent pendant la réplication.

Étape 2 : Sélection ou création d'un cluster de bases de données source

Après avoir créé un groupe de paramètres de cluster de bases de données personnalisés, choisissez ou créez un cluster de bases de données Aurora. Ce cluster est la source de réplication des données vers l'entrepôt de données cible. Vous pouvez spécifier un cluster de bases de données qui utilise des instances de base de données provisionnées ou des instances de base de données Aurora Serverless v2 comme source. Pour des instructions sur la création d'un cluster de bases de données, consultez the section called "Création d'un cluster de bases de données" ou the section called "Création d'un cluster de bases de données Aurora Serverless v2".

La base de données doit exécuter une version de moteur de base de données prise en charge. Pour une liste de versions prises en charge, consultez the section called "Intégrations sans ETL ".

Lorsque vous cliquez sur la base de données, sous Configuration supplémentaire, remplacez le groupe de paramètres de cluster de bases de données par défaut par le groupe de paramètres personnalisés que vous avez créé à l'étape précédente.

> **Note**
>
> Si vous associez le groupe de paramètres au cluster de bases de données après la création de celui-ci, vous devez redémarrer l'instance de base de données principale dans le cluster pour appliquer les modifications avant de créer une intégration zéro ETL. Pour obtenir des instructions, consultez the section called "Redémarrage d'un cluster de bases de données ou d'une instance de bases de données Aurora".

Étape 3a : Création d'un entrepôt de données cible

Après avoir créé votre cluster de bases de données source, vous devez créer et configurer un entrepôt de données cible. L'entrepôt de données doit respecter les exigences suivantes :

- En utilisant un type de RA3 nœud avec au moins deux nœuds, ou Redshift Serverless.
- Chiffré (si vous utilisez un cluster provisionné). Pour plus d'informations, consultez Chiffrement de base de données Amazon Redshift.

Pour obtenir des instructions sur la création d'un entrepôt de données, consultez Création d'un cluster pour les clusters provisionnés ou Création d'un groupe de travail avec un espace de noms pour Redshift sans serveur.

Activer la sensibilité à la casse sur l'entrepôt de données

Pour que l'intégration réussisse, le paramètre de sensibilité à la casse (`enable_case_sensitive_identifier`) doit être activé pour l'entrepôt de données. Par défaut, la sensibilité à la casse est désactivée sur tous les clusters provisionnés et les groupes de travail Redshift sans serveur.

Pour activer la sensibilité à la casse, effectuez les étapes suivantes en fonction du type de votre entrepôt de données :

- Cluster provisionné : pour activer la sensibilité à la casse sur un cluster provisionné, créez un groupe de paramètres personnalisé en activant le paramètre `enable_case_sensitive_identifier`. Associez ensuite le groupe de paramètres au cluster. Pour obtenir des instructions, consultez Gestion des groupes de paramètres à l'aide de la console ou Configuration des valeurs des paramètres à l'aide de l'AWS CLI.

> **Note**
>
> N'oubliez pas de redémarrer le cluster après lui avoir associé le groupe de paramètres personnalisé.

- Groupe de travail sans serveur : pour activer la sensibilité à la casse sur un groupe de travail Redshift sans serveur, vous devez utiliser l'AWS CLI. La console Amazon Redshift ne prend actuellement pas en charge la modification des valeurs des paramètres Redshift sans serveur. Envoyez la demande update-workgroup suivante :

```
aws redshift-serverless update-workgroup \
  --workgroup-name target-workgroup \
  --config-parameters
 parameterKey=enable_case_sensitive_identifier,parameterValue=true
```

  Vous n'avez pas besoin de redémarrer un groupe de travail après avoir modifié ses valeurs de paramètres.

Configuration de l’autorisation pour l’entrepôt de données

Après avoir créé un entrepôt de données, vous devez configurer le cluster de bases de données Aurora source en tant que source d’intégration autorisée. Pour obtenir des instructions, consultez Configuration de l’autorisation pour votre entrepôt de données Amazon Redshift.

Configurez une intégration à l'aide du AWS SDKs

Plutôt que de configurer chaque ressource manuellement, vous pouvez exécuter le script Python suivant pour configurer automatiquement les ressources requises pour vous. L’exemple de code utilise le AWS SDK pour Python (Boto3) pour créer un cluster de bases de données Amazon Aurora source et un entrepôt de données cible, chacun avec les valeurs de paramètres requises. Il attend ensuite que les bases de données soient disponibles avant de créer une intégration zéro ETL entre elles. Vous pouvez commenter différentes fonctions en fonction des ressources que vous devez configurer.

Pour installer les dépendances requises, exécutez les commandes suivantes :

```
pip install boto3
```

```
pip install time
```

Dans le script, modifiez éventuellement les noms de la source, de la cible et des groupes de paramètres. La fonction finale crée une intégration nommée `my-integration` d'après la configuration des ressources.

Exemple de code python

Aurora MySQL

```
import boto3
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')

source_cluster_name = 'my-source-cluster' # A name for the source cluster
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_cluster(*args):
    """Creates a source Aurora MySQL DB cluster"""

    response = rds.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        DBParameterGroupFamily='aurora-mysql8.0',
        Description='For Aurora MySQL binary logging'
    )
    print('Created source parameter group: ' + response['DBClusterParameterGroup']
['DBClusterParameterGroupName'])

    response = rds.modify_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        Parameters=[
            {
```

```
                'ParameterName': 'aurora_enhanced_binlog',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_backup',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_format',
                'ParameterValue': 'ROW',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_replication_globaldb',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_image',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'binlog_row_metadata',
                'ParameterValue': 'full',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
    print('Modified source parameter group: ' +
response['DBClusterParameterGroupName'])

    response = rds.create_db_cluster(
        DBClusterIdentifier=source_cluster_name,
        DBClusterParameterGroupName=source_param_group_name,
        Engine='aurora-mysql',
        EngineVersion='8.0.mysql_aurora.3.05.2',
        DatabaseName='myauroradb',
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
```

```
    print('Creating source cluster: ' + response['DBCluster']
['DBClusterIdentifier'])
    source_arn = (response['DBCluster']['DBClusterArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)

    response = rds.create_db_instance(
        DBInstanceClass='db.r6g.2xlarge',
        DBClusterIdentifier=source_cluster_name,
        DBInstanceIdentifier=source_cluster_name + '-instance',
        Engine='aurora-mysql'
    )
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""

    response = redshift.create_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='For Aurora MySQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])

    response = redshift.create_cluster(
        ClusterIdentifier=target_cluster_name,
        NodeType='ra3.4xlarge',
        NumberOfNodes=2,
        Encrypted=True,
        MasterUsername='username',
        MasterUserPassword='Password01**',
        ClusterParameterGroupName=target_param_group_name
    )
```

```
    print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

    # Retrieve the target cluster ARN
    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Retrieve the current user's account ID
    response = sts.get_caller_identity()
    account_id = response['Account']

    # Create a resource policy specifying cluster ARN and account ID
    response = redshift.put_resource_policy(
        ResourceArn=target_arn,
        Policy='''
        {
            \"Version\":\"2012-10-17\",
            \"Statement\":[
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"Service\":\"redshift.amazonaws.com\"
                },
                \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
                \"Condition\":{
                    \"StringEquals\":{
                        \"aws:SourceArn\":\"%s\"}
                    }
                },
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"AWS\":\"arn:aws:iam::%s:root\"},
                \"Action\":\"redshift:CreateInboundIntegration\"}
            ]
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_cluster_availability(*args):
    """Waits for both clusters to be available"""

    print('Waiting for clusters to be available...')
```

```
    response = rds.describe_db_clusters(
        DBClusterIdentifier=source_cluster_name
    )
    source_status = response['DBClusters'][0]['Status']
    source_arn = response['DBClusters'][0]['DBClusterArn']

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_cluster_name + '-instance'
    )
    source_instance_status = response['DBInstances'][0]['DBInstanceStatus']

    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Every 60 seconds, check whether the clusters are available.
    if source_status != 'available' or target_status != 'available' or
 source_instance_status != 'available':
        time.sleep(60)
        response = wait_for_cluster_availability(
            source_cluster_name, target_cluster_name)
    else:
        print('Clusters available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target clusters"""

    response = rds.create_integration(
        SourceArn=source_arn,
        TargetArn=target_arn,
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_cluster(source_cluster_name, source_param_group_name)
    wait_for_cluster_availability(source_cluster_name, target_cluster_name)

if __name__ == "__main__":
```

```
    main()
```

Aurora PostgreSQL

```
import boto3
import time

# Build the client using the default credential configuration.
# You can use the CLI and run 'aws configure' to set access key, secret
# key, and default Region.

rds = boto3.client('rds')
redshift = boto3.client('redshift')
sts = boto3.client('sts')

source_cluster_name = 'my-source-cluster' # A name for the source cluster
source_param_group_name = 'my-source-param-group' # A name for the source parameter
 group
target_cluster_name = 'my-target-cluster' # A name for the target cluster
target_param_group_name = 'my-target-param-group' # A name for the target parameter
 group

def create_source_cluster(*args):
    """Creates a source Aurora PostgreSQL DB cluster"""

    response = rds.create_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        DBParameterGroupFamily='aurora-postgresql16',
        Description='For Aurora PostgreSQL logical replication'
    )
    print('Created source parameter group: ' + response['DBClusterParameterGroup']
['DBClusterParameterGroupName'])

    response = rds.modify_db_cluster_parameter_group(
        DBClusterParameterGroupName=source_param_group_name,
        Parameters=[
            {
                'ParameterName': 'rds.logical_replication',
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.enhanced_logical_replication',
```

```
                'ParameterValue': '1',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.logical_replication_backup',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            },
            {
                'ParameterName': 'aurora.logical_replication_globaldb',
                'ParameterValue': '0',
                'ApplyMethod': 'pending-reboot'
            }
        ]
    )
    print('Modified source parameter group: ' +
 response['DBClusterParameterGroupName'])

    response = rds.create_db_cluster(
        DBClusterIdentifier=source_cluster_name,
        DBClusterParameterGroupName=source_param_group_name,
        Engine='aurora-postgresql',
        EngineVersion='16.4.aurora-postgresql',
        DatabaseName='mypostgresdb',
        MasterUsername='username',
        MasterUserPassword='Password01**'
    )
    print('Creating source cluster: ' + response['DBCluster']
['DBClusterIdentifier'])
    source_arn = (response['DBCluster']['DBClusterArn'])
    create_target_cluster(target_cluster_name, source_arn, target_param_group_name)

    response = rds.create_db_instance(
        DBInstanceClass='db.r6g.2xlarge',
        DBClusterIdentifier=source_cluster_name,
        DBInstanceIdentifier=source_cluster_name + '-instance',
        Engine='aurora-postgresql'
    )
    return(response)

def create_target_cluster(target_cluster_name, source_arn, target_param_group_name):
    """Creates a target Redshift cluster"""

    response = redshift.create_cluster_parameter_group(
```

```
        ParameterGroupName=target_param_group_name,
        ParameterGroupFamily='redshift-1.0',
        Description='For Aurora PostgreSQL zero-ETL integrations'
    )
    print('Created target parameter group: ' + response['ClusterParameterGroup']
['ParameterGroupName'])

    response = redshift.modify_cluster_parameter_group(
        ParameterGroupName=target_param_group_name,
        Parameters=[
            {
                'ParameterName': 'enable_case_sensitive_identifier',
                'ParameterValue': 'true'
            }
        ]
    )
    print('Modified target parameter group: ' + response['ParameterGroupName'])

    response = redshift.create_cluster(
        ClusterIdentifier=target_cluster_name,
        NodeType='ra3.4xlarge',
        NumberOfNodes=2,
        Encrypted=True,
        MasterUsername='username',
        MasterUserPassword='Password01**',
        ClusterParameterGroupName=target_param_group_name
    )
    print('Creating target cluster: ' + response['Cluster']['ClusterIdentifier'])

    # Retrieve the target cluster ARN
    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']

    # Retrieve the current user's account ID
    response = sts.get_caller_identity()
    account_id = response['Account']

    # Create a resource policy specifying cluster ARN and account ID
    response = redshift.put_resource_policy(
        ResourceArn=target_arn,
        Policy='''
        {
```

```
            \"Version\":\"2012-10-17\",
            \"Statement\":[
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"Service\":\"redshift.amazonaws.com\"
                },
                \"Action\":[\"redshift:AuthorizeInboundIntegration\"],
                \"Condition\":{
                    \"StringEquals\":{
                        \"aws:SourceArn\":\"%s\"}
                    }
                },
                {\"Effect\":\"Allow\",
                \"Principal\":{
                    \"AWS\":\"arn:aws:iam::%s:root\"},
                \"Action\":\"redshift:CreateInboundIntegration\"}
            ]
        }
        ''' % (source_arn, account_id)
    )
    return(response)

def wait_for_cluster_availability(*args):
    """Waits for both clusters to be available"""

    print('Waiting for clusters to be available...')

    response = rds.describe_db_clusters(
        DBClusterIdentifier=source_cluster_name
    )
    source_status = response['DBClusters'][0]['Status']
    source_arn = response['DBClusters'][0]['DBClusterArn']

    response = rds.describe_db_instances(
        DBInstanceIdentifier=source_cluster_name + '-instance'
    )
    source_instance_status = response['DBInstances'][0]['DBInstanceStatus']

    response = redshift.describe_clusters(
        ClusterIdentifier=target_cluster_name
    )
    target_status = response['Clusters'][0]['ClusterStatus']
    target_arn = response['Clusters'][0]['ClusterNamespaceArn']
```

```
    # Every 60 seconds, check whether the clusters are available.
    if source_status != 'available' or target_status != 'available' or
 source_instance_status != 'available':
        time.sleep(60)
        response = wait_for_cluster_availability(
            source_cluster_name, target_cluster_name)
    else:
        print('Clusters available. Ready to create zero-ETL integration.')
        create_integration(source_arn, target_arn)
        return

def create_integration(source_arn, target_arn):
    """Creates a zero-ETL integration using the source and target clusters"""

    response = rds.create_integration(
        SourceArn=source_arn,
        TargetArn=target_arn,
        IntegrationName='my-integration'
    )
    print('Creating integration: ' + response['IntegrationName'])

def main():
    """main function"""
    create_source_cluster(source_cluster_name, source_param_group_name)
    wait_for_cluster_availability(source_cluster_name, target_cluster_name)

if __name__ == "__main__":
    main()
```

Étape 3b : Création d'un AWS Glue catalogue pour l'intégration Amazon SageMaker AI Zero-ETL

Lorsque vous créez une intégration zéro ETL avec un Amazon SageMaker AI Lakehouse, vous devez créer un catalogue AWS Glue géré dans. AWS Lake Formation Le catalogue cible doit être un catalogue géré Amazon Redshift. Pour créer un catalogue géré Amazon Redshift, créez d'abord le rôle lié à un service `AWSServiceRoleForRedshift`. Dans la console Lake Formation, ajoutez le `AWSServiceRoleForRedshift` en tant qu'administrateur en lecture seule.

Pour plus d'informations sur les tâches précédentes, consultez les rubriques suivantes :

- Pour plus d'informations sur la création d'un catalogue géré Amazon Redshift, consultez la section Création d'un catalogue géré Amazon Redshift dans le AWS Glue Data Catalog du Guide du développeurAWS Lake Formation.
- Pour plus d'informations sur le rôle lié à un service pour Amazon Redshift, consultez la section Utilisation des rôles liés à un service pour Amazon Redshift du Guide de gestion Amazon Redshift.
- Pour plus d'informations sur les autorisations d'administrateur en lecture seule pour Lake Formation, consultez la section Références relatives aux personas de Lake Formation et aux autorisations IAM dans le Guide du développeur AWS Lake Formation.

Configurer les autorisations pour le AWS Glue catalogue cible

Avant de créer un catalogue cible pour une intégration Zero-ETL, vous devez créer le rôle de création de cible Lake Formation et le rôle de transfert de AWS Glue données. Utilisez le rôle de création de cible Lake Formation pour créer le catalogue cible. Lors de la création du catalogue cible, entrez le rôle de transfert de données Glue dans le champ Rôle IAM de la section Accès depuis les moteurs.

Rôle de création de cible Lake Formation

Le rôle de création de cible doit être un administrateur de Lake Formation et nécessite les autorisations suivantes.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "lakeformation:RegisterResource",
            "Resource": "*"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": [
                "s3:PutEncryptionConfiguration",
                "iam:PassRole",
                "glue:CreateCatalog",
                "glue:GetCatalog",
```

```
                "s3:PutBucketTagging",
                "s3:PutLifecycleConfiguration",
                "s3:PutBucketPolicy",
                "s3:CreateBucket",
                "redshift-serverless:CreateNamespace",
                "s3:DeleteBucket",
                "s3:PutBucketVersioning",
                "redshift-serverless:CreateWorkgroup"
            ],
            "Resource": [
                "arn:aws:glue:*:111122223333:catalog",
                "arn:aws:glue:*:111122223333:catalog/*",
                "arn:aws:s3:::*",
                "arn:aws:redshift-serverless:*:111122223333:workgroup/*",
                "arn:aws:redshift-serverless:*:111122223333:namespace/*",
                "arn:aws:iam::111122223333:role/GlueDataCatalogDataTransferRole"
            ]
        }
    ]
}
```

Le rôle de création de cible doit avoir la relation d'approbation suivante :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
          "Effect": "Allow",
          "Principal": {
            "AWS": "arn:aws:iam::111122223333:user/Username"
          },
          "Action": "sts:AssumeRole"
        }
```

```
    ]
}
```

Rôle de transfert de données Glue

Le rôle de transfert de données Glue est requis pour les opérations du catalogue MySQL et doit disposer des autorisations suivantes.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DataTransferRolePolicy",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "glue:GetCatalog",
                "glue:GetDatabase"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Le rôle de transfert de données Glue doit avoir la relation d’approbation suivante :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": [
                    "glue.amazonaws.com",
                    "redshift.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Étapes suivantes

Avec un cluster de Aurora DB source et un entrepôt de données cible Amazon Redshift ou SageMaker Amazon AI Lakehouse, vous pouvez créer une intégration zéro ETL et répliquer les données. Pour obtenir des instructions, veuillez consulter the section called “Création d’intégrations zéro ETL avec Amazon Redshift”.

Création d'intégrations zéro ETL d'Aurora à Amazon Redshift

Lorsque vous créez une intégration zéro ETL Aurora, vous spécifiez le cluster de bases de données Aurora source et l’entrepôt de données Amazon Redshift cible. Vous pouvez également personnaliser les paramètres de chiffrement et ajouter des balises. Aurora crée une intégration entre le cluster de bases de données source et sa cible. Une fois l’intégration active, toutes les données que vous insérez dans le cluster de bases de données source sont répliquées dans la cible Amazon Redshift configurée.

Conditions préalables

Avant de créer une intégration zéro ETL, vous devez spécifier un cluster de bases de données source et un entrepôt de données Amazon Redshift cible. Vous devez également autoriser la réplication dans l’entrepôt de données en ajoutant le cluster de bases de données en tant que source d’intégration autorisée.

Pour obtenir des instructions sur la réalisation de chacune de ces étapes, consultez the section called “Bien démarrer avec les intégrations zéro ETL”.

Autorisations requises

Certaines autorisations IAM sont requises pour créer une intégration zéro ETL. Vous avez au moins besoin des autorisations requises pour effectuer les actions suivantes :

- Créer des intégrations zéro ETL le cluster de bases de données Aurora source.
- Afficher et supprimer toutes les intégrations zéro ETL.
- Créer des intégrations entrantes dans l'entrepôt de données cible.

Les exemples de politiques suivants montrent les autorisations de moindre privilège requises pour créer et gérer des intégrations. Il se peut que vous n'ayez pas besoin de ces autorisations exactes si votre utilisateur ou votre rôle dispose d'autorisations plus étendues, telles qu'une politique gérée `AdministratorAccess`.

> **Note**
>
> Les noms de ressources Redshift Amazon (ARNs) ont le format suivant. Notez l'utilisation d'une barre oblique (`/`) à la place du caractère deux-points (`:`) avant l'UUID de l'espace de noms sans serveur.
>
> - Cluster provisionné : `arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid`
> - Sans serveur : `arn:aws:redshift-serverless:{region}:{account-id}:namespace/namespace-uuid`

Exemple de politique pour la cible Redshift

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CreateIntegration",
      "Effect": "Allow",
      "Action": [
```

```
        "rds:CreateIntegration"
      ],
      "Resource": [
        "arn:aws:rds:us-east-1:123456789012:db:source-db",
        "arn:aws:rds:us-east-1:123456789012:integration:*"
      ]
    },
    {
      "Sid": "DescribeIntegrationDetails",
      "Effect": "Allow",
      "Action": [
        "rds:DescribeIntegrations"
      ],
      "Resource": [
      "arn:aws:rds:us-east-1:123456789012:integration:*"
  ]
    },
    {
      "Sid": "ChangeIntegrationDetails",
      "Effect": "Allow",
      "Action": [
        "rds:DeleteIntegration",
        "rds:ModifyIntegration"
      ],
      "Resource": [
        "arn:aws:rds:us-east-1:123456789012:integration:*"
      ]
    },
    {
      "Sid": "AllowRedShiftIntegration",
      "Effect": "Allow",
      "Action": [
        "redshift:CreateInboundIntegration"
      ],
      "Resource": [
        "arn:aws:redshift:us-east-1:123456789012:namespace:namespace-uuid"
      ]
    }
  ]
}
```

Choix d'un entrepôt de données cible dans un autre compte

Si vous prévoyez de spécifier un entrepôt de données Amazon Redshift cible situé dans un autreCompte AWS, vous devez créer un rôle permettant aux utilisateurs du compte courant d'accéder aux ressources du compte cible. Pour plus d'informations, consultez la section Fournir un accès à un utilisateur IAM dans un autre utilisateur Compte AWS dont vous êtes le propriétaire.

Le rôle doit disposer des autorisations suivantes, qui permettent à l'utilisateur de consulter les clusters provisionnés Amazon Redshift et les espaces de noms Redshift sans serveur disponibles dans le compte cible.

Autorisations requises et politique d'approbation

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "redshift:DescribeClusters",
            "redshift-serverless:ListNamespaces"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Le rôle doit respecter la politique d’approbation suivante, qui spécifie l’ID du compte cible.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Pour obtenir des instructions quant à la création du rôle, consultez Création d'un rôle à l'aide de politiques d'approbation personnalisées.

Création d'intégrations zéro ETL

Vous pouvez créer une intégration zéro ETL à l'aide de l'APIAWS Management Console, deAWS CLI, ou de l'API RDS.

> **Important**
>
> Les intégrations zéro ETL ne prennent pas en charge les opérations d'actualisation ou de resynchronisation. Si vous rencontrez des problèmes avec une intégration après sa création, vous devez supprimer l'intégration et en créer une nouvelle.

Console RDS

Pour créer une intégration zéro ETL

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation de gauche, choisissez Intégrations zéro ETL.
3. Choisissez Créer une intégration zéro ETL.
4. Dans Identifiant d'intégration, saisissez un nom pour l'intégration. Ce nom peut comporter jusqu'à 63 caractères alphanumériques et peut inclure des traits d'union.

   > **Important**
   >
   > Les noms de catalogue sont limités à 19 caractères. Assurez-vous que votre identifiant d'intégration répond à cette exigence s'il doit être utilisé comme nom de catalogue.

5. Choisissez Suivant.
6. Pour Source, sélectionnez le cluster de bases de données Aurora d'où proviendront les données.

> **Note**
>
> RDS vous avertit si les paramètres du cluster de bases de données ne sont pas configurés correctement. Si vous recevez ce message, vous pouvez soit choisir Fix it for me, soit les configurer manuellement. Pour obtenir des instructions pour les corriger manuellement, reportez-vous à the section called "Étape 1 : Création d'un groupe de paramètres de cluster de bases de données personnalisé".
> La modification des paramètres du cluster de bases de données nécessite un redémarrage. Avant de créer l'intégration, le redémarrage doit être terminé et les nouvelles valeurs de paramètre doit être correctement appliquée au cluster.

7. (Facultatif) Sélectionnez Personnaliser les options de filtrage des données et ajoutez des filtres de données à votre intégration. Vous pouvez utiliser des filtres de données pour définir l'étendue de réplication vers l'entrepôt de données cible. Pour de plus amples informations, veuillez consulter the section called "Filtrage des données pour les intégrations zéro ETL".
8. Une fois la configuration du cluster de bases de données source terminée, choisissez Suivant.
9. Pour Cible, procédez comme suit :

    1. (Facultatif) Pour utiliser un autre Compte AWS compte pour la cible Amazon Redshift, choisissez Spécifier un autre compte. Saisissez ensuite l'ARN d'un rôle IAM doté d'autorisations pour afficher vos entrepôts des données. Pour obtenir des instructions sur la création du rôle IAM, consultez the section called "Choix d'un entrepôt de données cible dans un autre compte".
    2. Pour Entrepôt de données Amazon Redshift, sélectionnez la cible pour les données répliquées à partir du cluster de bases de données source. Vous pouvez choisir un cluster Amazon Redshift provisionné ou un espace de noms Redshift sans serveur comme cible.

> **Note**
>
> RDS vous avertit si la politique de ressources ou les paramètres de sensibilité à la casse pour l'entrepôt de données spécifié ne sont pas correctement configurés. Si vous recevez ce message, vous pouvez soit choisir Fix it for me, soit les configurer

manuellement. Pour obtenir des instructions pour les corriger manuellement, consultez Activation de la sensibilité à la casse pour votre entrepôt de données et Configuration de l'autorisation pour votre entrepôt de données dans le Guide de gestion Amazon Redshift.
La modification de la sensibilité à la casse pour un cluster Redshift provisionné nécessite un redémarrage. Avant de créer l'intégration, le redémarrage doit être terminé et la nouvelle valeur de paramètre doit être correctement appliquée au cluster.
Si la source et la cible que vous avez sélectionnées se trouvent dans des Comptes AWS différents, Amazon RDS ne peut pas corriger ces paramètres pour vous. Vous devez accéder à l'autre compte et les corriger manuellement dans Amazon Redshift.

10. Une fois que votre entrepôt de données cible est correctement configuré, choisissez Suivant.
11. (Facultatif) Pour Balises, ajoutez une ou plusieurs balises à l'intégration. Pour plus d'informations, consultez the section called "Marquage des ressources Aurora et RDS".
12. Pour Chiffrement, spécifiez la manière dont vous souhaitez que votre intégration soit chiffrée. Par défaut, RDS chiffre toutes les intégrations avec un. Clé détenue par AWS Pour choisir plutôt une clé gérée par le client, activez Personnaliser les paramètres de chiffrement et choisissez une clé KMS à utiliser pour le chiffrement. Pour plus d'informations, consultez the section called "Chiffrement des ressources Amazon Aurora".

    Ajoutez éventuellement un contexte de chiffrement. Consultez Contexte de chiffrement dans le AWS Key Management Serviceguide du développeur pour en savoir plus.

    > **Note**
    >
    > Amazon RDS ajoute les paires de contextes de chiffrement suivantes en plus de celles que vous ajoutez :
    >
    > - `aws:redshift:integration:arn` - `IntegrationArn`
    > - `aws:servicename:id` - `Redshift`
    >
    > Cela réduit le nombre total de paires que vous pouvez ajouter de 8 à 6 et contribue à la limite de caractères globale de la contrainte de subvention. Pour plus d'informations, consultez Utilisation des contraintes d'octroi dans le Guide du développeur AWS Key Management Service.

13. Choisissez Suivant.
14. Vérifiez vos paramètres d'intégration et choisissez Créer une intégration zéro ETL.

Si la création échoue, consultez the section called “Je ne parviens pas à créer une intégration zéro ETL” pour obtenir les étapes de résolution des problèmes.

L'intégration a un statut de `Creating` lors de sa création et l'entrepôt de données Amazon Redshift cible a un statut de `Modifying`. Pendant ce temps, vous ne pouvez pas interroger l'entrepôt de données ni y apporter aucune modification de configuration.

Quand l'intégration est créée avec succès, le statut de l'intégration et celui de l'entrepôt de données Amazon Redshift cible passent tous deux à `Active`.

AWS CLI

Pour créer une intégration zéro ETL à l'aide deAWS CLI, utilisez la commande create-integration avec les options suivantes :

> **Note**
>
> N’oubliez pas que les noms de catalogue sont limités à 19 caractères. Choisissez le nom de votre intégration en conséquence s’il doit être utilisé comme nom de catalogue.

- `--integration-name` : spécifiez le nom de l’intégration.
- `--source-arn` : spécifiez l’ARN du cluster de bases de données Aurora qui sera la source de l’intégration.
- `--target-arn` : spécifiez l’ARN de l’entrepôt de données Amazon Redshift qui sera la cible de l’intégration.

Example

Pour Linux, macOS ou Unix :

```
aws rds create-integration \
    --integration-name my-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:my-db \
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

Pour Windows :

```
aws rds create-integration ^
    --integration-name my-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:my-db ^
    --target-arn arn:aws:redshift:{region}:{account-id}:namespace:namespace-uuid
```

API RDS

Pour créer une intégration zéro ETL à l'aide de l'API Amazon RDS, utilisez l'opération `CreateIntegration` avec les paramètres suivants :

> **Note**
>
> Les noms de catalogue sont limités à 19 caractères. Assurez-vous que votre IntegrationName paramètre répond à cette exigence s'il doit être utilisé comme nom de catalogue.

- `IntegrationName` : spécifiez le nom de l'intégration.
- `SourceArn` : spécifiez l'ARN du cluster de bases de données Aurora qui sera la source de l'intégration.
- `TargetArn` : spécifiez l'ARN de l'entrepôt de données Amazon Redshift qui sera la cible de l'intégration.

Chiffrement des intégrations avec une clé gérée par le client

Si vous spécifiez une clé KMS personnalisée plutôt qu'une clé Clé détenue par AWS lorsque vous créez une intégration, la politique de clé doit fournir au service Amazon Redshift l'accès principal à l'`CreateGrant`action. En outre, elle doit autoriser l'utilisateur actuel à effectuer les actions `DescribeKey` et `CreateGrant`.

L'exemple de stratégie suivant montre comment fournir les autorisations requises dans la stratégie de clé. Il inclut des clés contextuelles pour réduire davantage la portée des autorisations.

Exemple de stratégie de clé

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "Enables IAM user permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allows the Redshift service principal to add a grant to a KMS
key",
            "Effect": "Allow",
            "Principal": {
                "Service": "redshift.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}": "{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current user or role to add a grant to a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}": "{context-value}",
```

```
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "Allows the current uer or role to retrieve information about
 a KMS key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

Pour plus d'informations, consultez Création d'une stratégie de clé dans le Guide du développeur AWS Key Management Service.

Étapes suivantes

Une fois que vous avez réussi à créer une intégration zéro ETL, vous devez créer une base de données de destination au sein de votre cluster ou groupe de travail Amazon Redshift cible. Ensuite, vous pouvez commencer à ajouter des données au cluster de bases de données Aurora source et à les interroger dans Amazon Redshift. Pour obtenir des instructions, consultez Création de bases de données de destination dans Amazon Redshift.

Création d'intégrations zéro ETL Aurora avec un Amazon SageMaker Lakehouse

Lorsque vous créez une intégration Aurora Zero-ETL avec un Amazon SageMaker Lakehouse, vous spécifiez le cluster de DB source et le catalogue géré cible. AWS Glue Vous pouvez également

personnaliser les paramètres de chiffrement et ajouter des balises. Aurora crée une intégration entre le cluster de bases de données source et sa cible. Une fois l'intégration active, toutes les données que vous insérez dans le cluster de bases de données source sont répliquées dans la cible configurée.

Conditions préalables

Avant de créer une intégration zéro ETL avec un Amazon SageMaker Lakehouse, vous devez créer un cluster de source et un catalogue géré cibleAWS Glue. Vous devez également autoriser la réplication dans le catalogue en ajoutant le cluster de bases de données en tant que source d'intégration autorisée.

Pour obtenir des instructions sur la réalisation de chacune de ces étapes, consultez the section called "Bien démarrer avec les intégrations zéro ETL".

Autorisations requises

Certaines autorisations IAM sont requises pour créer une intégration zéro ETL avec un Amazon SageMaker Lakehouse. Vous avez au moins besoin des autorisations requises pour effectuer les actions suivantes :

- Créer des intégrations zéro ETL le cluster de bases de données Aurora source.
- Afficher et supprimer toutes les intégrations zéro ETL.
- Créez des intégrations entrantes dans le catalogue AWS Glue géré cible.
- Accédez aux compartiments Amazon S3 utilisés par le catalogue AWS Glue géré.
- Utilisez AWS KMS des clés pour le chiffrement si le chiffrement personnalisé est configuré.
- Enregistrez les ressources auprès de Lake Formation.
- Appliquez une politique de ressources au catalogue AWS Glue géré pour autoriser les intégrations entrantes.

L'exemple de politique suivant illustre les autorisations de moindre privilège requises pour créer et gérer des intégrations avec un Amazon SageMaker Lakehouse. Il se peut que vous n'ayez pas besoin de ces autorisations exactes si votre utilisateur ou votre rôle dispose d'autorisations plus étendues, telles qu'une politique gérée `AdministratorAccess`.

En outre, vous devez configurer une politique de ressources sur le catalogue AWS Glue géré cible afin d'autoriser les intégrations entrantes. Utilisez la AWS CLI commande suivante pour appliquer la politique de ressources.

Exemple de AWS CLI commande pour autoriser les intégrations entrantes sur le catalogue cible

```
aws glue put-resource-policy \
      --policy-in-json  '{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": "glue.amazonaws.com"
        },
        "Action": [
            "glue:AuthorizeInboundIntegration"
        ],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"],
        "Condition": {
            "StringEquals": {
                "aws:SourceArn": "arn:aws:rds:region:account_id:db:source_name"
            }
        }
    },
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "account_id"
        },
        "Action": ["glue:CreateInboundIntegration"],
        "Resource": ["arn:aws:glue:region:account_id:catalog/catalog_name"]
    }
    ]
}' \
      --region region
```

> **Note**
>
> Les noms de ressources Amazon du catalogue Glue (ARNs) ont le format suivant :
>
> - Catalogue Glue : `arn:aws:glue:{region}:{account-id}:catalog/catalog-name`

Choix d'un catalogue AWS Glue géré cible dans un autre compte

Si vous envisagez de spécifier un catalogue AWS Glue géré cible qui se trouve dans un autreCompte AWS, vous devez créer un rôle permettant aux utilisateurs du compte courant d'accéder aux ressources du compte cible. Pour plus d'informations, consultez la section Fournir un accès à un utilisateur IAM dans un autre utilisateur Compte AWS dont vous êtes le propriétaire.

Le rôle doit disposer des autorisations suivantes, qui permettent à l'utilisateur de consulter les AWS Glue catalogues disponibles dans le compte cible.

Autorisations requises et politique d’approbation

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "glue:GetCatalog"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

Le rôle doit respecter la politique d’approbation suivante, qui spécifie l’ID du compte cible.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "AWS": "arn:aws:iam::111122223333:root"
```

```
            },
            "Action":"sts:AssumeRole"
        }
    ]
}
```

Pour obtenir des instructions quant à la création du rôle, consultez Création d’un rôle à l’aide de politiques d’approbation personnalisées.

Création d’Intégrations zéro ETL avec un Amazon SageMaker Lakehouse

Vous pouvez créer une intégration zéro ETL avec un Amazon SageMaker lakehouse à l'aide de l'APIAWS Management Console, deAWS CLI, ou de l'API RDS.

> **Important**
>
> Les intégrations zéro ETL avec un Amazon SageMaker Lakehouse ne prennent pas en charge les opérations d’actualisation ou de resynchronisation. Si vous rencontrez des problèmes avec une intégration après sa création, vous devez supprimer l’intégration et en créer une nouvelle.

Console RDS

Pour créer une intégration zéro ETL avec un Amazon SageMaker Lakehouse

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation de gauche, choisissez Intégrations zéro ETL.
3. Choisissez Créer une intégration zéro ETL.
4. Dans Identifiant d’intégration, saisissez un nom pour l’intégration. Ce nom peut comporter jusqu’à 63 caractères alphanumériques et peut inclure des traits d’union.
5. Choisissez Suivant.
6. Pour Source, sélectionnez le cluster de bases de données Aurora d’où proviendront les données.

> **Note**
>
> RDS vous avertit si les paramètres du cluster de bases de données ne sont pas configurés correctement. Si vous recevez ce message, vous pouvez soit choisir Fix it for me, soit les configurer manuellement. Pour obtenir des instructions pour les corriger manuellement, reportez-vous à the section called “Étape 1 : Création d’un groupe de paramètres de cluster de bases de données personnalisé”.
> La modification des paramètres du cluster de bases de données nécessite un redémarrage. Avant de créer l’intégration, le redémarrage doit être terminé et les nouvelles valeurs de paramètre doit être correctement appliquée au cluster.

7. (Facultatif) Sélectionnez Personnaliser les options de filtrage des données et ajoutez des filtres de données à votre intégration. Vous pouvez utiliser des filtres de données pour définir l’étendue de réplication vers le Amazon SageMaker Lakehouse cible. Pour plus d’informations, consultez the section called “Filtrage des données pour les intégrations zéro ETL”.
8. Une fois la configuration du cluster de bases de données source terminée, choisissez Suivant.
9. Pour Cible, procédez comme suit :

    1. (Facultatif) Pour utiliser un autre Compte AWS compte pour la cible Amazon SageMaker Lakehouse, choisissez Spécifier un autre compte. Entrez ensuite l'ARN d'un rôle IAM autorisé à afficher vos AWS Glue catalogues. Pour obtenir des instructions sur la création du rôle IAM, consultez the section called “Choix d'un catalogue AWS Glue géré cible dans un autre compte”.
    2. Pour Catalogue AWS Glue, sélectionnez la cible pour les données répliquées à partir du cluster de bases de données source. Vous pouvez choisir un catalogue géré AWS Glue existant comme cible.
    3. Le rôle IAM cible doit décrire les autorisations disponibles dans le catalogue cible et doit disposer des autorisations suivantes :

        JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
```

```
            "Action": "glue:GetCatalog",
            "Resource": [
                "arn:aws:glue:us-east-1:111122223333:catalog/*",
                "arn:aws:glue:us-east-1:111122223333:catalog"
            ]
        }
    ]
}
```

Le rôle IAM doit avoir la relation d'approbation suivante :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

4. Vous devez accorder au rôle IAM cible des autorisations de description pour le catalogue AWS Glue géré cible avec le rôle d'administrateur de Lake Formation créé dansÉtape 3b : Création d'un AWS Glue catalogue pour l'intégration Amazon SageMaker AI Zero-ETL.

> **Note**
>
> RDS vous avertit si la politique de ressources ou les paramètres de configuration du catalogue AWS Glue géré spécifié ne sont pas correctement configurés. Si vous recevez ce message, vous pouvez soit choisir Fix it for me, soit les configurer manuellement.
> Si la source et la cible que vous avez sélectionnées se trouvent dans des Comptes AWS différents, Amazon RDS ne peut pas corriger ces paramètres pour vous. Vous devez accéder à l'autre compte et les corriger manuellement dans SageMaker Unified Studio.

10. Une fois que votre catalogue AWS Glue géré cible est correctement configuré, choisissez Next.

11. (Facultatif) Pour Balises, ajoutez une ou plusieurs balises à l'intégration. Pour plus d'informations, consultez the section called "Marquage des ressources Aurora et RDS".
12. Pour Chiffrement, spécifiez la manière dont vous souhaitez que votre intégration soit chiffrée. Par défaut, RDS chiffre toutes les intégrations avec un. Clé détenue par AWS Pour choisir plutôt une clé gérée par le client, activez Personnaliser les paramètres de chiffrement et choisissez une clé KMS à utiliser pour le chiffrement. Pour plus d'informations, consultez the section called "Chiffrement des ressources Amazon Aurora".

    Ajoutez éventuellement un contexte de chiffrement. Consultez Contexte de chiffrement dans le AWS Key Management Serviceguide du développeur pour en savoir plus.

    > **Note**
    >
    > Amazon RDS ajoute les paires de contextes de chiffrement suivantes en plus de celles que vous ajoutez :
    >
    > - `aws:glue:integration:arn` - `IntegrationArn`
    > - `aws:servicename:id` - `glue`
    >
    > Cela réduit le nombre total de paires que vous pouvez ajouter de 8 à 6 et contribue à la limite de caractères globale de la contrainte de subvention. Pour plus d'informations, consultez Utilisation des contraintes d'octroi dans le Guide du développeur AWS Key Management Service.

13. Choisissez Suivant.
14. Vérifiez vos paramètres d'intégration et choisissez Créer une intégration zéro ETL.

    Si la création échoue, consultez the section called "Résolution des problèmes liés aux intégrations zéro ETL" pour obtenir les étapes de résolution des problèmes.

L'intégration a un statut de `Creating` lors de sa création et l'Amazon SageMaker Lakehouse cible a un statut de `Modifying`. Pendant ce temps, vous ne pouvez pas interroger le catalogue ni y apporter aucune modification de configuration.

Quand l'intégration est créée avec succès, le statut de l'intégration et celui de l'Amazon SageMaker Lakehouse cible passent tous deux à `Active`.

AWS CLI

Pour préparer un catalogue AWS Glue géré cible pour une intégration zéro ETL à l'aide deAWS CLI, vous devez d'abord utiliser la create-integration-resource-propertycommande avec les options suivantes :

- `--resource-arn`— Spécifiez l'ARN du catalogue AWS Glue géré qui sera la cible de l'intégration.
- `--target-processing-properties`— Spécifiez l'ARN du rôle IAM pour accéder au catalogue AWS Glue géré cible

```
aws glue create-integration-resource-property --region us-east-1
 --resource-arn arn:aws:glue:region:account_id:catalog/catalog_name \
 --target-processing-properties '{"RoleArn" : "arn:aws:iam::account_id:role/
TargetIamRole"}'
```

Pour créer une intégration zéro ETL avec un Amazon SageMaker lakehouse à l'aide deAWS CLI, utilisez la commande create-integration avec les options suivantes :

- `--integration-name` : spécifiez le nom de l'intégration.
- `--source-arn` : spécifiez l'ARN du cluster de bases de données Aurora qui sera la source de l'intégration.
- `--target-arn`— Spécifiez l'ARN du catalogue AWS Glue géré qui sera la cible de l'intégration.

Example

Pour Linux, macOS ou Unix :

```
aws rds create-integration \
    --integration-name my-sagemaker-integration \
    --source-arn arn:aws:rds:{region}:{account-id}:cluster:my-db \
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

Pour Windows :

```
aws rds create-integration ^
    --integration-name my-sagemaker-integration ^
    --source-arn arn:aws:rds:{region}:{account-id}:cluster:my-db ^
    --target-arn arn:aws:glue:{region}:{account-id}:catalog/catalog-name
```

API RDS

Pour créer une intégration zéro ETL à Amazon SageMaker à l'aide de l'API Amazon RDS, utilisez l'opération `CreateIntegration` avec les paramètres suivants :

> **Note**
>
> Les noms de catalogue sont limités à 19 caractères. Assurez-vous que votre IntegrationName paramètre répond à cette exigence s'il doit être utilisé comme nom de catalogue.

- `IntegrationName` : spécifiez le nom de l'intégration.
- `SourceArn` : spécifiez l'ARN du cluster de bases de données Aurora qui sera la source de l'intégration.
- `TargetArn`— Spécifiez l'ARN du catalogue AWS Glue géré qui sera la cible de l'intégration.

Chiffrement des intégrations avec une clé gérée par le client

Si vous spécifiez une clé KMS personnalisée plutôt qu'une clé Clé détenue par AWS lorsque vous créez une intégrationAmazon SageMaker, la politique de clé doit fournir au SageMaker Unified Studio service principal un accès à l'`CreateGrant`action. En outre, elle doit autoriser l'utilisateur actuel à effectuer les actions `DescribeKey` et `CreateGrant`.

L'exemple de stratégie suivant montre comment fournir les autorisations requises dans la stratégie de clé. Il inclut des clés contextuelles pour réduire davantage la portée des autorisations.

Exemple de stratégie de clé

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Key policy",
    "Statement": [
        {
            "Sid": "EnablesIAMUserPermissions",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
```

```
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "GlueServicePrincipalAddGrant",
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
                }
            }
        },
        {
            "Sid": "AllowsCurrentUserRoleAddGrantKMSKey",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:EncryptionContext:{context-key}":"{context-value}",
                    "kms:ViaService": "rds.us-east-1.amazonaws.com"
                },
                "ForAllValues:StringEquals": {
                    "kms:GrantOperations": [
                        "Decrypt",
                        "GenerateDataKey",
                        "CreateGrant"
                    ]
```

```
                }
            }
        },
        {
            "Sid": "AllowsCurrentUserRoleRetrieveKMSKeyInformation",
            "Effect": "Allow",
            "Principal": {
            "AWS": "arn:aws:iam::111122223333:role/{role-name}"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        }
    ]
}
```

Pour plus d'informations, consultez Création d'une stratégie de clé dans le Guide du développeur AWS Key Management Service.

Étapes suivantes

Une fois que vous avez créé avec succès une intégration zéro ETL avec Amazon SageMaker, vous pouvez commencer à ajouter des données au cluster de bases de données cluster Aurora source et à les interroger dans votre Amazon SageMaker Lakehouse. Les données seront automatiquement répliquées et mises à disposition pour les charges de travail d'analytique et de machine learning.

Filtrage des données pour les intégrations zéro ETL Aurora

Les intégrations zéro ETL Aurora prennent en charge le filtrage des données, ce qui vous permet de contrôler quelles données sont répliquées depuis votre cluster de bases de données Aurora source vers votre entrepôt de données cible. Au lieu de répliquer l'intégralité de la base de données, vous pouvez appliquer un ou plusieurs filtres pour inclure ou exclure des tables spécifiques de manière sélective. Cela vous permet d'optimiser les performances de stockage et de requête en garantissant que seules les données pertinentes sont transférées. Actuellement, le filtrage est limité aux niveaux de base de données et de table. Le filtrage au niveau des colonnes et des lignes n'est pas pris en charge.

Le filtrage des données peut être utile lorsque vous souhaitez :

- Joindre certaines tables provenant d'au moins deux bases de données source différents, et vous n'avez pas besoin de données complètes provenant de l'un ou l'autre des bases de données.

- Réduisez les coûts en effectuant des opérations d'analytique en utilisant uniquement un sous-ensemble de tables plutôt qu'une flotte complète de bases de données.
- Filtrez les informations sensibles, telles que les numéros de téléphone, les adresses ou les informations de carte de crédit, de certaines tables.

Vous pouvez ajouter des filtres de données à une intégration sans ETL à l' AWS Management Console aide de l' AWS Command Line Interface API,AWS CLI the () ou Amazon RDS.

Si l'intégration a un cluster provisionné comme cible, le cluster doit être sur le correctif 180 ou supérieur pour utiliser le filtrage des données.

Rubriques

Format d'un filtre de données

Vous pouvez définir plusieurs filtres pour une seule intégration. Chaque filtre inclut ou exclut les tables de base de données existantes et futures qui correspondent à l'un des modèles de l'expression du filtre. Les intégrations zéro ETL Aurora utilisent la syntaxe du filtre Maxwell pour le filtrage des données.

Chaque filtre contient les éléments suivants :

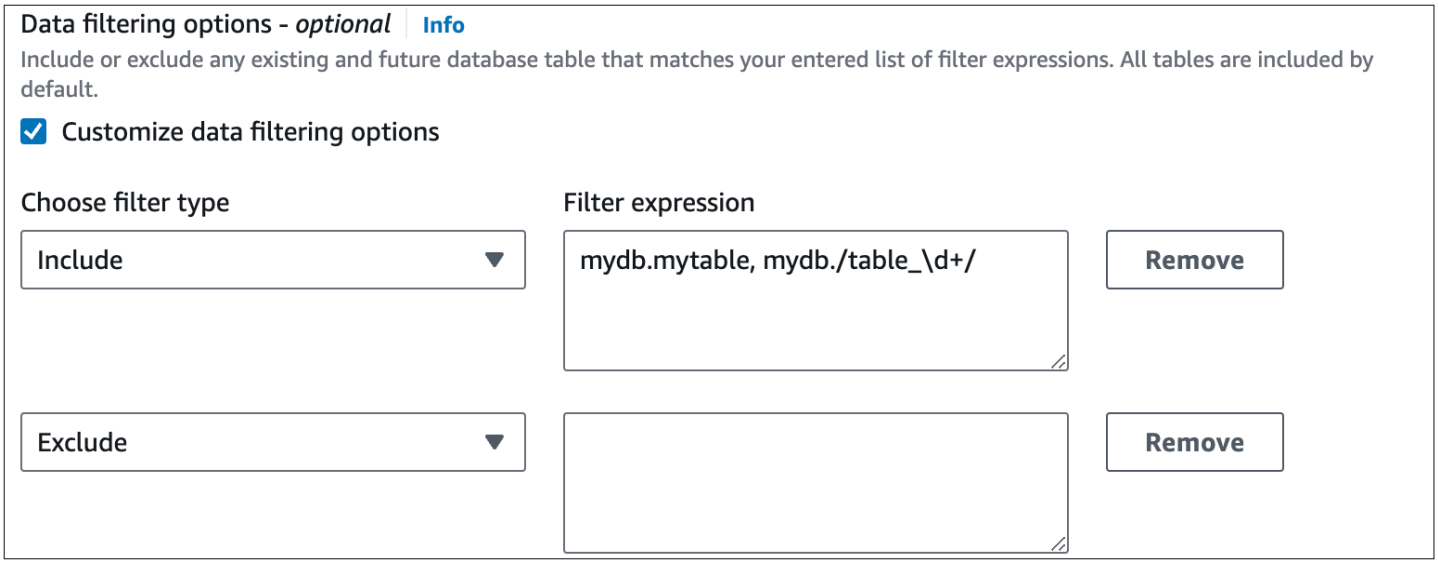
Element	Description
Type de filtre	Un type de filtre `Include` inclut toutes les tables qui correspondent à l'un des modèles de l'expression du filtre. Un type de filtre

Element	Description
	`Exclude` exclut toutes les tables correspon dant à l'un des modèles.
Expression de filtre	Une liste de modèles séparée par des virgules. Les expressions doivent utiliser la syntaxe du filtre Maxwell.

Element	Description
Modèle	Un modèle de filtre au format *`database`*`.`*`table`* pour Aurora MySQL, ou *`database`*`.`*`schema`*`.`*`table`* pour Aurora PostgreSQL. Vous pouvez spécifier des noms littéraux ou définir des expressions régulières. **Note** Pour Aurora MySQL, les expressions régulières sont prises en charge à la fois dans le nom de la base de données et dans le nom de la table. Pour Aurora PostgreSQL, les expressions régulières ne sont prises en charge que dans le schéma et le nom de la table, et non dans le nom de la base de données. Vous ne pouvez pas inclure de filtres ou de listes de refus au niveau des colonnes. Une intégration unique peut avoir un maximum de 99 modèles. Dans la console, vous pouvez saisir des modèles dans une seule expression de filtre ou les répartir entre plusieurs expressions. La longueur d'un modèle unique ne peut pas dépasser 256 caractères.

> **Important**
>
> Si vous sélectionnez un cluster de bases de données Aurora PostgreSQL source, vous devez spécifier au moins un modèle de filtre de données. Le modèle doit au minimum inclure une seule base de données (*`database-name`*`.*.*`) pour la réplication vers l'entrepôt de données cible.

L'image suivante montre la structure des filtres de données Aurora MySQL dans la console :



> **Important**
>
> N'incluez pas de données d'identification personnelle, confidentielles ou sensibles dans vos modèles de filtrage.

Filtres de données dans le AWS CLI

Lorsque vous utilisez le AWS CLI pour ajouter un filtre de données, la syntaxe est légèrement différente de celle de la console. Vous devez attribuer un type de filtre (`Include` ou `Exclude`) à chaque modèle individuellement, afin de ne pas pouvoir regrouper plusieurs modèles sous un même type de filtre.

Par exemple, dans la console, vous pouvez regrouper les modèles suivants, séparés par des virgules, sous une seule instruction `Include` :

Aurora MySQL

```
mydb.mytable, mydb./table_\d+/
```

Aurora PostgreSQL

```
mydb.myschema.mytable, mydb.myschema./table_\d+/
```

Toutefois, lorsque vous utilisez le AWS CLI, le même filtre de données doit être au format suivant :

Aurora MySQL

```
'include: mydb.mytable, include: mydb./table_\d+/'
```

Aurora PostgreSQL

```
'include: mydb.myschema.mytable, include: mydb.myschema./table_\d+/'
```

Logique de filtrage

Si vous ne spécifiez aucun filtre de données dans votre intégration, Aurora suppose un filtre par défaut de `include:*.*`, qui réplique toutes les tables dans l'entrepôt de données cible. Toutefois, si vous ajoutez au moins un filtre, la logique par défaut passe à `exclude:*.*`, ce qui exclut toutes les tables par défaut. Cela vous permet de définir explicitement les bases de données et les tables à inclure dans la réplication.

Par exemple, si vous définissez le filtre suivant :

```
'include: db.table1, include: db.table2'
```

Aurora évalue le filtre comme suit :

```
'exclude:*.*, include: db.table1, include: db.table2'
```

Par conséquent, Aurora réplique uniquement `table1` et `table2` depuis la base de données nommée db vers l'entrepôt de données cible.

Ordre de priorité de filtre

Aurora évalue les filtres de données dans l'ordre que vous spécifiez. Dans le AWS Management Console, il traite les expressions de filtre de gauche à droite et de haut en bas. Un second filtre ou un modèle individuel qui suit le premier peut le remplacer.

Par exemple, si le premier filtre est `Include books.stephenking`, il inclut uniquement la table `stephenking` de la base de données `books`. Toutefois, si vous ajoutez un second filtre, `Exclude books.*`, il remplace le premier filtre. Cela empêche la réplication des tables de l'index `books` vers l'entrepôt de données cible.

Lorsque vous spécifiez au moins un filtre, la logique commence par l'hypothèse `exclude:*.*` par défaut, ce qui exclut automatiquement toutes les tables de la réplication. Une bonne pratique consiste à définir des filtres du plus large au plus spécifique. Commencez par une ou plusieurs instructions `Include` pour spécifier les données à répliquer, puis ajoutez des filtres `Exclude` pour supprimer certaines tables de manière sélective.

Le même principe s'applique aux filtres que vous définissez à l'aide de l' AWS CLI. Aurora évalue ces modèles de filtre dans l'ordre dans lequel vous les spécifiez, de sorte qu'un modèle peut remplacer celui que vous avez spécifié avant lui.

Exemples Aurora MySQL

Les exemples suivants montrent comment fonctionne le filtrage des données pour intégrations zéro ETL Exemples Aurora MySQL :

- Incluez toutes les bases de données et toutes les tables :

```
'include: *.*'
```

- Incluez toutes les tables de la base de données `books` :

```
'include: books.*'
```

- Excluez toutes les tables nommées `mystery` :

```
'include: *.*, exclude: *.mystery'
```

- Incluez deux tables spécifiques dans la base de données `books` :

```
'include: books.stephen_king, include: books.carolyn_keene'
```

- Incluez toutes les tables de la base de données `books`, à l'exception de celles contenant la sous-chaîne `mystery` :

```
'include: books.*, exclude: books./.*mystery.*/'
```

- Incluez toutes les tables de la base de données `books`, à l'exception de celles commençant par `mystery` :

```
'include: books.*, exclude: books./mystery.*/'
```

- Incluez toutes les tables de la base de données `books`, à l'exception de celles se terminant par `mystery` :

```
'include: books.*, exclude: books./.*mystery/'
```

- Incluez toutes les tables de la base de données `books` qui commencent par `table_`, à l'exception de celle nommée `table_stephen_king`. Par exemple, `table_movies` ou `table_books` serait répliqué, mais pas `table_stephen_king`.

```
'include: books./table_.*/, exclude: books.table_stephen_king'
```

Exemples Aurora PostgreSQL

Les exemples suivants montrent comment fonctionne le filtrage des données pour intégrations zéro ETL Aurora PostgreSQL :

- Incluez toutes les tables de la base de données `books` :

```
'include: books.*.*'
```

- Excluez toutes les tables nommées `mystery` de la base de données `books` :

```
'include: books.*.*, exclude: books.*.mystery'
```

- Incluez une table dans la base de données `books` dans le schéma `mystery`, et une table dans la base de données `employee` dans le schéma `finance` :

```
'include: books.mystery.stephen_king, include: employee.finance.benefits'
```

- Incluez toutes les tables de la base de données `books` et le schéma `science_fiction`, à l'exception de celles contenant la sous-chaîne `king` :

```
'include: books.science_fiction.*, exclude: books.*./.*king.*/
```

- Incluez toutes les tables de la base de données `books`, à l'exception de celles avec un nom de schéma commençant par `sci` :

```
'include: books.*.*, exclude: books./sci.*/.*'
```

- Incluez toutes les tables de la base de données `books`, à l'exception de celles dans le schéma `mystery` se terminant par `king` :

```
'include: books.*.*, exclude: books.mystery./.*king/'
```

- Incluez toutes les tables de la base de données `books` qui commencent par `table_`, à l'exception de celle nommée `table_stephen_king`. Par exemple, `table_movies` dans le schéma `fiction` et `table_books` dans le schéma `mystery` sont répliqués, mais pas `table_stephen_king` dans l'un ou l'autre des schémas :

```
'include: books.*./table_.*/, exclude: books.*.table_stephen_king'
```

Ajout de filtres de données à une intégration

Vous pouvez configurer le filtrage des données à l'aide de l' AWS Management Console API AWS CLI, de, ou de l'API Amazon RDS.

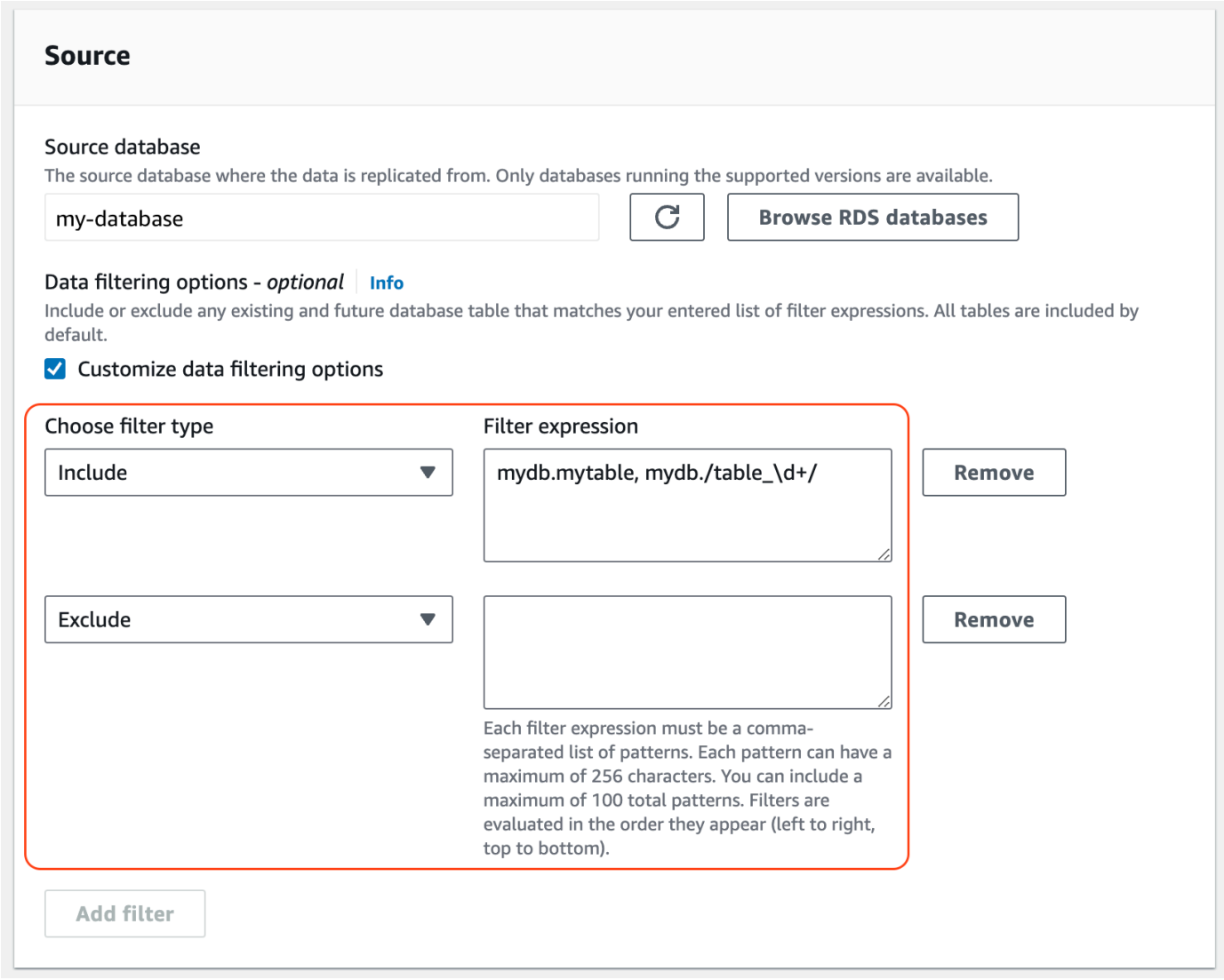
> **Important**
>
> Si vous ajoutez un filtre après avoir créé une intégration, Aurora le traite comme s'il avait toujours existé. Il supprime toutes les données de l'entrepôt de données cible qui ne correspondent pas aux nouveaux critères de filtrage et resynchronise toutes les tables concernées.

Console RDS

Pour ajouter des filtres de données à une intégration zéro ETL

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation de gauche, choisissez Intégrations zéro ETL. Sélectionnez l'intégration à laquelle vous souhaitez ajouter des filtres de données, puis choisissez Modifier.
3. Sous Source, ajoutez une ou plusieurs instructions `Include` et `Exclude`.

   L'image suivante montre un exemple de filtres de données pour une intégration MySQL :



4. Lorsque vous êtes satisfait des modifications, choisissez Continuer et Enregistrer les modifications.

AWS CLI

Pour ajouter des filtres de données à une intégration zéro ETL à l'aide de AWS CLI, appelez la commande modify-integration. Outre l'identifiant d'intégration, spécifiez le paramètre `--data-filter` à l'aide d'une liste séparée par des virgules de filtres Maxwell `Include` et `Exclude`.

Example

L'exemple suivant ajoute des modèles de filtre à `my-integration`.

Pour Linux, macOS ou Unix :

```
aws rds modify-integration \
    --integration-identifier my-integration \
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

Pour Windows :

```
aws rds modify-integration ^
    --integration-identifier my-integration ^
    --data-filter 'include: foodb.*, exclude: foodb.tbl, exclude: foodb./table_\d+/'
```

API RDS

Pour modifier une intégration zéro ETL à l'aide de l'API RDS, appelez l'ModifyIntegrationopération. Spécifiez l'identifiant d'intégration et fournissez une liste de modèles de filtre séparée par des virgules.

Suppression des filtres de données d'une intégration

Lorsque vous supprimez un filtre de données d'une intégration, Aurora réévalue les filtres restants comme si le filtre supprimé n'avait jamais existé. Il réplique ensuite toutes les données précédemment exclues qui répondent désormais aux critères dans l'entrepôt de données cible. Cela déclenche une resynchronisation de toutes les tables concernées.

Ajouter des données à un cluster de bases de données Aurora source et les interroger

Pour finir de créer une intégration zéro ETL qui réplique les données d'Amazon Aurora vers Amazon Redshift, vous devez créer une base de données dans la destination cible.

Pour les connections avec Amazon Redshift, connectez-vous à votre cluster ou groupe de travail Amazon Redshift et créez une base de données avec une référence à votre identifiant d'intégration. Ensuite, vous pouvez ajouter des données dans votre cluster de bases de données Aurora source afin qu'elles soient répliquées dans Amazon Redshift ou Amazon SageMaker.

Rubriques

Création d'une base de données cible

Avant de pouvoir commencer à répliquer des données dans Amazon Redshift, après avoir créé une intégration, vous devez créer une base de données dans votre entrepôt de données cible. Cette base de données de doit inclure une référence à l'identifiant d'intégration. Vous pouvez utiliser la console Amazon Redshift ou l'éditeur de requête v2 pour créer la base de données.

Pour obtenir des instructions sur la création d'une base de données de destination, consultez Création d'une base de données de destination dans Amazon Redshift.

Ajout de données à la base de données source

Après avoir configuré votre intégration, vous pouvez remplir le cluster de bases de données Aurora source avec les données que vous souhaitez répliquer dans votre entrepôt de données.

> **Note**
>
> Il existe des différences entre les types de données dans Amazon Aurora et dans l'entrepôt d'analytique cible. Pour un tableau des mappages de types de données, consultez the section called "Différences de type de données".

Tout d'abord, connectez-vous au cluster de bases de données source à l'aide du client MySQL ou PostgreSQL de votre choix. Pour obtenir des instructions, consultez the section called "Connexion à un cluster de bases de données".

Ensuite, créez une table et insérez une ligne d'exemples de données.

> **Important**
>
> Assurez-vous que la table possède une clé primaire. Sinon, elle ne peut pas être répliquée vers l'entrepôt de données cible.

Les utilitaires PostgreSQL pg_dump et pg_restore créent d'abord des tables sans clé primaire, puis les ajoutent par la suite. Si vous utilisez l'un de ces utilitaires, nous vous recommandons de créer d'abord un schéma, puis de charger les données dans une commande distincte.

MySQL

L'exemple suivant utilise l'utilitaire MySQL Workbench.

```
CREATE DATABASE my_db;

USE my_db;

CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL, Author
 VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));

INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977, 'Supernatural
 fiction');
```

PostgreSQL

L'exemple suivant utilise le terminal interactif PostgreSQL psql. Lorsque vous vous connectez au cluster, incluez la base de données nommée que vous avez spécifiée lors de la création de l'intégration.

```
psql -h mycluster.cluster-123456789012.us-east-2.rds.amazonaws.com -p 5432 -U username
 -d named_db;

named_db=> CREATE TABLE books_table (ID int NOT NULL, Title VARCHAR(50) NOT NULL,
 Author VARCHAR(50) NOT NULL,
Copyright INT NOT NULL, Genre VARCHAR(50) NOT NULL, PRIMARY KEY (ID));
```

```
named_db=> INSERT INTO books_table VALUES (1, 'The Shining', 'Stephen King', 1977,
 'Supernatural fiction');
```

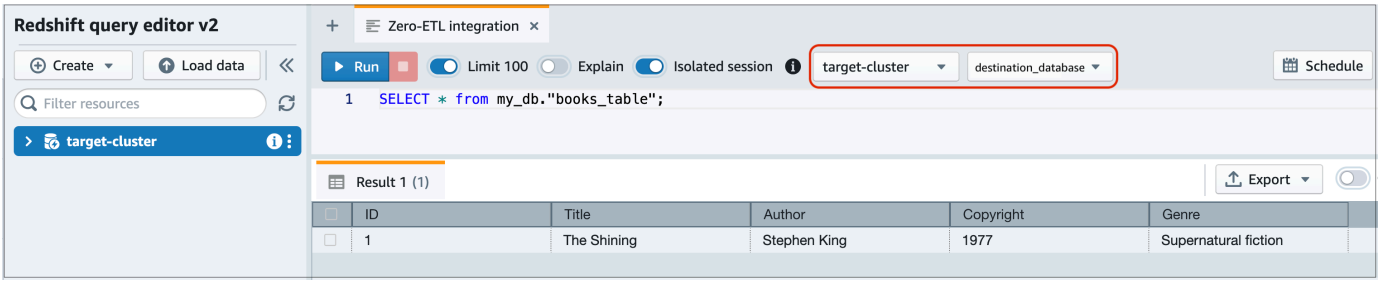
Interrogation de vos données Aurora dans Amazon Redshift

Une fois que vous avez ajouté des données à la base de données RDS, elles sont répliquées dans la base de données de destination et sont prêtes à être interrogées.

Pour interroger les données répliquées

1. Accédez à la console Amazon Redshift et choisissez Éditeur de requête v2 dans le panneau de navigation de gauche.
2. Connectez-vous à votre cluster ou groupe de travail et choisissez votre base de données de destination (que vous avez créée à partir de l'intégration) dans le menu déroulant (destination_database dans cet exemple). Pour obtenir des instructions sur la création d'une base de données de destination, consultez Création d'une base de données de destination dans Amazon Redshift.
3. Utilisez une instruction SELECT pour interroger vos données. Dans cet exemple, vous pouvez exécuter la commande suivante pour sélectionner toutes les données de la table que vous avez créée dans le cluster de bases de données Aurora source :

```
SELECT * from my_db."books_table";
```



- *my_db* est le nom du schéma de base de données Aurora. Cette option n'est nécessaire que pour les bases de données MySQL.
- *books_table* est le nom de la table Aurora.

Vous pouvez également interroger les données à l'aide d'un client de ligne de commande : Par exemple :

```
destination_database=# select * from my_db."books_table";

 ID |       Title |        Author |   Copyright |                  Genre |  txn_seq |
 txn_id
----+-------------+---------------+-------------+------------------------+----------
+--------+
  1 | The Shining |  Stephen King |        1977 |   Supernatural fiction |        2 |
 12192
```

> **Note**
>
> Pour appliquer la sensibilité à la casse, utilisez des guillemets doubles (« ») pour les noms de schéma, de table et de colonne. Pour plus d'informations, consultez enable_case_sensitive_identifier.

Différences de type de données entre les bases de données Aurora et Amazon Redshift

Les tableaux suivants montrent les mappages des types de données Aurora MySQL et Aurora PostgreSQL avec les types de données de destination correspondants. Amazon Aurora ne prend actuellement en charge que ces types de données pour les intégrations zéro ETL.

Si une table de votre cluster de bases de données source inclut un type de données non pris en charge, la table est désynchronisée et n'est pas consommable par la destination cible. Le streaming de la source vers la cible se poursuit, mais le tableau contenant le type de données non pris en charge n'est pas disponible. Pour corriger la table et la mettre à disposition dans la destination cible, vous devez annuler manuellement la modification importante, puis actualiser l'intégration en exécutant `ALTER DATABASE...INTEGRATION REFRESH`.

> **Note**
>
> Vous ne pouvez pas actualiser les intégrations zéro ETL à un Amazon SageMaker Lakehouse. Supprimez plutôt l'intégration et essayez de la créer à nouveau.

Rubriques

- Aurora MySQL

- Aurora PostgreSQL

Aurora MySQL

Type de données Aurora MySQL	Types de données cibles	Description	Limitations
INT	INTEGER	Entier signé sur quatre octets	Aucun
SMALLINT	SMALLINT	Entier signé sur deux octets	Aucun
TINYINT	SMALLINT	Entier signé sur deux octets	Aucun
MEDIUMINT	INTEGER	Entier signé sur quatre octets	Aucun
BIGINT	BIGINT	Entier signé sur huit octets	Aucun
INT UNSIGNED	BIGINT	Entier signé sur huit octets	Aucun
TINYINT UNSIGNED	SMALLINT	Entier signé sur deux octets	Aucun
MEDIUMINT UNSIGNED	INTEGER	Entier signé sur quatre octets	Aucun
BIGINT UNSIGNED	DECIMAL(20,0)	Valeur numérique exacte avec précision sélectionnable	Aucun
DECIMAL(p,s) = NUMERIC(p,s)	DECIMAL(p,s)	Valeur numérique	Une précision supérieure

Type de données Aurora MySQL	Types de données cibles	Description	Limitations
		exacte avec précision sélectionnable	à 38 et une mise à l'échelle supérieure à 37 ne sont pas prises en charge
DECIMAL(p,s) UNSIGNED = NUMERIC(p,s) UNSIGNED	DECIMAL(p,s)	Valeur numérique exacte avec précision sélectionnable	Une précision supérieure à 38 et une mise à l'échelle supérieure à 37 ne sont pas prises en charge
FLOAT4/REAL	REAL	Nombre à virgule flottante simple précision	Aucun
FLOAT4/REAL UNSIGNED	REAL	Nombre à virgule flottante simple précision	Aucun
DOUBLE/REAL/FLOAT8	DOUBLE PRECISION	Nombre à virgule flottante de double précision	Aucun
DOUBLE/REAL/FLOAT8 UNSIGNED	DOUBLE PRECISION	Nombre à virgule flottante de double précision	Aucun
BIT(n)	VARBYTE(8)	Valeur binaire de longueur variable	Aucun
BINARY(n)	VARBYTE(n)	Valeur binaire de longueur variable	Aucun

Type de données Aurora MySQL	Types de données cibles	Description	Limitations
VARBINARY(n)	VARBYTE(n)	Valeur binaire de longueur variable	Aucun
CHAR(n)	VARCHAR(n)	Valeur de chaîne de longueur variable	Aucun
VARCHAR(n)	VARCHAR(n)	Valeur de chaîne de longueur variable	Aucun
TEXT	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun
TINYTEXT	VARCHAR(255)	Valeur de chaîne de longueur variable jusqu'à 255 caracte s	Aucun
MEDIUMTEXT	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun
LONGTEXT	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun

Type de données Aurora MySQL	Types de données cibles	Description	Limitations
ENUM	VARCHAR(1020)	Valeur de chaîne de longueur variable jusqu'à 1 020 caract s	Aucun
SET	VARCHAR(1020)	Valeur de chaîne de longueur variable jusqu'à 1 020 caract s	Aucun
DATE	DATE	Date calendaire (année, mois, jour)	Aucun
DATETIME	TIMESTAMP	Date et heure (sans fuseau horaire)	Aucun
TIMESTAMP(p)	TIMESTAMP	Date et heure (sans fuseau horaire)	Aucun
TIME	VARCHAR(18)	Valeur de chaîne de longueur variable jusqu'à 18 caractères	Aucun
YEAR	VARCHAR(4)	Valeur de chaîne de longueur variable jusqu'à 4 caractères	Aucun

Type de données Aurora MySQL	Types de données cibles	Description	Limitations
JSON	SUPER	Données ou documents semi-structurés sous forme de valeurs	Aucun

Aurora PostgreSQL

Les intégrations zéro ETL pour Aurora PostgreSQL ne prennent pas en charge les types de données personnalisés ni les types de données créés par des extensions.

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
array	SUPER	Données ou documents semi-structurés sous forme de valeurs	Aucun
bigint	BIGINT	Entier signé sur huit octets	Aucun
bigserial	BIGINT	Entier signé sur huit octets	Aucun
bit varying(n)	VARBYTE(n)	Valeur de chaîne de longueur variable jusqu'à 16 777,216	Aucun
bit(n)	VARBYTE(n)	Valeur de chaîne de longueur variable jusqu'à 16 777,216	Aucun

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
bit, bit varying	VARBYTE(16777216)	Valeur de chaîne de longueur variable jusqu'à 16 777,216	Aucun
booléen	BOOLEAN	Booléen logique (true/false)	Aucun
bytea	VARBYTE(16777216)	Valeur de chaîne de longueur variable jusqu'à 16 777,216	Aucun
char(n)	CHAR(n)	Valeur de chaîne de caractères de longueur fixe jusqu'à 65 535 octe	Aucun
char varying(n)	VARCHAR(65535)	Valeur de chaîne de caractères de longueur variable jusqu'à 65 535 cara s	Aucun
cid	BIGINT	Entier signé sur huit octets	Aucun
cidr	VARCHAR(19)	Valeur de chaîne de longueur variable jusqu'à 19 caractèr s	Aucun

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
date	DATE	Date calendaire (année, mois, jour)	Valeurs supérieures à 294 276 après Jésus-Christ non prises en charge
double precision	DOUBLE PRECISION	Nombres à virgule flottante de double précision	Valeurs inférieures à la normale non entièrement prises en charge
gtsvector	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 caractères	Aucun
inet	VARCHAR(19)	Valeur de chaîne de longueur variable jusqu'à 19 caractères	Aucun
entier	INTEGER	Entier signé sur quatre octets	Aucun
int2vector	SUPER	Données ou documents semi-structurés sous forme de valeurs.	Aucun

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
interval	INTERVAL	Durée	Seuls les types INTERVAL qui spécifient un qualificatif annuel par mois ou un qualifica tif journalier par seconde sont pris en charge.
json	SUPER	Données ou documents semi-structurés sous forme de valeurs	Aucun
jsonb	SUPER	Données ou documents semi-structurés sous forme de valeurs	Aucun
jsonpath	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu’à 65 535 cara s	Aucun
macaddr	VARCHAR(17)	Valeur de chaîne de longueur variable jusqu’à 17 caractèr s	Aucun

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
macaddr8	VARCHAR(23)	Valeur de chaîne de longueur variable jusqu'à 23 caractères	Aucun
money	DECIMAL(20,3)	Montant en devises	Aucun
name	VARCHAR(64)	Valeur de chaîne de longueur variable jusqu'à 64 caractères	Aucun

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
numeric(p,s)	DECIMAL(p,s)	Valeur de précision fixe définie par l'utilisateur	• Valeurs NaN non prises en charge • La précision et la mise à l'échelle doivent être définies de manière explicite et ne pas dépasser 38 (précision) et 37 (mise à l'échelle) • Mise à l'échelle négative non prise en charge
oid	BIGINT	Entier signé sur huit octets	Aucun
oidvector	SUPER	Données ou documents semi-structurés sous forme de valeurs.	Aucun

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
pg_brin_bloom_summary	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun
pg_dependencies	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun
pg_lsn	VARCHAR(17)	Valeur de chaîne de longueur variable jusqu'à 17 caractèr s	Aucun
pg_mcv_list	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun
pg_ndistinct	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun
pg_node_tree	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
pg_snapshot	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun
real	REAL	Nombre à virgule flottante simple précision	Valeurs inférieur es à la normale non entièrement prises en charge
refcursor	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun
smallint	SMALLINT	Entier signé sur deux octets	Aucun
smallserial	SMALLINT	Entier signé sur deux octets	Aucun
serial	INTEGER	Entier signé sur quatre octets	Aucun
text	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
tid	VARCHAR(23)	Valeur de chaîne de longueur variable jusqu'à 23 caractères	Aucun
time [(p)] without time zone	VARCHAR(19)	Valeur de chaîne de longueur variable jusqu'à 19 caractères	Valeurs `Infinity` et `-Infinity` non prises en charge
time [(p)] with time zone	VARCHAR(22)	Valeur de chaîne de longueur variable jusqu'à 22 caractères	Valeurs `Infinity` et `-Infinity` non prises en charge
timestamp [(p)] without time zone	TIMESTAMP	Date et heure (sans fuseau horaire)	• Valeurs `Infinity` et `-Infinity` non prises en charge • Valeurs supérieures à `9999-12-31` non prises en charge • Valeurs avant Jésus-Christ non prises en charge

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
timestamp [(p)] with time zone	TIMESTAMPTZ	Date et heure (avec fuseau horaire)	• Valeurs `Infinity` et `-Infinity` non prises en charge • Valeurs supérieures à `9999-12-31` non prises en charge • Valeurs avant Jésus-Christ non prises en charge
TSQUERY	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 caractères	Aucun
tsvector	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 caractères	Aucun
txid_snapshot	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 caractères	Aucun

Type de données Aurora PostgreSQL	Type de données Amazon Redshift	Description	Limitations
uuid	VARCHAR(36)	Chaîne de 36 caractères de longueur variable	Aucun
xid	BIGINT	Entier signé sur huit octets	Aucun
xid8	DECIMAL(20, 0)	Décimal de précision fixe	Aucun
xml	VARCHAR(65535)	Valeur de chaîne de longueur variable jusqu'à 65 535 cara s	Aucun

Opérations DDL pour Aurora PostgreSQL

Amazon Redshift est dérivé de PostgreSQL. Il partage donc plusieurs fonctionnalités avec Aurora PostgreSQL en raison de leur architecture PostgreSQL commune. Les intégrations zéro ETL tirent parti de ces similitudes pour rationaliser la réplication des données de Aurora PostgreSQL vers Amazon Redshift, en mappant les bases de données par nom et en utilisant la base de données, le schéma et la structure de table partagés.

Tenez compte des points suivants lors de la gestion des intégrations zéro ETL Aurora PostgreSQL :

- L'isolation est gérée au niveau de la base de données.
- La réplication s'effectue au niveau de la base de données.
- Les bases de données Aurora PostgreSQL sont mappées aux bases de données Amazon Redshift par leur nom, les données étant transmises à la base de données Redshift renommée correspondante si l'original est renommé.

Malgré leurs similitudes, Amazon Redshift et Aurora PostgreSQL présentent des différences importantes. Les sections suivantes décrivent les réponses du système Amazon Redshift pour les opérations DDL courantes.

Rubriques

Opérations de base de données

Le tableau suivant présente les réponses du système aux opérations DDL de base de données.

Opération DDL	Réponse du système Redshift
`CREATE DATABASE`	Aucune opération
`DROP DATABASE`	Amazon Redshift supprime toutes les données de la base de données Redshift cible.
`RENAME DATABASE`	Amazon Redshift supprime toutes les données de la base de données cible d'origine et resynchronise les données de la nouvelle base de données cible. Si la nouvelle base de données n'existe pas, vous devez la créer manuellement. Pour obtenir des instructions, consultez Création d'une base de données de destination dans Amazon Redshift.

Opérations de schéma

Le tableau suivant présente les réponses du système aux opérations DDL de schéma.

Opération DDL	Réponse du système Redshift
`CREATE SCHEMA`	Aucune opération

Opération DDL	Réponse du système Redshift
`DROP SCHEMA`	Amazon Redshift supprime le schéma d'origine.
`RENAME SCHEMA`	Amazon Redshift supprime le schéma d'origine puis resynchronise les données dans le nouveau schéma.

Opérations de table

Le tableau suivant présente les réponses du système aux opérations DDL de table.

Opération DDL	Réponse du système Redshift
`CREATE TABLE`	Amazon Redshift crée la table. Certaines opérations entraînent l'échec de la création de tables, telles que la création d'une table sans clé primaire ou le partitionnement déclaratif. Pour plus d'informations, consultez the section called "Limitations" et the section called "Résolution des problèmes liés aux intégrations zéro ETL".
`DROP TABLE`	Amazon Redshift supprime la table.
`TRUNCATE TABLE`	Amazon Redshift tronque la table.
`ALTER TABLE (RENAME... )`	Amazon Redshift renomme la table ou la colonne.
`ALTER TABLE (SET SCHEMA)`	Amazon Redshift supprime la table dans le schéma d'origine, puis resynchronise la table dans le nouveau schéma.
`ALTER TABLE (ADD PRIMARY KEY)`	Amazon Redshift ajoute une clé primaire et resynchronise la table.

Opération DDL	Réponse du système Redshift
`ALTER TABLE (ADD COLUMN)`	Amazon Redshift ajoute une colonne à la table.
`ALTER TABLE (DROP COLUMN)`	Amazon Redshift supprime la colonne s'il ne s'agit pas d'une colonne de clé primaire. Dans le cas contraire, il resynchronise la table.
`ALTER TABLE (SET LOGGED/UNLOGGED )`	Si vous remplacez la table par Journalisé, Amazon Redshift la resynchronise. Si vous remplacez la table par Non journalisé, Amazon Redshift la supprime.

Affichage et surveillance des intégrations zéro ETL Aurora

Vous pouvez afficher les détails d'une intégration zéro ETL d'Amazon Aurora pour voir ses informations de configuration et son statut actuel. Vous pouvez également surveiller le statut de votre intégration en interrogeant des vues système spécifiques dans Amazon Redshift. En outre, Amazon Redshift publie certaines métriques liées à l'intégration sur Amazon CloudWatch, que vous pouvez consulter dans la console Amazon Redshift.

Rubriques

- Affichage des intégrations
- Surveillance des intégrations à l'aide des tables système pour Amazon Redshift
- Surveillance des intégrations avec Amazon EventBridge pour Amazon Redshift

Affichage des intégrations

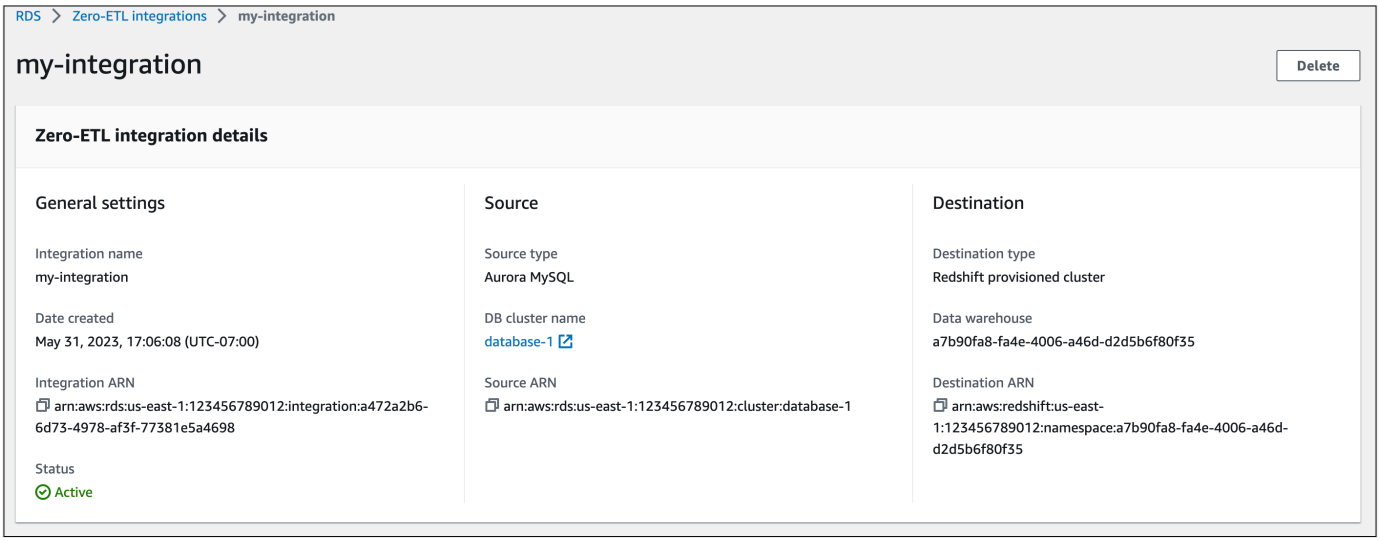
Vous pouvez consulter les intégrations Aurora Zero-ETL à l'aide de l'API AWS Management Console, de ou de l' AWS CLI API RDS.

Console

Pour afficher les détails d'une intégration zéro ETL

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.

2. Dans le panneau de navigation de gauche, choisissez Intégrations zéro ETL.
3. Sélectionnez une intégration pour afficher plus de détails à son sujet, tels que son cluster de bases de données source et son entrepôt de données cible.



Une intégration peut avoir les statuts suivants :

- `Creating` : l'intégration est en cours de création.
- `Active` : l’intégration envoie des données transactionnelles à l’entrepôt de données cible.
- `Syncing` : l'intégration a rencontré une erreur récupérable et réensemence les données. Les tables concernées ne peuvent pas être interrogées tant que leur resynchronisation n’est pas terminée.
- `Needs attention` : l’intégration a rencontré un événement ou une erreur nécessitant une intervention manuelle pour être résolu. Pour corriger le problème, suivez les instructions du message d'erreur dans la page des détails relatifs à l'intégration.
- `Failed` : l'intégration a rencontré un événement ou une erreur irrécupérable qui ne peut pas être corrigé. Vous devez supprimer et recréer l'intégration.
- `Deleting` : l’intégration est en cours de suppression.

AWS CLI

Pour afficher toutes les intégrations Zero-ETL du compte courant à l'aide de AWS CLI, utilisez la commande describe-integrations et spécifiez l'option. `--integration-identifier`

Example

Pour Linux, macOS ou Unix :

```
aws rds describe-integrations \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Pour Windows :

```
aws rds describe-integrations ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

API RDS

Pour afficher une intégration zéro ETL à l'aide de l'API Amazon RDS, utilisez l'opération `DescribeIntegrations` avec le paramètre `IntegrationIdentifier`.

Surveillance des intégrations à l’aide des tables système pour Amazon Redshift

Amazon Redshift comporte des tables et vues système contenant des informations sur le fonctionnement du système. Vous pouvez interroger ces tables et vues système de la même manière que vous interrogez toute autre table de base de données. Pour plus d’informations sur les vues et les tables système dans Amazon Redshift, consultez Informations de référence sur les tables et les vues système dans le Guide du développeur de base de données Amazon Redshift.

Vous pouvez interroger les vues et tables système suivantes pour obtenir des informations sur vos intégrations zéro ETL Aurora :

- SVV_INTEGRATION : fournit les détails de configuration de vos intégrations.
- SVV_INTEGRATION_TABLE_STATE : décrit l'état de chaque table au sein d'une intégration.
- SYS_INTEGRATION_TABLE_STATE_CHANGE : affiche les journaux de changement d'état des tables pour une intégration.
- SYS_INTEGRATION_ACTIVITY : fournit des informations sur les cycles d'intégration terminés.

Toutes les CloudWatch métriques Amazon liées à l'intégration proviennent d'Amazon Redshift. Pour plus d’informations, consultez la section Métriques pour les intégrations zéro ETL dans le Guide de

gestion Amazon Redshift. Actuellement, (Amazon Aurora) ne publie aucune métrique d'intégration sur CloudWatch.

Surveillance des intégrations avec Amazon EventBridge pour Amazon Redshift

Amazon Redshift envoie des événements liés à l'intégration à Amazon. EventBridge Pour obtenir la liste des événements et de l'événement correspondant IDs, consultez la section Notifications d'événements d'intégration Zero-ETL avec Amazon EventBridge dans le guide de gestion Amazon Redshift.

Modification des intégrations zéro ETL Aurora

Vous pouvez uniquement modifier le nom, la description et les options de filtrage des données pour une intégration zéro ETL dans un entrepôt de données pris en charge. Vous ne pouvez pas modifier la clé AWS KMS utilisée pour chiffrer l'intégration, ni les bases de données source ou cible.

Si vous ajoutez un filtre de données à une intégration existante, Aurora réévalue le filtre comme s'il avait toujours existé. Il supprime toutes les données qui se trouvent actuellement dans l'entrepôt de données cible et qui ne correspondent pas aux nouveaux critères de filtrage. Si vous supprimez un filtre de données d'une intégration, il réplique toutes les données qui ne répondaient pas auparavant aux critères de filtrage (mais qui le sont désormais) dans l'entrepôt de données cible. Pour plus d'informations, consultez the section called "Filtrage des données pour les intégrations zéro ETL".

Vous pouvez modifier une intégration zéro ETL à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API Amazon RDS.

Console RDS

Pour modifier une intégration zéro ETL

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez intégrations zéro ETL, puis l'intégration que vous souhaitez modifier.
3. Choisissez Modifier et apportez des modifications aux paramètres disponibles.
4. Une fois que toutes les modifications correspondent à vos attentes, choisissez Modifier.

AWS CLI

Pour modifier une intégration zéro ETL à l'aide de AWS CLI, appelez la commande modify-integration. Avec `--integration-identifier`, spécifiez l'une des options suivantes :

- `--integration-name` : spécifiez un nouveau nom pour l'intégration.
- `--description` : spécifiez une nouvelle description pour l'intégration.
- `--data-filter` : spécifiez les options de filtrage des données pour l'intégration. Pour plus d'informations, consultez the section called "Filtrage des données pour les intégrations zéro ETL".

Example

La demande suivante modifie une intégration existante.

Pour Linux, macOS ou Unix :

```
aws rds modify-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 \
    --integration-name my-renamed-integration
```

Pour Windows :

```
aws rds modify-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374 ^
    --integration-name my-renamed-integration
```

API RDS

Pour modifier une intégration zéro ETL à l'aide de l'API RDS, appelez l'opération ModifyIntegration. Spécifiez l'identifiant d'intégration et les paramètres que vous souhaitez modifier.

Suppression d'intégrations zéro ETL Aurora

Lorsque vous supprimez une intégration zéro ETL, Amazon Aurora la supprime du cluster de bases de données Aurora source. Vos données transactionnelles ne sont pas supprimées d'Amazon Aurora ou de la destination d'analytique, mais Aurora n'envoie pas de nouvelles données à Amazon Redshift ou Amazon SageMaker.

Vous ne pouvez supprimer une intégration que si son état est `Active`, `Failed`, `Syncing` ou `Needs attention`.

Vous pouvez supprimer l'intégration zéro ETL à l'aide de la AWS Management Console, de l'AWS CLI ou de l'API RDS.

Console

Pour supprimer une intégration zéro ETL

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation de gauche, choisissez Intégrations zéro ETL.
3. Sélectionnez l'intégration zéro ETL que vous souhaitez supprimer.
4. Choisissez Actions et Supprimer, puis confirmez la suppression.

AWS CLI

Pour supprimer une intégration zéro ETL, utilisez la commande delete-integration et spécifiez l'option `--integration-identifier`.

Example

Pour Linux, macOS ou Unix :

```
aws rds delete-integration \
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

Pour Windows :

```
aws rds delete-integration ^
    --integration-identifier ee605691-6c47-48e8-8622-83f99b1af374
```

API RDS

Pour supprimer une intégration zéro ETL à l'aide de l'API Amazon RDS, utilisez l'opération `DeleteIntegration` avec le paramètre `IntegrationIdentifier`.

Résolution des problèmes liés aux intégrations zéro ETL Aurora

Vous pouvez vérifier l'état d'une intégration zéro ETL en interrogeant la table système SVV_INTEGRATION dans la destination d'analytique. Si la valeur de la colonne `state` est `ErrorState`, cela signifie que quelque chose ne va pas. Pour plus d'informations, consultez the section called "Surveillance à l'aide des tables système".

Utilisez les informations suivantes pour résoudre les problèmes courants liés aux intégrations zéro ETL Aurora.

> **Important**
>
> Les opérations de resynchronisation et d'actualisation ne sont pas disponibles pour les intégrations zéro ETL à un Amazon SageMaker AI Lakehouse. En cas de problème avec une intégration, vous devez la supprimer et en créer une nouvelle. Vous ne pouvez pas actualiser ou resynchroniser une intégration existante.

Rubriques

Je ne parviens pas à créer une intégration zéro ETL

Si vous ne pouvez pas créer une intégration zéro ETL, assurez-vous que les points suivants sont corrects pour votre base de données source :

- La base de données source doit exécuter une version de moteur de base de données prise en charge. Pour une liste des versions prises en charge, consultez the section called "Intégrations sans ETL ".

- Vous avez correctement configuré les paramètres de la base de données. Si les paramètres requis ne sont pas définis correctement ou ne sont pas associés à la base de données, la création échoue. Consultez the section called “Étape 1 : Création d’un groupe de paramètres de cluster de bases de données personnalisé”.

En outre, assurez-vous que les informations suivantes sont correctes pour votre entrepôt de données cible :

- La sensibilité à la casse est activée. Consultez Activation de la sensibilité à la casse pour votre entrepôt de données.
- Vous avez ajouté le principal autorisé et la source d’intégration appropriés. Consultez Configuration de l’autorisation pour votre entrepôt de données Amazon Redshift.
- L’entrepôt de données est chiffré (s’il s’agit d’un cluster provisionné). Consultez Chiffrement de base de données Amazon Redshift.

Mon intégration est bloquée à l’état de **`Syncing`**

Il est possible que votre intégration affiche systématiquement le statut `Syncing` si vous modifiez la valeur de l’un des paramètres de base de données requis.

Pour résoudre ce problème, vérifiez les valeurs des paramètres du groupe de paramètres associé au cluster de bases de données source et assurez-vous qu’elles correspondent aux valeurs requises. Pour plus d’informations, consultez the section called “Étape 1 : Création d’un groupe de paramètres de cluster de bases de données personnalisé”.

Si vous modifiez des paramètres, veillez à redémarrer la le cluster de bases de données pour appliquer les modifications.

Mes tables ne sont pas répliquées sur Amazon Redshift

Si vous n’avez pas de table reflétée dans Amazon Redshift, vous pouvez exécuter la commande suivante pour les resynchroniser :

```
ALTER DATABASE dbname INTEGRATION REFRESH TABLES table1, table2;
```

Pour plus d’informations, consultez ALTER DATABASE dans la référence SQL Amazon Redshift.

Vos données ne sont peut-être pas répliquées, car une ou plusieurs de vos tables sources ne possèdent pas de clé primaire. Le tableau de bord de surveillance d'Amazon Redshift affiche l'état de ces tables comme `Failed` et l'état de l'intégration zéro ETL globale passe à `Needs attention`. Pour résoudre ce problème, vous pouvez identifier une clé existante dans votre table qui peut devenir une clé primaire, ou vous pouvez ajouter une clé primaire synthétique. Pour des solutions détaillées, consultez les ressources suivantes :

- Gestion des tables sans clés primaires lors de la création d'intégrations zéro ETL Amazon Aurora MySQL ou Amazon RDS for MySQL avec Amazon Redshift
- Gestion des tables sans clés primaires lors de la création d'intégrations zéro ETL Amazon Aurora PostgreSQL avec Amazon Redshift

Une ou plusieurs de mes tables Amazon Redshift nécessitent une resynchronisation

L'exécution de certaines commandes sur votre instance de base de données source peut nécessiter la resynchronisation de vos tables. Dans ce cas, la vue système SVV_INTEGRATION_TABLE_STATE affiche un `table_state` de `ResyncRequired`, ce qui signifie que l'intégration doit complètement recharger les données de cette table spécifique depuis MySQL vers Amazon Redshift.

Lorsque la table commence à se resynchroniser, elle passe à l'état `Syncing`. Aucune action manuelle n'est requise pour resynchroniser une table. Pendant la resynchronisation des données d'une table, vous ne pouvez pas accéder à ces données dans Amazon Redshift.

Vous trouverez ci-dessous quelques exemples d'opérations permettant de mettre une table dans un état `ResyncRequired` et les alternatives possibles à envisager.

Opération	exemple	Autrement
Ajout d'une colonne à une position spécifique	`ALTER TABLE table_name ADD COLUMN column_name INTEGER NOT NULL first;`	Amazon Redshift ne prend pas en charge l'ajout de colonnes à des positions spécifiques

Opération	exemple	Autrement
		à l’aide des mots clés `first` et `after`. Si l’ordre des colonnes de la table cible n’est pas critique, ajoutez la colonne à la fin de la table à l’aide d’une commande plus simple : `ALTER TABLE table_name ADD COLUMN column_name column_type ;`

Opération	exemple	Autrement
Ajout d'une colonne d'horodatage avec la valeur par défaut de `CURRENT_TIMESTAMP`	`ALTER TABLE table_name ADD COLUMN column_name TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP;`	La valeur `CURRENT_TIMESTAMP` pour les lignes de la table existante est calculée par Aurora MySQL et ne peut pas être simulée dans Amazon Redshift sans une resynchronisation complète des données de la table. Si possible, remplacez la valeur par défaut par une constante littérale comme `2023-01-01 00:00:15` afin d'éviter toute latence dans la disponibilité de la table.

Opération	exemple	Autrement
Réalisation d'opérations sur plusieurs colonnes au sein d'une seule commande	`ALTER TABLE table_name ADD COLUMN column_1, RENAME COLUMN column_2 TO column_3;`	Envisagez de diviser la commande en deux opérations distinctes, ADD et RENAME, qui ne nécessiteront pas de resynchronisation.

Problèmes liés à l'échec d'intégration pour les intégrations zéro ETL d'Amazon SageMaker AI Lakehouse

Si vous rencontrez des problèmes avec une intégration zéro ETL existante avec un Amazon SageMaker AI Lakehouse, la seule solution consiste à supprimer l'intégration et à en créer une nouvelle. Contrairement aux autres services AWS, les intégrations zéro ETL ne prennent pas en charge les opérations d'actualisation ou de resynchronisation.

Pour résoudre les problèmes d'intégration :

1. Supprimez l'intégration zéro ETL à l'aide de la console, de l'interface de ligne de commande ou de l'API.
2. Vérifiez que les configurations de la base de données source et de l'entrepôt de données cible sont correctes.
3. Créez une nouvelle intégration zéro ETL à la configuration identique ou mise à jour.

Ce processus entraîne une réinitialisation complète du pipeline de données, ce qui peut prendre du temps en fonction de la taille de votre base de données source.

Les modifications DDL sont effectuées dans Amazon Redshift avant que la transaction DDL ne soit terminée pour Aurora PostgreSQL

Les modifications DDL peuvent apparaître dans Amazon Redshift avant la fin d'une opération DDL dans les intégrations zéro ETL Aurora PostgreSQL. Pour plus d'informations, consultez Opérations DDL pour Aurora PostgreSQL.

Utiliser Aurora Serverless v2

Aurora Serverless v2 est une configuration à scalabilité automatique et à la demande pour Amazon Aurora. Aurora Serverless v2 permet d'automatiser les processus de surveillance de la charge de travail et d'ajustement de la capacité de vos bases de données. La capacité est ajustée automatiquement en fonction de la demande des applications. Seules les ressources consommées par vos clusters de bases de données vous sont facturées. Ainsi, Aurora Serverless v2 peut vous aider à respecter votre budget et à éviter de payer pour des ressources informatiques que vous n'utilisez pas.

Ce type d'automatisation est particulièrement utile pour les bases de données multilocataires, les bases de données distribuées, les systèmes de développement et de test, et d'autres environnements présentant des charges de travail très variables et imprévisibles.

Rubriques

Cas d'utilisation d'Aurora Serverless v2

Aurora Serverless v2 prend en charge de nombreux types de charges de travail de bases de données. Ils vont des environnements de développement et de test aux sites Web et applications soumis à des charges de travail imprévisibles, en passant par les applications critiques les plus exigeantes qui nécessitent une évolutivité et une disponibilité élevées.

Aurora Serverless v2 est particulièrement utile pour les cas d'utilisation suivants :

- Charges de travail variables – Vous exécutez des charges de travail présentant des hausses soudaines et imprévisibles en termes d'activité. Par exemple, un site d'informations sur la circulation routière qui pourrait connaître un pic d'activité lorsqu'il commence à pleuvoir. Autre exemple, un site de e-commerce dont le trafic augmente lorsque vous proposez des soldes ou des promotions spéciales. Avec Aurora Serverless v2, la capacité de votre base de données augmente automatiquement pour répondre aux besoins de la charge de pointe de l'application et revient à la normale lorsque la hausse d'activité est terminée. Avec Aurora Serverless v2, vous n'avez plus besoin d'approvisionner de la capacité pour les pics ou la moyenne d'utilisation. Vous pouvez spécifier une limite de capacité supérieure pour faire face au pire des cas, cette capacité n'étant utilisée que si elle est nécessaire.

  La granularité de la mise à l'échelle dans Aurora Serverless v2 vous permet de faire correspondre étroitement la capacité aux besoins de votre base de données. Pour un cluster approvisionné, l'augmentation d'échelle exige d'ajouter une toute nouvelle instance de base de données. Pour un cluster Aurora Serverless v1, l'augmentation d'échelle exige de doubler le nombre d'unités de capacité Aurora (ACU) du cluster, par exemple de 16 à 32 ou de 32 à 64. En revanche, Aurora Serverless v2 peut ajouter la moitié d'une ACU lorsque seul un volume de capacité un peu plus important est nécessaire. Il peut ajouter 0,5, 1, 1,5, 2 ou des moitiés d'ACU supplémentaires en fonction de la capacité supplémentaire nécessaire pour gérer une augmentation de la charge de travail. Il peut également retirer 0,5, 1, 1,5, 2 ou des moitiés d'ACU supplémentaires lorsque la charge de travail diminue et que cette capacité n'est plus nécessaire.
- Applications multilocataires – Avec Aurora Serverless v2, vous n'avez pas à gérer individuellement la capacité de base de données pour chaque application de votre flotte. Aurora Serverless v2 gère la capacité de base de données individuelle pour vous.

  Vous pouvez créer un cluster pour chaque locataire. De cette façon, vous pouvez utiliser des fonctionnalités telles que le clonage, la restauration d'instantané et les bases de données globales Aurora pour améliorer la haute disponibilité et la reprise après sinistre, selon les besoins de chaque locataire.

  Chaque locataire peut avoir des périodes de pointe ou d'inactivité spécifiques en fonction de l'heure de la journée, de la période de l'année, des événements promotionnels, etc. Chaque cluster peut avoir une plage de capacité étendue. De cette façon, les clusters à faible activité entraînent des frais d'instance de base de données minimes. N'importe quel cluster peut rapidement subir une augmentation d'échelle pour gérer les périodes de forte activité.
- Nouvelles applications – Vous déployez une nouvelle application et vous n'êtes pas sûr de la taille de l'instance de base de données dont vous avez besoin. Aurora Serverless v2 vous permet de

configurer un cluster avec une ou plusieurs instances de base de données et faire en sorte que la base de données soit mise à l'échelle automatiquement selon les exigences en capacité de votre application.

- Applications mixtes : supposons que vous ayez une application de traitement des transactions en ligne (OLTP), mais que vous rencontrez régulièrement des pics de trafic de requêtes. En spécifiant les niveaux de promotion pour les instances de base de données Aurora Serverless v2 d'un cluster, vous pouvez configurer votre cluster de sorte que les instances de base de données de lecteur puissent être mises à l'échelle indépendamment de l'instance de base de données d'enregistreur pour gérer la charge supplémentaire. Lorsque le pic d'utilisation diminue, les instances de base de données de lecteur font à nouveau l'objet d'une réduction d'échelle de sorte à correspondre à la capacité de l'instance de base de données d'enregistreur.
- Planification de la capacité : supposons que vous ajustez généralement la capacité de votre base de données ou que vous vérifiez la capacité de base de données optimale pour votre charge de travail, en modifiant les classes de toutes les instances de base de données d'un cluster. Avec Aurora Serverless v2, vous pouvez éviter ces frais administratifs. Vous pouvez déterminer les capacités minimale et maximale appropriées en exécutant la charge de travail et en vérifiant les proportions réelles de mise à l'échelle des instances de base de données.

  Vous pouvez basculer les instances de base de données existantes du mode approvisionné vers Aurora Serverless v2 ou inversement. Dans ce cas, vous n'avez pas besoin de créer un cluster ou une instance de base de données.

  Avec une base de données globale Aurora, vous n'aurez peut-être pas besoin d'autant de capacité pour les clusters secondaires que pour le cluster principal. Vous pouvez utiliser des instances de base de données Aurora Serverless v2 dans les clusters secondaires. De cette façon, la capacité du cluster peut faire l'objet d'une augmentation d'échelle si une région secondaire est promue et prend en charge la charge de travail de votre application.
- Développement et tests : outre l'exécution de vos applications les plus exigeantes, vous pouvez également utiliser Aurora Serverless v2 pour les environnements de développement et de test. Aurora Serverless v2 vous permet de créer des instances de base de données ayant une faible capacité minimale plutôt que d'utiliser des classes d'instance de base de données db.t*. Vous pouvez définir une capacité maximale suffisamment élevée pour que ces instances de base de données puissent toujours exécuter des charges de travail substantielles sans manquer de mémoire. Lorsque la base de données n'est pas utilisée, toutes les instances de base de données font l'objet d'une réduction d'échelle pour éviter des frais inutiles.

> **Tip**
>
> Pour qu'Aurora Serverless v2 soit pratique à utiliser dans les environnements de développement et de test, AWS Management Console fournit le raccourci Easy create (Création facile) lorsque vous créez un cluster. Si vous choisissez l'option Dev/Test, Aurora crée un cluster avec une instance de base de données Aurora Serverless v2 et une plage de capacité standard pour un système de développement et de test.

Utilisation d'Aurora Serverless v2 pour les charges de travail approvisionnées existantes

Supposons que vous ayez déjà une application Aurora s'exécutant sur un cluster approvisionné. Vous pouvez vérifier le fonctionnement de l'application avec Aurora Serverless v2 en ajoutant une ou plusieurs instances de base de données Aurora Serverless v2 au cluster existant en tant qu'instances de base de données de lecteur. Vous pouvez vérifier la fréquence à laquelle les instances de base de données de lecteur font l'objet d'une augmentation ou d'une réduction d'échelle. Vous pouvez utiliser le mécanisme de basculement Aurora pour promouvoir une instance de base de données Aurora Serverless v2 en enregistreur et vérifier comment l'application en lecture/écriture est gérée. De cette façon, vous pouvez basculer avec un temps d'arrêt minimal, sans modifier le point de terminaison utilisé par vos applications clientes. Pour plus de détails sur la procédure de conversion des clusters existants en Aurora Serverless v2, consultez Migration vers Aurora Serverless v2.

Avantages d'Aurora Serverless v2

Aurora Serverless v2 est destiné aux charges de travail variables ou « irrégulières ». Avec ces charges de travail imprévisibles, il est possible que vous ayez des difficultés à planifier le moment où modifier la capacité de votre base de données. Il se peut également que vous rencontriez des difficultés à modifier la capacité assez rapidement à l'aide des mécanismes familiers, tels que l'ajout d'instances de base de données ou la modification de classes d'instance de base de données. Aurora Serverless v2 offre les avantages suivants pour vous aider dans de tels cas d'utilisation :

- Gestion de la capacité plus simple que les clusters approvisionnés : Aurora Serverless v2 réduit les efforts de planification des tailles d'instance de base de données et de redimensionnement des instances de base de données en fonction de l'évolution de la charge de travail. Il réduit également

les efforts déployés pour maintenir une capacité cohérente pour l'ensemble des instances de base de données d'un cluster.

- Mise à l'échelle plus rapide et plus facile en périodes de forte activité : Aurora Serverless v2 met à l'échelle la capacité de calcul et de mémoire selon les besoins, sans perturber les transactions client ou votre charge de travail globale. La possibilité d'utiliser des instances de base de données de lecteur avec Aurora Serverless v2 vous aide à tirer parti de la mise à l'échelle horizontale en plus de la mise à l'échelle verticale. La possibilité d'utiliser des bases de données globales Aurora implique que vous pouvez répartir votre charge de travail en lecture Aurora Serverless v2 sur plusieurs Régions AWS. Cette fonctionnalité est plus pratique que les mécanismes de mise à l'échelle des clusters approvisionnés. Elle est également plus rapide et plus granulaire que les fonctionnalités de mise à l'échelle d'Aurora Serverless v1.
- Rentable en périodes de faible activité : Aurora Serverless v2 vous permet d'éviter de surapprovisionner vos instances de base de données. Aurora Serverless v2 ajoute des ressources par incréments granulaires lorsque les instances de base de données font l'objet d'une augmentation d'échelle. Vous payez uniquement pour les ressources de base de données que vous consommez. L'utilisation des ressources Aurora Serverless v2 est mesurée à la seconde. De cette façon, lorsqu'une instance de base de données fait l'objet d'une réduction d'échelle, l'utilisation réduite des ressources est immédiatement enregistrée.
- Plus grande parité des fonctionnalités avec les clusters approvisionnés : vous pouvez utiliser de nombreuses fonctionnalités Aurora avec Aurora Serverless v2 qui ne sont pas disponibles pour Aurora Serverless v1. Par exemple, avec Aurora Serverless v2, vous pouvez utiliser des instances de base de données de lecteur, des bases de données globales, l'authentification de base de données Gestion des identités et des accès AWS (IAM) et Performance Insights. Vous pouvez également utiliser beaucoup plus de paramètres de configuration qu'avec Aurora Serverless v1.

  Avec Aurora Serverless v2, vous pouvez notamment tirer parti des fonctionnalités suivantes issues des clusters approvisionnés :

  - Instances de base de données de lecteur : Aurora Serverless v2 peut tirer parti des instances de base de données de lecteur pour procéder à une mise à l'échelle horizontale. Lorsqu'un cluster contient une ou plusieurs instances de base de données de lecteur, le cluster peut basculer immédiatement en cas de problème avec l'instance de base de données d'enregistreur. Il s'agit d'une fonctionnalité qui n'est pas disponible avec Aurora Serverless v1.
  - Clusters multi-AZ : vous pouvez répartir les instances de base de données Aurora Serverless v2 d'un cluster sur plusieurs zones de disponibilité. La configuration d'un cluster multi-AZ permet d'assurer la continuité de l'activité, même dans les rares cas de problèmes qui affectent

l'ensemble d'une zone de disponibilité. Il s'agit d'une fonctionnalité qui n'est pas disponible avec Aurora Serverless v1.

- Bases de données globales : vous pouvez utiliser Aurora Serverless v2 avec les bases de données globales Aurora pour créer des copies supplémentaires en lecture seule de votre cluster dans d'autres Régions AWS à des fins de reprise après sinistre.
- RDS Proxy – Vous pouvez utiliser Amazon RDS Proxy pour autoriser vos applications à grouper et à partager des connexions de bases de données pour améliorer leur capacité de mise à l'échelle.

- Mise à l'échelle plus rapide, plus granulaire et moins perturbatrice qu'Aurora Serverless v1 : Aurora Serverless v2 peut procéder à une augmentation/réduction d'échelle plus rapidement. La mise à l'échelle peut modifier la capacité de 0,5 ACU au lieu de doubler ou de réduire de moitié le nombre d'ACU. La mise à l'échelle s'effectue généralement sans interrompre le traitement. La mise à l'échelle n'implique pas d'événement dont vous devez être conscient, comme avec Aurora Serverless v1. La mise à l'échelle peut intervenir lorsque les instructions SQL sont en cours d'exécution et que les transactions sont ouvertes, sans qu'il soit nécessaire d'attendre un point silencieux.

Fonctionnement d’Aurora Serverless v2

La présentation suivante décrit le fonctionnement d’Aurora Serverless v2.

Rubriques

Présentation d’Aurora Serverless v2

Amazon Aurora Serverless v2 convient aux charges de travail hautement variables les plus exigeantes. Par exemple, l’utilisation de votre base de données peut être intensive pendant une

courte période, suivie de longues périodes avec une activité légère, voire pas d'activité. On peut citer, par exemple, les sites Web de vente au détail, de jeux ou de sport proposant des événements promotionnels périodiques, ainsi que les bases de données qui produisent des rapports selon les besoins. On peut également citer les environnements de développement et de test, ainsi que les nouvelles applications où l'utilisation peut rapidement augmenter. Dans de tels cas et bien d'autres, la configuration à l'avance d'une capacité adaptée n'est pas toujours possible avec le modèle provisionné. Cela peut également entraîner des coûts plus élevés en cas de sur-approvisionnement et de capacité non utilisée.

En revanche, les clusters approvisionnés Aurora sont adaptés aux charges de travail stables. Avec les clusters approvisionnés, vous choisissez une classe d'instance de base de données qui possède un volume prédéfini de mémoire, de puissance du processeur, de bande passante d'E/S, etc. Si votre charge de travail change, vous modifiez manuellement la classe d'instance de votre enregistreur et de vos lecteurs. Le modèle approvisionné fonctionne bien lorsque vous pouvez ajuster la capacité avant les modèles de consommation attendus et il est acceptable que de brèves pannes se produisent lorsque vous modifiez la classe d'instance de l'enregistreur et des lecteurs de votre cluster.

Aurora Serverless v2 est entièrement conçu pour prendre en charge les clusters de bases de données sans serveur qui sont instantanément évolutifs. Aurora Serverless v2 est conçu pour assurer le même degré de sécurité et d'isolement que les enregistreurs et les lecteurs approvisionnés. Ces aspects sont essentiels dans les environnements cloud multilocataires sans serveur. Le mécanisme de mise à l'échelle dynamique n'engendre que très peu de frais généraux, ce qui lui permet de réagir rapidement aux modifications de la charge de travail de la base de données. En outre, elle est suffisamment puissante pour répondre à d'importantes augmentations en termes de demande de traitement.

Aurora Serverless v2 vous permet de créer un cluster de bases de données Aurora sans être limité à une capacité de base de données spécifique pour chaque enregistreur et lecteur. Vous spécifiez la plage de capacité minimale et maximale. Aurora met à l'échelle chaque enregistreur ou lecteur Aurora Serverless v2 du cluster dans cette plage de capacité. En utilisant un cluster multi-AZ où chaque enregistreur ou lecteur peut faire l'objet d'une mise à l'échelle dynamique, vous pouvez tirer parti de la mise à l'échelle dynamique et de la haute disponibilité.

Aurora Serverless v2 met automatiquement à l'échelle les ressources de base de données en fonction de vos spécifications minimale et maximale de capacité. La mise à l'échelle est rapide, car la plupart des opérations sur les événements de mise à l'échelle maintiennent l'enregistreur ou le lecteur sur le même hôte. Dans les rares cas où un enregistreur/lecteur Aurora Serverless v2 est

déplacé d'un hôte à un autre, Aurora Serverless v2 gère automatiquement les connexions. Vous n'avez pas besoin de modifier le code d'application cliente de votre base de données ou les chaînes de connexion à la base de données.

Avec Aurora Serverless v2, comme pour les clusters approvisionnés, la capacité de stockage et la capacité de calcul sont séparées. Lorsqu'il est fait référence à la capacité et à la mise à l'échelle Aurora Serverless v2, il s'agit toujours de la capacité de calcul qui augmente ou diminue. Ainsi, votre cluster peut contenir un grand nombre de téraoctets de données, même lorsque la capacité du processeur et de la mémoire fait l'objet d'une réduction d'échelle à de faibles niveaux.

Au lieu d'approvisionner et de gérer les serveurs de base de données, vous spécifiez la capacité de base de données. Pour plus de détails sur la capacité Aurora Serverless v2, consultez Capacité Aurora Serverless v2. La capacité réelle de chaque enregistreur ou lecteur Aurora Serverless v2 varie au fil du temps, en fonction de votre charge de travail. Pour plus de détails sur ce mécanisme, consultez Mise à l'échelle d'Aurora Serverless v2.

> **Important**
>
> Avec Aurora Serverless v1, votre cluster dispose d'une seule mesure de la capacité de calcul qui peut être mise à l'échelle entre les valeurs de capacité minimale et maximale. Avec Aurora Serverless v2, votre cluster peut contenir des lecteurs en plus de l'enregistreur. Chaque enregistreur et lecteur Aurora Serverless v2 peut être mis à l'échelle entre les valeurs de capacité minimale et maximale. Ainsi, la capacité totale de votre cluster Aurora Serverless v2 dépend de la plage de capacité que vous définissez pour votre cluster de bases de données, mais également du nombre d'enregistreurs et de lecteurs dans le cluster. Quelle que soit l'heure, seule la capacité Aurora Serverless v2 qui est activement utilisée dans votre cluster de bases de données Aurora vous est facturée.

Configurations pour les clusters de bases de données Aurora

Pour chacun de vos clusters de bases de données Aurora, vous pouvez choisir n'importe quelle combinaison de capacité Aurora Serverless v2 et/ou de capacité approvisionnée.

Vous pouvez configurer un cluster qui contient la capacité Aurora Serverless v2 et la capacité approvisionnée, appelé cluster à configuration mixte. Supposons par exemple que vous ayez besoin d'une capacité de lecture/écriture supérieure à celle disponible pour un enregistreur Aurora Serverless v2. Dans ce cas, vous pouvez configurer le cluster avec un enregistreur approvisionné très volumineux. Dans ce cas, vous pouvez toujours utiliser Aurora Serverless v2 pour les lecteurs.

Supposons également que la charge de travail d'écriture de votre cluster varie mais que la charge de travail de lecture est stable. Dans ce cas, vous pouvez configurer votre cluster avec un enregistreur Aurora Serverless v2 et un ou plusieurs lecteurs approvisionnés.

Vous pouvez également configurer un cluster de bases de données où l'ensemble de la capacité est géré par Aurora Serverless v2. Pour ce faire, vous pouvez créer un cluster et utiliser Aurora Serverless v2 dès le départ. Vous pouvez également remplacer l'ensemble de la capacité approvisionnée d'un cluster existant par Aurora Serverless v2. Par exemple, certains chemins de mise à niveau des anciennes versions du moteur exigent de commencer avec un enregistreur approvisionné et de le remplacer par un enregistreur Aurora Serverless v2. Pour obtenir les procédures de création d'un cluster de bases de données avec Aurora Serverless v2 ou de basculement d'un cluster de bases de données existant vers Aurora Serverless v2, consultez Création d'un cluster de bases de données Aurora Serverless v2 et Basculement d'un cluster approvisionné vers Aurora Serverless v2.

Si vous n'utilisez pas du tout Aurora Serverless v2 dans un cluster de bases de données, tous les enregistreurs et lecteurs du cluster de bases de données sont approvisionnés. Il s'agit du type de cluster de bases de données le plus ancien et le plus courant que la plupart des utilisateurs connaissent. En fait, avant Aurora Serverless, ce type de cluster de bases de données Aurora ne portait pas de nom spécifique. La capacité approvisionnée est constante. Les frais sont relativement faciles à prévoir. Cependant, vous devez prévoir à l'avance la capacité dont vous avez besoin. Dans certains cas, vos prévisions peuvent être inexactes ou vos besoins en capacité peuvent évoluer. Dans ces cas, votre cluster de bases de données peut devenir sous-approvisionné (plus lent que vous le souhaitez) ou surapprovisionné (plus cher que vous le souhaitez).

Capacité Aurora Serverless v2

L'unité de mesure pour Aurora Serverless v2 est l'unité de capacité Aurora (ACU). La capacité Aurora Serverless v2 n'est pas liée aux classes d'instance de base de données que vous utilisez pour les clusters approvisionnés.

Chaque ACU combine environ 2 gibioctets (Gio) de mémoire, avec une UC et une mise en réseau correspondantes. Vous spécifiez la plage de capacité de la base de données à l'aide de cette unité de mesure. Les métriques `ServerlessDatabaseCapacity` et `ACUUtilization` vous permettent de déterminer le volume de capacité que votre base de données utilise réellement et où se situe cette capacité dans la plage spécifiée.

À un moment donné, chaque enregistreur ou lecteur de base de données Aurora Serverless v2 possède une capacité. La capacité est représentée sous la forme d'un nombre à virgule flottante

représentant les ACU. La capacité augmente ou diminue chaque fois que l'enregistreur ou le lecteur est mis à l'échelle. Cette valeur est mesurée toutes les secondes. Pour chaque cluster de bases de données sur lequel vous prévoyez d'utiliser Aurora Serverless v2, vous définissez une plage de capacité : il s'agit des valeurs de capacité minimale et maximale entre lesquelles chaque enregistreur ou lecteur Aurora Serverless v2 peut être mis à l'échelle. La plage de capacité est identique pour chaque enregistreur ou lecteur Aurora Serverless v2 dans un cluster de bases de données. Chaque enregistreur ou lecteur Aurora Serverless v2 possède sa propre capacité, qui se situe quelque part dans cette plage.

Le tableau suivant affiche les plages de capacité Aurora Serverless v2 et les versions de moteur prises en charge pour Aurora MySQL et Aurora PostgreSQL.

Plage de capacité (ACU)	Versions Aurora MySQL prises en charge	Versions Aurora PostgreSQL prises en charge
0,5 à 128	3.02.0 et versions ultérieures	Versions 13.6 et ultérieures, 14.3 et ultérieures, 15.2 et ultérieures, 16.1 et ultérieures
0,5 à 256	3.06.0 et versions ultérieures	Versions 13.13 et ultérieures, 14.10 et ultérieures, 15.5 et ultérieures, 16.1 et ultérieures
0 à 256	3.08.0 et versions ultérieures	Versions 13.15 et ultérieures, 14.12 et ultérieures, 15.7 et ultérieures, 16.3 et ultérieures

Le tableau suivant présente les versions de plateforme Aurora Serverless v2 avec leurs plages d'ACU et leurs caractéristiques de performance.

Version de plateforme Aurora Serverless v2	Plage d'ACU	Performances
1	0 à 128	Performances de référence
2	0 à 256	Performances de référence

Version de plateforme Aurora Serverless v2	Plage d'ACU	Performances
3	0 à 256	Amélioration des performances de jusqu'à 30 % par rapport à la version de plateforme 2

> **Note**
>
> La plage de mise à l'échelle disponible pour un cluster donné est déterminée à la fois par la version du moteur et par la version de plateforme. Il est possible qu'une version de moteur plus performante fonctionne sur une version de plateforme moins performante, et vice-versa. La plage de mise à l'échelle est déterminée par la version de moteur ou de plateforme la moins performante.

Vous pouvez déterminer la version de plateforme sur laquelle votre cluster s'exécute dans la section Configuration de l'instance de l'API AWS Management Console ou via l'API en consultant la valeur `ServerlessV2PlatformVersion` d'un cluster de bases de données.

La capacité Aurora Serverless v2 minimale que vous pouvez définir est de 0 ACU, pour les versions Aurora Serverless v2 qui prennent en charge la fonctionnalité de pause automatique. Vous pouvez spécifier un nombre plus élevé s'il reste inférieur ou égal à la valeur de capacité maximale. La définition de la capacité minimale sur une faible valeur permet aux clusters de bases de données à faible charge de consommer des ressources de calcul minimales. En même temps, ils restent prêts à accepter des connexions immédiatement et font l'objet d'une augmentation d'échelle lorsqu'ils deviennent occupés.

Nous vous recommandons de définir la capacité minimale sur une valeur qui permet à chaque enregistreur ou lecteur de base de données de conserver l'ensemble de travail de l'application dans le pool de mémoires tampons. De cette façon, le contenu du pool de mémoires tampons n'est pas ignoré pendant les périodes d'inactivité. Pour connaître tous les éléments à prendre en compte lors du choix de la valeur de capacité minimale, consultez Choix de la valeur minimale de capacité Aurora Serverless v2 pour un cluster. Pour connaître tous les éléments à prendre en compte lors du choix de la valeur de capacité maximale, consultez Choix de la valeur maximale de capacité Aurora Serverless v2 pour un cluster.

Selon la façon dont vous configurez les lecteurs dans un déploiement multi-AZ, leur capacité peut être liée à la capacité de l'enregistreur ou être indépendante. Pour savoir comment procéder, consultez Mise à l'échelle d'Aurora Serverless v2.

La surveillance d'Aurora Serverless v2 implique de mesurer les valeurs de capacité de l'enregistreur et des lecteurs de votre cluster de bases de données au fil du temps. Si votre base de données ne fait pas l'objet d'une réduction d'échelle à la capacité minimale, vous pouvez prendre des mesures telles que l'ajustement de la capacité minimale et l'optimisation de votre application de base de données. Si votre base de données atteint systématiquement sa capacité maximale, vous pouvez prendre des mesures telles que l'augmentation de la capacité maximale. Vous pouvez également optimiser votre application de base de données et répartir la charge de requête sur un plus grand nombre de lecteurs.

Les frais associés à la capacité Aurora Serverless v2 sont mesurés en termes d'heures ACU. Pour plus d'informations sur la procédure de calcul des frais associés à Aurora Serverless v2, consultez la page de tarification Aurora.

Supposons que le nombre total d'enregistreurs et de lecteurs de votre cluster soit de *n*. Dans ce cas, le cluster consomme environ *n* x *minimum ACUs* lorsque vous n'exécutez aucune opération de base de données. Il est possible qu'Aurora exécute des opérations de surveillance ou de maintenance qui entraînent une faible charge. Ce cluster ne consomme pas plus de *n* x *maximum ACUs* lorsque la base de données est exécutée à pleine capacité.

Pour plus de détails sur le choix des valeurs d'ACU minimale et maximale appropriées, consultez Choix de la plage de capacité Aurora Serverless v2 pour un cluster Aurora. Les valeurs d'ACU minimale et maximale que vous spécifiez affectent également la façon dont certains paramètres de configuration Aurora fonctionnent pour Aurora Serverless v2. Pour plus de détails sur l'interaction entre la plage de capacité et les paramètres de configuration, consultez Utilisation des groupes de paramètres pour Aurora Serverless v2.

Mise à l'échelle d'Aurora Serverless v2

Pour chaque enregistreur ou lecteur Aurora Serverless v2, Aurora suit en permanence l'utilisation des ressources telles que le processeur, la mémoire et le réseau. L'ensemble de ces mesures est appelé la charge. La charge inclut les opérations de base de données effectuées par votre application. Elle inclut également le traitement en arrière-plan pour le serveur de base de données et les tâches administratives Aurora. Lorsque la capacité est limitée par l'un de ces facteurs, Aurora Serverless v2 fait l'objet d'une augmentation d'échelle. Aurora Serverless v2 fait également l'objet

d'une augmentation d'échelle lorsqu'il détecte des problèmes de performances qu'il peut résoudre en procédant ainsi. Vous pouvez surveiller l'utilisation des ressources et son impact sur la mise à l'échelle d'Aurora Serverless v2 en utilisant les procédures décrites aux rubriques Métriques Amazon CloudWatch importantes pour Aurora Serverless v2 et Surveillance des performances d'Aurora Serverless v2 avec Performance Insights.

La charge peut varier selon l'enregistreur et les lecteurs de votre cluster de bases de données. L'enregistreur gère l'ensemble des instructions de langage de définition de données (DDL), telles que `CREATE TABLE`, `ALTER TABLE` et `DROP TABLE`. Il gère également l'ensemble des instructions de langage de manipulation de données (DML), telles que `INSERT` et `UPDATE`. Les lecteurs peuvent traiter les instructions en lecture seule, telles que les requêtes `SELECT`.

La mise à l'échelle est l'opération qui augmente ou diminue la capacité Aurora Serverless v2 de votre base de données. Avec Aurora Serverless v2, chaque enregistreur et lecteur possède sa propre valeur de capacité actuelle, mesurée en ACU. Aurora Serverless v2 met à l'échelle un enregistreur ou un lecteur jusqu'à une capacité supérieure lorsque sa capacité actuelle est trop faible pour gérer la charge. Il procède à une réduction d'échelle pour l'enregistreur ou le lecteur jusqu'à une capacité inférieure lorsque sa capacité actuelle est supérieure à celle nécessaire.

Contrairement à Aurora Serverless v1, dont la mise à l'échelle double la capacité chaque fois que le cluster de bases de données atteint un certain seuil, Aurora Serverless v2 peut augmenter la capacité de manière incrémentielle. Lorsque la demande de charge de travail commence à atteindre la capacité de base de données actuelle d'un enregistreur ou d'un lecteur, Aurora Serverless v2 augmente le nombre d'ACU de cet enregistreur ou lecteur. Aurora Serverless v2 met à l'échelle la capacité par incréments nécessaires pour optimiser les performances des ressources consommées. La mise à l'échelle se fait par incréments de 0,5 ACU. Plus la capacité actuelle est grande, plus l'incrément de mise à l'échelle est important et plus la mise à l'échelle peut être rapide.

Etant donné que la mise à l'échelle Aurora Serverless v2 est si fréquente, granulaire et non perturbatrice, elle ne provoque pas d'événements ponctuels dans AWS Management Console comme c'est le cas pour Aurora Serverless v1. Vous pouvez plutôt mesurer les métriques Amazon CloudWatch telles que `ServerlessDatabaseCapacity` et `ACUUtilization` et suivre leurs valeurs minimales, maximales et moyennes au fil du temps. Pour en savoir plus sur les métriques Aurora, consultez Surveillance des métriques d'un cluster de bases de données Amazon Aurora. Pour obtenir des conseils sur la surveillance d'Aurora Serverless v2, consultez Métriques Amazon CloudWatch importantes pour Aurora Serverless v2.

Vous pouvez choisir de mettre à l'échelle un lecteur en même temps que l'enregistreur associé, ou indépendamment de l'enregistreur. Pour ce faire, spécifiez le niveau de promotion pour ce lecteur.

- Les lecteurs aux niveaux de promotion 0 et 1 sont mis à l'échelle en même temps que l'enregistreur. Ce comportement de mise à l'échelle rend les lecteurs aux niveaux de priorité 0 et 1 parfaitement adaptés à la disponibilité. En effet, ils sont toujours dimensionnés à la bonne capacité pour prendre en charge la charge de travail de l'enregistreur en cas de basculement.
- Les lecteurs aux niveaux de promotion 2 à 15 sont mis à l'échelle indépendamment de l'enregistreur. Chaque lecteur reste dans les valeurs d'ACU minimale et maximale que vous avez spécifiées pour votre cluster. Lorsqu'un lecteur est mis à l'échelle indépendamment de la base de données d'enregistreur associée, il peut devenir inactif et faire l'objet d'une réduction d'échelle pendant que l'enregistreur continue à traiter un volume élevé de transactions. Il est toujours disponible en tant que cible de basculement, si aucun autre lecteur n'est disponible aux niveaux de promotion inférieurs. Toutefois, s'il est promu en enregistreur, il devra peut-être faire l'objet d'une augmentation d'échelle pour gérer l'ensemble de la charge de travail de l'enregistreur.

Pour plus de détails sur les niveaux de promotion, consultez Choix du niveau de promotion pour un lecteur Aurora Serverless v2.

Les notions Aurora Serverless v1 de points de mise à l'échelle et de délais d'expiration associés ne s'appliquent pas dans Aurora Serverless v2. La mise à l'échelle d'Aurora Serverless v2 peut intervenir lorsque les connexions à la base de données sont ouvertes, alors que les transactions SQL sont en cours de traitement, que les tables sont verrouillées et que les tables temporaires sont en cours d'utilisation. Aurora Serverless v2 n'attend pas de point silencieux pour commencer la mise à l'échelle. La mise à l'échelle ne perturbe pas les opérations de base de données en cours.

Si votre charge de travail exige une capacité de lecture supérieure à celle disponible avec un seul enregistreur et un seul lecteur, vous pouvez ajouter plusieurs lecteurs Aurora Serverless v2 au cluster. Chaque lecteur Aurora Serverless v2 peut être mis à l'échelle dans la plage de valeurs d'ACU minimale et maximale que vous avez spécifiée pour votre cluster de bases de données. Vous pouvez utiliser le point de terminaison du lecteur du cluster pour diriger les séances en lecture seule vers les lecteurs et réduire la charge sur l'enregistreur.

Les valeurs d'ACU minimale et maximale du cluster ont également un impact sur le fait qu'Aurora Serverless v2 effectue une mise à l'échelle ou non, et sur la rapidité de la mise à l'échelle une fois qu'elle démarre. Cela dépend également du fait qu'un lecteur soit configuré pour être mis à l'échelle en même temps que l'enregistreur ou indépendamment de celui-ci. Pour plus de détails sur les facteurs qui affectent la mise à l'échelle d'Aurora Serverless v2, consultez Performances et mise à l'échelle pour Aurora Serverless v2.

Réduction verticale à zéro

Dans les versions Aurora MySQL et Aurora PostgreSQL, les enregistreurs et les lecteurs Aurora Serverless v2 peuvent faire l'objet d'une réduction verticale rabaissant le nombre d'ACU à 0. C'est ce que nous appelons la pause et la reprise automatiques. Vous pouvez choisir d'autoriser ce comportement en spécifiant une valeur nulle ou différente de zéro pour la capacité minimale. Vous pouvez également choisir le temps d'attente avant qu'une instance Aurora Serverless v2 ne soit mise en pause. Pour en savoir plus sur les versions dotées de cette fonctionnalité, consultez Capacité Aurora Serverless v2. Pour en savoir plus sur la façon de l'utiliser efficacement, consultez Réduction verticale à zéro ACU avec pause et reprise automatiques pour Aurora Serverless v2.

Dans les versions Aurora MySQL et Aurora PostgreSQL, les enregistreurs et les lecteurs Aurora Serverless v2 inactifs peuvent faire l'objet d'une réduction d'échelle atteignant la valeur d'ACU minimale que vous avez spécifiée pour le cluster, sans toutefois atteindre 0 ACU. Dans ce cas, l'option zéro ACU n'est pas disponible lorsque vous définissez la plage de capacité. Ce comportement est différent d'Aurora Serverless v1, qui peut faire une pause après une période d'inactivité, mais qui prend ensuite un certain temps pour reprendre lorsque vous ouvrez une nouvelle connexion.

Lorsque votre cluster de bases de données avec la capacité Aurora Serverless v2 n'est pas nécessaire pendant un certain temps, vous pouvez arrêter et démarrer le cluster entier comme pour les clusters de bases de données provisionnés. Cette technique est particulièrement adaptée aux systèmes de développement et de test, où ils peuvent ne pas être nécessaires pendant de nombreuses heures d'affilée et où la rapidité de reprise du cluster n'est pas cruciale. La fonctionnalité d'arrêt et de démarrage de cluster est disponible pour toutes les versions Aurora Serverless v2. Pour plus d'informations sur cette fonctionnalité, consultez Arrêt et démarrage d'un cluster de bases de données Amazon Aurora.

Aurora Serverless v2 et haute disponibilité

Pour établir une haute disponibilité pour un cluster de bases de données Aurora, il convient d'en faire un cluster de bases de données multi-AZ. Un cluster de bases de données Aurora multi-AZ possède une capacité de calcul disponible à tout moment dans plusieurs zones de disponibilité. Cette configuration permet de maintenir votre base de données opérationnelle même en cas de panne importante. Aurora effectue un basculement automatique en cas de problème affectant l'enregistreur ou même l'intégralité de la zone de disponibilité. Avec Aurora Serverless v2, vous pouvez définir que la capacité de calcul de secours fasse l'objet d'une augmentation et d'une réduction d'échelle en même temps que la capacité de l'enregistreur. De cette façon, la capacité de

calcul de la deuxième zone de disponibilité est prête à prendre en charge la charge de travail actuelle à tout moment. Dans le même temps, la capacité de calcul de toutes les zones de disponibilité peut faire l'objet d'une réduction d'échelle lorsque la base de données est inactive. Pour en savoir plus sur le fonctionnement d'Aurora avec Régions AWS et les zones de disponibilité, consultez Haute disponibilité pour les instances de base de données Aurora.

La fonctionnalité multi-AZ d'Aurora Serverless v2 utilise des lecteurs en plus de l'enregistreur. La prise en charge des lecteurs est une nouveauté d'Aurora Serverless v2 par rapport à Aurora Serverless v1. Vous pouvez ajouter jusqu'à 15 lecteurs Aurora Serverless v2 répartis sur 3 zones de disponibilité à un cluster de bases de données Aurora.

Pour les applications stratégiques qui doivent rester disponibles même en cas de problème affectant l'ensemble de votre cluster ou de la région AWS, vous pouvez configurer une base de données globale Aurora. Vous pouvez utiliser la capacité Aurora Serverless v2 dans les clusters secondaires afin qu'ils soient prêts à prendre le relais pendant la reprise après sinistre. Ils peuvent également faire l'objet d'une réduction d'échelle lorsque la base de données n'est pas occupée. Pour plus de détails sur les bases de données globales Aurora, consultez Utilisation d'Amazon Aurora Global Database.

Aurora Serverless v2 fonctionne comme en mode approvisionné pour le basculement et d'autres fonctionnalités de haute disponibilité. Pour plus d'informations, consultez Haute disponibilité pour Amazon Aurora.

Supposons que vous souhaitiez garantir la disponibilité maximale de votre cluster Aurora Serverless v2. Vous pouvez créer un lecteur en plus de l'enregistreur. Si vous affectez le lecteur au niveau de promotion 0 ou 1, la mise à l'échelle appliquée à l'enregistreur est également appliquée au lecteur. De cette façon, un lecteur ayant une capacité identique est toujours prêt à prendre le relais de l'enregistreur en cas de basculement.

Supposons que vous souhaitiez exécuter des rapports trimestriels pour votre entreprise pendant que votre cluster continue de traiter les transactions. Si vous ajoutez un lecteur Aurora Serverless v2 au cluster et lui attribuez un niveau de promotion compris entre 2 et 15, vous pouvez vous connecter directement à ce lecteur pour exécuter les rapports. Selon la quantité de mémoire et de processeur exigée par les requêtes de rapport, ce lecteur peut faire l'objet d'une augmentation d'échelle en fonction de la charge de travail. Il peut ensuite faire l'objet d'une réduction d'échelle une fois les rapports terminés.

Aurora Serverless v2 et stockage

Le stockage de chaque cluster de bases de données Aurora se compose de six copies de toutes vos données, réparties sur trois zones de disponibilité. Cette réplication de données intégrée s'applique, que votre cluster de bases de données comporte ou non des lecteurs en plus de l'enregistreur. De cette façon, vos données sont sécurisées, même contre les problèmes qui affectent la capacité de calcul du cluster.

Le stockage Aurora Serverless v2 présente les mêmes caractéristiques de fiabilité et de durabilité que celles décrites à la rubrique Stockage Amazon Aurora. En effet, le stockage des clusters de bases de données Aurora fonctionne de la même manière, que la capacité de calcul utilise Aurora Serverless v2 ou le mode approvisionné.

Paramètres de configuration des clusters Aurora

Vous pouvez ajuster les mêmes paramètres de configuration de cluster et de base de données pour les clusters avec la capacité Aurora Serverless v2 que pour les clusters de bases de données approvisionnés. Cependant, certains paramètres de capacité sont traités différemment pour Aurora Serverless v2. Dans un cluster à configuration mixte, les valeurs que vous spécifiez pour ces paramètres de capacité s'appliquent toujours à tous les enregistreurs et lecteurs approvisionnés.

Presque tous les paramètres fonctionnent de la même manière pour les enregistreurs et les lecteurs Aurora Serverless v2 que pour ceux en mode approvisionné. Les exceptions concernent certains paramètres ajustés automatiquement par Aurora pendant la mise à l'échelle, et certains paramètres qu'Aurora conserve à des valeurs fixes qui dépendent de la valeur de la capacité maximale.

Par exemple, la quantité de mémoire réservée au cache de la mémoire tampon augmente à mesure de l'augmentation d'échelle d'un enregistreur ou d'un lecteur, et diminue à mesure de sa réduction d'échelle. De cette façon, la mémoire peut être libérée lorsque votre base de données n'est pas occupée. À l'inverse, Aurora définit automatiquement le nombre maximal de connexions sur une valeur appropriée en fonction de la valeur de la capacité maximale. De cette façon, les connexions actives ne sont pas interrompues si la charge diminue et si Aurora Serverless v2 fait l'objet d'une réduction d'échelle. Pour obtenir des informations sur la façon dont Aurora Serverless v2 gère certains paramètres, consultez Utilisation des groupes de paramètres pour Aurora Serverless v2.

Exigences et limites relatives à Aurora Serverless v2

Lorsque vous créez un cluster dans lequel vous prévoyez d'utiliser des instances de base de données Aurora Serverless v2, prêtez une attention particulière aux exigences et limites suivantes.

Rubriques

Disponibilité des régions et des versions

La disponibilité et la prise en charge des fonctions varient selon les versions spécifiques de chaque moteur de base de données Aurora, et selon les Régions AWS. Pour en savoir plus sur les versions et la disponibilité des régions avec Aurora et Aurora Serverless v2, consultez Régions et moteurs de base de données Aurora pris en charge pour Aurora Serverless v2.

L'exemple suivant illustre les commandes AWS CLI qui permettent de confirmer les valeurs exactes du moteur de base de données que vous pouvez utiliser avec Aurora Serverless v2 pour une Région AWS spécifique. Le paramètre `--db-instance-class` pour Aurora Serverless v2 est toujours `db.serverless`. Le paramètre `--engine` peut être `aurora-mysql` ou `aurora-postgresql`. Remplacez les valeurs `--region` et `--engine` appropriées pour confirmer les valeurs `--engine-version` que vous pouvez utiliser. Si la commande ne génère aucune sortie, Aurora Serverless v2 n'est pas disponible pour cette combinaison de Région AWS et de moteur de base de données.

```
aws rds describe-orderable-db-instance-options --engine aurora-mysql --db-instance-
class db.serverless \
  --region my_region --query 'OrderableDBInstanceOptions[].[EngineVersion]' --output
 text

aws rds describe-orderable-db-instance-options --engine aurora-postgresql --db-
instance-class db.serverless \
  --region my_region --query 'OrderableDBInstanceOptions[].[EngineVersion]' --output
 text
```

Les clusters qui utilisent Aurora Serverless v2 doivent avoir une plage de capacité spécifiée

Un cluster Aurora doit avoir un attribut `ServerlessV2ScalingConfiguration` avant de pouvoir ajouter des instances de base de données qui utilisent la classe d'instance de base de données

`db.serverless`. Cet attribut spécifie la plage de capacité La capacité d'Aurora Serverless v2 est comprise entre 0 unité de capacité Aurora (ACU) minimum et un maximum de 256 ACU, par incréments de 0,5 ACU. La valeur minimale autorisée dépend de la version d'Aurora. Chaque ACU fournit l'équivalent d'environ 2 gibioctets (Gio) de RAM, ainsi que d'UC et de mise en réseau associées. Pour plus de détails sur la façon dont Aurora Serverless v2 utilise les paramètres de plage de capacité, consultez Fonctionnement d'Aurora Serverless v2.

Pour connaître les plages de capacité autorisées pour les différentes versions du moteur de base de données et versions de plateforme, consultez Capacité Aurora Serverless v2. La plage de mise à l'échelle disponible pour un cluster donné dépend à la fois de la version du moteur et du matériel (version de plateforme).

Vous pouvez spécifier les valeurs minimales et maximales d'ACU dans AWS Management Console lorsque vous créez un cluster et une instance de base de données Aurora Serverless v2 associée. Vous pouvez également spécifier l'option `--serverless-v2-scaling-configuration` dans AWS CLI. Sinon, vous pouvez spécifier le paramètre `ServerlessV2ScalingConfiguration` avec l'API Amazon RDS. Vous pouvez spécifier cet attribut lorsque vous créez un cluster ou modifiez un cluster existant. Pour connaître les procédures de définition de la plage de capacité, consultez Définition de la plage de capacité Aurora Serverless v2 d'un cluster. Pour obtenir la procédure détaillée de sélection des valeurs de capacité minimale et maximale et pour savoir comment ces paramètres affectent certains paramètres de base de données, consultez Choix de la plage de capacité Aurora Serverless v2 pour un cluster Aurora.

Certaines fonctionnalités approvisionnées ne sont pas prises en charge dans Aurora Serverless v2

Les fonctionnalités suivantes des instances de base de données approvisionnées Aurora ne sont actuellement pas disponibles pour Amazon Aurora Serverless v2 :

- Flux d'activité de base de données (DAS).
- Gestion du cache de clusters pour Aurora PostgreSQL. Le paramètre de configuration `apg_ccm_enabled` ne s'applique pas aux instances de base de données Aurora Serverless v2.

Certaines fonctionnalités Aurora fonctionnent avec Aurora Serverless v2, mais cela peut poser problème si votre plage de capacité est inférieure à celle nécessaire pour les besoins en mémoire de ces fonctionnalités avec votre charge de travail spécifique. Dans ce cas, votre base de données risque de ne pas fonctionner aussi bien que d'habitude ou de rencontrer des erreurs de mémoire

insuffisante. Pour obtenir des recommandations sur la définition de la plage de capacité appropriée, consultez Choix de la plage de capacité Aurora Serverless v2 pour un cluster Aurora. Pour obtenir des informations de dépannage si votre base de données rencontre des erreurs de mémoire insuffisante dues à une mauvaise configuration de la plage de capacité, consultez Éviter les erreurs de mémoire insuffisante.

Aurora Auto Scaling n'est pas pris en charge. Ce type de mise à l'échelle ajoute de nouveaux lecteurs pour gérer une charge de travail supplémentaire exigeante en lecture, basée sur l'utilisation du processeur. Toutefois, la mise à l'échelle basée sur l'utilisation du processeur n'est pas significative pour Aurora Serverless v2. En guise d'alternative, vous pouvez créer des instances de base de données de lecteur Aurora Serverless v2 à l'avance et conserver leur réduction d'échelle à faible capacité. Il s'agit d'une méthode plus rapide et moins perturbatrice pour mettre à l'échelle la capacité de lecture d'un cluster plutôt que d'ajouter dynamiquement de nouvelles instances de base de données.

Certains aspects d'Aurora Serverless v2 sont différents d'Aurora Serverless v1

Si vous êtes un utilisateur d'Aurora Serverless v1 et que vous utilisez Aurora Serverless v2 pour la première fois, consultez les différences entre les exigences relatives à Aurora Serverless v2 et Aurora Serverless v1 pour comprendre en quoi les exigences sont différentes entre Aurora Serverless v1 et Aurora Serverless v2.

Création d'un cluster de bases de données qui utilise Aurora Serverless v2

Pour créer un cluster Aurora dans lequel vous pouvez ajouter des instances de base de données Aurora Serverless v2, suivez la même procédure que la rubrique Création d'un cluster de bases de données Amazon Aurora. Avec Aurora Serverless v2, vos clusters sont interchangeables avec les clusters approvisionnés. Vos clusters peuvent contenir des instances de base de données qui utilisent Aurora Serverless v2 et d'autres instances de base de données qui sont approvisionnées.

Rubriques

- Paramètres pour les clusters de bases de données Aurora Serverless v2
- Création d'un cluster de bases de données Aurora Serverless v2
- Création d'une instance de base de données d'enregistreur Aurora Serverless v2

Paramètres pour les clusters de bases de données Aurora Serverless v2

Assurez-vous que les paramètres initiaux du cluster répondent aux exigences répertoriées dans Exigences et limites relatives à Aurora Serverless v2. Spécifiez les paramètres suivants pour vous assurer que vous pouvez ajouter des instances de base de données Aurora Serverless v2 au cluster :

Région AWS

Créez le cluster dans une Région AWS où les instances de base de données Aurora Serverless v2 sont disponibles. Pour plus de détails sur les régions disponibles, consultez Régions et moteurs de base de données Aurora pris en charge pour Aurora Serverless v2.

Version du moteur de base de données

Choisissez une version de moteur compatible avec Aurora Serverless v2. Pour plus d'informations sur la version requise pour Aurora Serverless v2, consultez Exigences et limites relatives à Aurora Serverless v2.

Classe d'instance de base de données

Si vous créez un cluster à l'aide d'AWS Management Console, vous choisissez également la classe d'instance de base de données pour l'instance de base de données du dispositif d'écriture. Choisissez la classe d'instance de base de données Sans serveur. Lorsque vous choisissez cette classe d'instance de base de données, vous spécifiez également la plage de capacité de l'instance de base de données d'enregistreur. Cette même plage de capacité s'applique à toutes les autres instances de base de données Aurora Serverless v2 que vous ajoutez à ce cluster.

Si l'option Sans serveur ne s'affiche pas pour la classe d'instance de base de données, assurez-vous d'avoir choisi une version de moteur de base de données prise en charge par Régions et moteurs de base de données Aurora pris en charge pour Aurora Serverless v2.

Lorsque vous utilisez AWS CLI ou l'API Amazon RDS, le paramètre que vous spécifiez pour la classe d'instance de base de données est `db.serverless`.

Plage de capacité

Renseignez les valeurs minimale et maximale d'unité de capacité Aurora (ACU) qui s'appliquent à toutes les instances de base de données du cluster. Cette option est disponible sur les pages de console Créer un cluster et Ajouter un lecteur lorsque vous choisissez Sans serveur pour la classe d'instance de base de données.

Pour connaître les plages de capacité autorisées pour les différentes versions du moteur de base de données, consultez Capacité Aurora Serverless v2.

Si les champs de valeur minimale et maximale d'ACU ne s'affichent pas, assurez-vous d'avoir choisi la classe d'instance de base de données Sans serveur pour l'instance de base de données d'enregistreur.

Si vous créez initialement le cluster avec une instance de base de données approvisionnée, vous ne spécifiez pas les nombres minimal et maximal d'ACU. Dans ce cas, vous pouvez modifier le cluster par la suite pour ajouter ce paramètre. Vous pouvez également ajouter une instance de base de données de lecteur Aurora Serverless v2 au cluster. Vous spécifiez la plage de capacité dans le cadre de ce processus.

Tant que vous ne spécifiez pas la plage de capacité de votre cluster, vous ne pouvez ajouter aucune instance de base de données Aurora Serverless v2 au cluster à l'aide d'AWS CLI ou de l'API RDS. Si vous essayez d'ajouter une instance de base de données Aurora Serverless v2, vous obtenez une erreur. Dans les procédures d'AWS CLI ou de l'API RDS, la plage de capacité est représentée par l'attribut `ServerlessV2ScalingConfiguration`.

Pour les clusters contenant plusieurs instances de base de données de lecteur, la priorité de basculement de chaque instance de base de données de lecteur Aurora Serverless v2 joue un rôle important dans l'augmentation et la réduction d'échelle de cette instance de base de données. Vous ne pouvez pas spécifier la priorité lors de la création initiale du cluster. Gardez cette propriété à l'esprit lorsque vous ajoutez une seconde instance de base de données de lecteur à votre cluster. Pour plus d'informations, consultez Choix du niveau de promotion pour un lecteur Aurora Serverless v2.

Création d'un cluster de bases de données Aurora Serverless v2

Vous pouvez utiliser la AWS Management Console, l'AWS CLI ou l'API RDS pour créer un cluster de bases de données Aurora Serverless v2.

Console

Pour créer un cluster avec un enregistreur Aurora Serverless v2

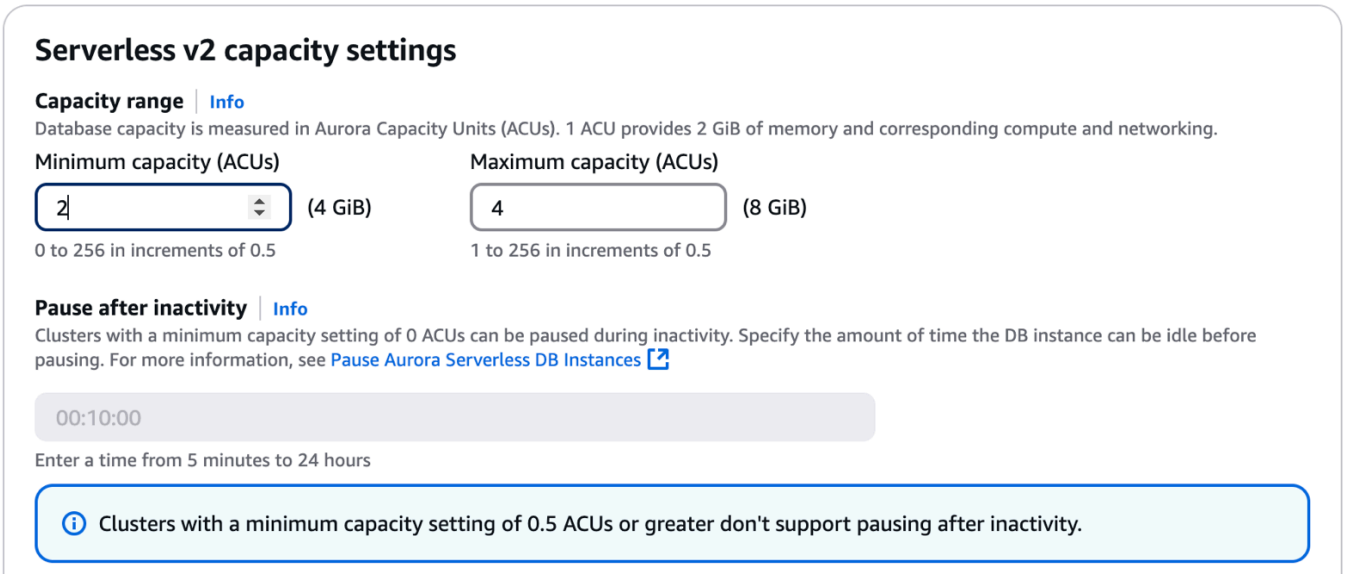
1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Choisissez Create database (Créer une base de données). Sur la page qui s'affiche, choisissez les options suivantes :

- Pour Type de moteur, choisissez Aurora (compatible MySQL) ou Aurora (compatible PostgreSQL).
- Dans Version, choisissez l'une des versions prises en charge pour Régions et moteurs de base de données Aurora pris en charge pour Aurora Serverless v2.

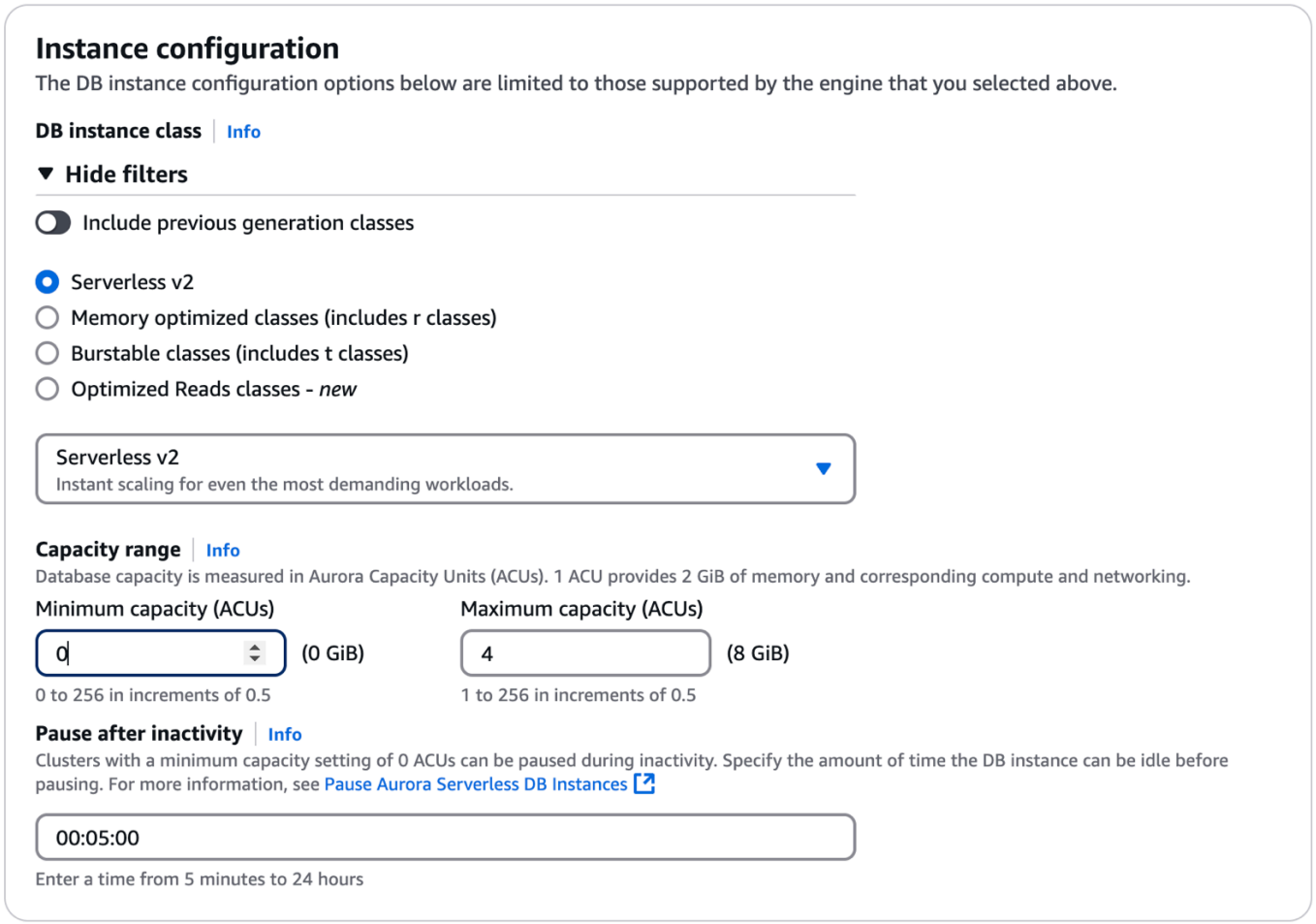
4. Pour Classe d'instance de base de données, choisissez Sans serveur v2.
5. Pour Paramètres de capacité, vous pouvez accepter la plage par défaut. Vous pouvez également choisir d'autres valeurs pour les unités de capacité minimales ou maximales. Vous pouvez choisir entre 0 ACU minimum et 256 ACU maximum, par incréments de 0,5 ACU.

   Pour plus d'informations sur les unités de capacité Aurora Serverless v2, consultez Capacité Aurora Serverless v2 et Performances et mise à l'échelle pour Aurora Serverless v2.

   Selon le moteur et la version que vous choisissez, la limite supérieure peut être de 128 ACU, la limite inférieure de 0,5 ACU, ou les deux. Pour plus de détails sur la limite pour chaque combinaison de moteur et de version Aurora, consultez Capacité Aurora Serverless v2.



   Le choix d'une capacité minimale de 0 ACU active la fonctionnalité Aurora Serverless v2 de pause et de reprise automatiques. Dans ce cas, vous pouvez également choisir la durée d'attente des instances de base de données Aurora Serverless v2 sans connexion à la base de données avant de se mettre en pause automatiquement. Pour plus d'informations sur la fonctionnalité de pause et de reprise automatiques, consultez Réduction verticale à zéro ACU avec pause et reprise automatiques pour Aurora Serverless v2.



6. Choisissez n'importe quel autre paramètre de cluster de bases de données, comme décrit dans Paramètres pour les clusters de bases de données Aurora.
7. Choisissez Créer une base de données pour créer votre cluster de bases de données Aurora avec une instance de base de données Aurora Serverless v2 en tant qu'instance d'enregistreur, également connue sous le nom d'instance de base de données principale.

Interface de ligne de commande (CLI)

Pour créer un cluster de bases de données compatible avec les instances de base de données Aurora Serverless v2 à l'aide d'AWS CLI, suivez la procédure de la rubrique Création d'un cluster de bases de données Amazon Aurora pour l'interface de ligne de commande. Incluez les paramètres suivants dans votre commande `create-db-cluster` :

- --region *`région_AWS_où_instances_Aurora Serverless v2_sont_disponibles`*
- --engine-version *`version_moteur_compatible_serverless_v2`*

- --serverless-v2-scaling-configuration MinCapacity=*capacité_minimale*,MaxCapacity=*capacité_maximale*

L'exemple suivant illustre la création d'un cluster de bases de données Aurora Serverless v2.

```
aws rds create-db-cluster \
    --db-cluster-identifier my-serverless-v2-cluster \
    --region eu-central-1 \
    --engine aurora-mysql \
    --engine-version 8.0.mysql_aurora.3.04.1 \
    --serverless-v2-scaling-configuration MinCapacity=1,MaxCapacity=4 \
    --master-username myuser \
    --manage-master-user-password
```

> **Note**
>
> Lorsque vous créez un cluster de bases de données Aurora Serverless v2 à l'aide de AWS CLI, le mode moteur apparaît dans la sortie sous la forme de `provisioned` plutôt que de `serverless`. Le mode moteur `serverless` fait référence à Aurora Serverless v1.

Cet exemple spécifie l'option `--manage-master-user-password` permettant de générer le mot de passe administratif et de le gérer dans Secrets Manager. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager. Vous pouvez également utiliser l'option `--master-password` pour spécifier et gérer vous-même le mot de passe.

Pour plus d'informations sur la version requise pour Aurora Serverless v2, consultez Exigences et limites relatives à Aurora Serverless v2. Pour plus d'informations sur les nombres autorisés pour la plage de capacité et ce que représentent ces nombres, consultez Capacité Aurora Serverless v2 et Performances et mise à l'échelle pour Aurora Serverless v2.

Pour vérifier si un cluster existant possède les paramètres de capacité spécifiés, vérifiez la sortie de la commande `describe-db-clusters` pour l'attribut `ServerlessV2ScalingConfiguration`. Cet attribut ressemble à ce qui suit.

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": 1.5,
    "MaxCapacity": 24.0
}
```

> **Tip**
>
> Si vous ne spécifiez pas les nombres minimal et maximal d'ACU lors de la création du cluster, vous pouvez utiliser la commande `modify-db-cluster` pour ajouter ce paramètre par la suite. Tant que vous ne le faites pas, vous ne pouvez ajouter aucune instance de base de données Aurora Serverless v2 au cluster. Si vous essayez d'ajouter une instance de base de données `db.serverless`, vous obtenez une erreur.

« Hello, World! »

Pour créer un cluster de bases de données compatible avec les instances de base de données Aurora Serverless v2 à l'aide de l'API RDS, suivez la procédure de la rubrique Création d'un cluster de bases de données Amazon Aurora pour l'API. Sélectionnez les paramètres suivants. Vérifiez que votre opération `CreateDBCluster` inclut les paramètres suivants :

```
EngineVersion serverless_v2_compatible_engine_version
ServerlessV2ScalingConfiguration with MinCapacity=minimum_capacity and
 MaxCapacity=maximum_capacity
```

Pour plus d'informations sur la version requise pour Aurora Serverless v2, consultez Exigences et limites relatives à Aurora Serverless v2. Pour plus d'informations sur les nombres autorisés pour la plage de capacité et ce que représentent ces nombres, consultez Capacité Aurora Serverless v2 et Performances et mise à l'échelle pour Aurora Serverless v2.

Pour vérifier si un cluster existant possède les paramètres de capacité spécifiés, vérifiez la sortie de l'opération `DescribeDBClusters` pour l'attribut `ServerlessV2ScalingConfiguration`. Cet attribut ressemble à ce qui suit.

```
"ServerlessV2ScalingConfiguration": {
    "MinCapacity": 1.5,
    "MaxCapacity": 24.0
}
```

> **Tip**
>
> Si vous ne spécifiez pas les nombres minimal et maximal d'ACU lors de la création du cluster, vous pouvez utiliser l'opération `ModifyDBCluster` pour ajouter ce paramètre par la suite.

Tant que vous ne le faites pas, vous ne pouvez ajouter aucune instance de base de données Aurora Serverless v2 au cluster. Si vous essayez d'ajouter une instance de base de données `db.serverless`, vous obtenez une erreur.

Création d'une instance de base de données d'enregistreur Aurora Serverless v2

Bien que vous spécifiiez la plage de capacité Aurora Serverless v2 lorsque vous créez un cluster Aurora, vous pouvez choisir d'utiliser Aurora Serverless v2 ou provisionné pour chaque instance de base de données du cluster. Pour commencer à utiliser Aurora Serverless v2 immédiatement dans votre cluster de bases de données, ajoutez une instance de base de données d'enregistreur qui utilise la classe d'instance `db.serverless`. Dans la console, vous faites généralement ce choix dans le cadre du processus de création du cluster de bases de données. Par conséquent, cette procédure s'applique principalement lorsque vous effectuez la configuration via l'interface de ligne de commande.

Console

Lorsque vous créez un cluster de bases de données à l'aide de la AWS Management Console, vous spécifiez également les propriétés de l'instance de base de données d'enregistreur. Pour faire en sorte que l'instance de base de données d'enregistreur utilise Aurora Serverless v2, choisissez la classe d'instance de base de données Sans serveur.

Vous définissez ensuite la plage de capacité du cluster en spécifiant les valeurs minimale et maximale d'unité de capacité Aurora (ACU). Ces mêmes valeurs minimale et maximale s'appliquent à chaque instance de base de données Aurora Serverless v2 du cluster. Pour en savoir plus sur cette procédure et sur l'importance des valeurs de capacité minimale et maximale, consultez Création d'un cluster de bases de données Aurora Serverless v2.

Si vous ne créez pas d'instance de base de données Aurora Serverless v2 lorsque vous créez le cluster pour la première fois, vous pouvez ajouter une ou plusieurs instances de base de données Aurora Serverless v2 ultérieurement. Pour ce faire, suivez les procédures des rubriques Ajout d'un lecteur Aurora Serverless v2 et Conversion d'un lecteur ou d'un enregistreur approvisionné en Aurora Serverless v2. Vous spécifiez la plage de capacité au moment où vous ajoutez la première instance de base de données Aurora Serverless v2 au cluster. Vous pouvez modifier la plage de capacité ultérieurement en suivant la procédure de la rubrique Définition de la plage de capacité Aurora Serverless v2 d'un cluster.

Interface de ligne de commande (CLI)

Lorsque vous créez un cluster de bases de données Aurora Serverless v2 à l'aide d'AWS CLI, vous ajoutez explicitement l'instance de base de données d'enregistreur à l'aide de la commande create-db-instance. Incluez le paramètre suivant :

- `--db-instance-class db.serverless`

L'exemple suivant illustre la création d'une instance de base de données d'enregistreur Aurora Serverless v2.

```
aws rds create-db-instance \
    --db-cluster-identifier my-serverless-v2-cluster \
    --db-instance-identifier my-serverless-v2-instance \
    --db-instance-class db.serverless \
    --engine aurora-mysql
```

Gestion des clusters de bases de données Aurora Serverless v2

Avec Aurora Serverless v2, vos clusters sont interchangeables avec les clusters provisionnés. Les propriétés Aurora Serverless v2 s'appliquent à une ou plusieurs instances de base de données au sein d'un cluster de bases de données. Ainsi, les procédures de création de clusters, de modification de clusters, de création et de restauration d'instantanés, etc., sont essentiellement les mêmes que pour les autres types de clusters Aurora. Pour obtenir les procédures générales de gestion des clusters et des instances de base de données Aurora, consultez Gestion d'un cluster de bases de données Amazon Aurora.

Dans les rubriques suivantes, vous vous familiariserez avec les considérations relatives à la gestion des clusters contenant des instances de base de données Aurora Serverless v2.

Rubriques

- Définition de la plage de capacité Aurora Serverless v2 d'un cluster
- Vérification de la plage de capacité pour Aurora Serverless v2
- Ajout d'un lecteur Aurora Serverless v2
- Conversion d'un lecteur ou d'un enregistreur approvisionné en Aurora Serverless v2
- Conversion d'un lecteur ou d'un enregistreur Aurora Serverless v2 en mode approvisionné

- Mise en pause des instances de base de données Aurora Serverless v2
- Choix du niveau de promotion pour un lecteur Aurora Serverless v2
- Utilisation de TLS/SSL avec Aurora Serverless v2
- Affichage d'enregistreurs et de lecteurs Aurora Serverless v2
- Journalisation pour Aurora Serverless v2

Définition de la plage de capacité Aurora Serverless v2 d'un cluster

Pour modifier les paramètres de configuration ou d'autres paramètres pour les clusters contenant des instances de base de données Aurora Serverless v2, ou pour les instances de base de données elles-mêmes, suivez les mêmes procédures générales que pour les clusters approvisionnés. Pour en savoir plus, consultez Modification d'un cluster de bases de données Amazon Aurora.

Le paramètre le plus important propre à Aurora Serverless v2 est la plage de capacité. Après avoir défini les valeurs minimale et maximale d'unité de capacité Aurora (ACU) pour un cluster Aurora, vous n'avez pas besoin d'ajuster activement la capacité des instances de base de données Aurora Serverless v2 dans le cluster. Aurora le fait pour vous. Ce paramètre est géré au niveau du cluster. Les mêmes valeurs minimale et maximale d'ACU s'appliquent à chaque instance de base de données Aurora Serverless v2 dans le cluster.

Vous pouvez définir les valeurs spécifiques suivantes :

- Minimum ACUs (Nombre minimal d'ACU) : l'instance de base de données Aurora Serverless v2 peut réduire la capacité à ce nombre d'ACU.
- Maximum ACUs (Nombre maximal d'ACU) – L'instance de base de données Aurora Serverless v2 peut augmenter la capacité à ce nombre d'ACU.

La plage de capacité disponible pour Aurora Serverless v2 dépend à la fois de la version du moteur de base de données et de la version de plateforme. Les nouvelles versions du moteur de base de données autorisent une capacité maximale de 256 ACU, une capacité minimale de 0 ACU ou les deux. Pour connaître les plages de capacité des différentes versions du moteur de base de données, consultez Capacité Aurora Serverless v2.

Pour la fonctionnalité de pause et de reprise automatiques activée en définissant la capacité minimale sur 0 ACU, consultez Réduction verticale à zéro ACU avec pause et reprise automatiques pour Aurora Serverless v2.

Pour déterminer comment garantir que vos clusters de bases de données Aurora Serverless v2 peuvent être augmentés verticalement jusqu'à atteindre 256 ACU, consultez Mise à niveau de votre cluster de bases de données Aurora Serverless v2 pour permettre une mise à l'échelle atteignant jusqu'à 256 ACU.

> **Note**
>
> Lorsque vous modifiez la plage de capacité d'un cluster de bases de données Aurora Serverless v2, la modification intervient immédiatement, que vous choisissiez de l'appliquer immédiatement ou lors du prochain créneau de maintenance planifié.

Dans Aurora Serverless v2, vous ne pouvez pas définir directement la capacité actuelle sans modifier la plage de capacité, car la capacité s'ajuste dynamiquement en fonction de la charge de travail dans la plage spécifiée. Toutefois, vous pouvez indirectement influencer la capacité de la manière suivante :

- Ajuster la capacité minimale : réduisez temporairement la capacité minimale pour réduire la capacité de base lorsque les charges de travail sont faibles.
- Interrompre temporairement la mise à l'échelle : pour fixer la capacité sur une valeur spécifique, réglez les capacités minimale et maximale au même niveau.
- Mettre à l'échelle de manière proactive en cas d'utilisation intensive : si vous prévoyez des charges de travail élevées, augmentez au préalable la capacité minimale afin de maintenir une base de référence plus élevée pendant les périodes de forte activité.

Pour plus de détails sur les effets de la plage de capacité et sur la procédure de surveillance et d'ajustement de cette dernière, consultez Métriques Amazon CloudWatch importantes pour Aurora Serverless v2 et Performances et mise à l'échelle pour Aurora Serverless v2. Votre objectif est de garantir que la capacité maximale du cluster est suffisamment élevée pour gérer les pics de charge de travail et que sa capacité minimale est suffisamment faible pour réduire les coûts lorsque le cluster n'est pas occupé.

Supposons que vous déterminiez, en fonction de votre surveillance, que la plage d'ACU du cluster doit être supérieure, inférieure, plus large ou plus étroite. Vous pouvez définir la capacité d'un cluster Aurora sur une plage spécifique d'ACU avec AWS Management Console, AWS CLI ou l'API Amazon RDS. Cette plage de capacité s'applique à chaque instance de base de données Aurora Serverless v2 dans le cluster.

Supposons par exemple que votre cluster ait une plage de capacité comprise entre 1 et 16 ACU et qu'il contienne deux instances de base de données Aurora Serverless v2. Le cluster dans son ensemble consomme entre 2 ACU (lorsqu'il est inactif) et 32 ACU (lorsqu'il est entièrement utilisé). Si vous modifiez la plage de capacité en passant de 8 à 40,5 ACU, le cluster consomme désormais 16 ACU lorsqu'il est inactif et jusqu'à 81 ACU lorsqu'il est entièrement utilisé.

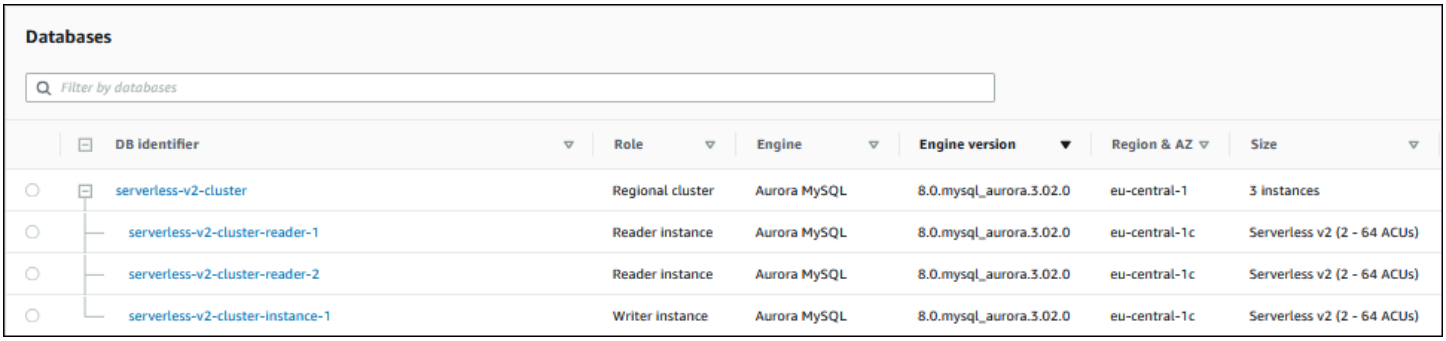
Aurora définit automatiquement certains paramètres pour les instances de base de données Aurora Serverless v2 sur des valeurs qui dépendent de la valeur maximale d'ACU dans la plage de capacité. Pour obtenir la liste de ces paramètres, consultez Nombre maximal de connexions pour Aurora Serverless v2. Pour les paramètres statiques qui reposent sur ce type de calcul, la valeur est à nouveau évaluée lorsque vous redémarrez l'instance de base de données. Vous pouvez ainsi mettre à jour la valeur de ces paramètres en redémarrant l'instance de base de données après avoir modifié la plage de capacité. Pour vérifier si vous devez redémarrer votre instance de base de données afin d'appliquer les modifications apportées aux paramètres, vérifiez l'attribut `ParameterApplyStatus` de l'instance de base de données. La valeur `pending-reboot` indique que le redémarrage appliquera les modifications à certaines valeurs de paramètres.

Console

Vous pouvez définir la plage de capacité d'un cluster qui contient des instances de base de données Aurora Serverless v2 avec AWS Management Console.

Lorsque vous utilisez la console, vous définissez la plage de capacité du cluster au moment d'ajouter la première instance de base de données Aurora Serverless v2 à ce cluster. Vous pouvez le faire lorsque vous choisissez la classe d'instance de base de données sans serveur v2 pour l'instance de base de données d'enregistreur lorsque vous créez le cluster. Vous pouvez également le faire lorsque vous choisissez la classe d'instance de base de données Sans serveur au moment d'ajouter une instance de base de données de lecteur Aurora Serverless v2 au cluster. Vous pouvez aussi le faire lorsque vous convertissez une instance de base de données approvisionnée existante dans le cluster en classe d'instance de base de données Sans serveur. Pour en savoir plus sur les versions complètes de ces procédures, consultez Création d'une instance de base de données d'enregistreur Aurora Serverless v2, Ajout d'un lecteur Aurora Serverless v2 et Conversion d'un lecteur ou d'un enregistreur approvisionné en Aurora Serverless v2.

Quelle que soit la plage de capacité que vous définissez au niveau du cluster, elle s'applique à toutes les instances de base de données Aurora Serverless v2 de votre cluster. L'image suivante représente un cluster contenant plusieurs instances de base de données de lecteur Aurora Serverless v2. Chacune possède une plage de capacité identique de 2 à 64 ACU.

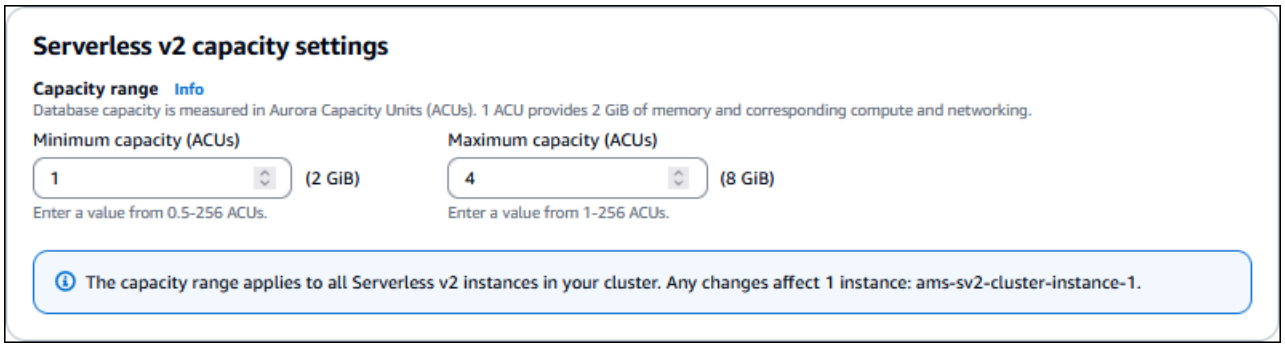


Pour modifier la plage de capacité d'un cluster Aurora Serverless v2

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster contenant vos instances de base de données Aurora Serverless v2 dans la liste. Le cluster doit déjà contenir au moins une instance de base de données Aurora Serverless v2. Sinon, Aurora n'affiche pas la section Capacity range (Plage de capacité).
4. Pour Actions, choisissez Modify (Modifier).
5. Dans la section Capacity range (Plage de capacité), choisissez les valeurs suivantes :

   a. Saisissez une valeur pour Minimum ACUs (Nombre minimal d'ACU). La console affiche la plage de valeurs autorisée. Vous pouvez choisir une capacité minimale comprise entre 0 et 256 ACU. Vous pouvez choisir une capacité maximale comprise entre 1 et 256 ACU. Vous pouvez ajuster les valeurs de capacité par incréments de 0,5 ACU. La plage de capacité disponible dépend à la fois de la version de votre moteur de base de données et de la version de plateforme.

   b. Saisissez une valeur pour Maximum ACUs (Nombre maximal d'ACU). Cette valeur doit être supérieure ou égale à la valeur de Minimum ACUs (Nombre minimal d'ACU). La console affiche la plage de valeurs autorisée. La figure suivante illustre ce choix.



6. Choisissez Continuer. La page Récapitulatif des modifications s'affiche.

7. Choisissez Apply immediately (Appliquer immédiatement).

   La modification de la capacité a lieu immédiatement, que vous choisissiez de l'appliquer immédiatement ou au cours du prochain créneau de maintenance planifié.
8. Choisissez Modifier le cluster pour accepter le récapitulatif des modifications. Vous pouvez également choisir Précédent pour modifier vos modifications ou Annuler pour les ignorer.

AWS CLI

Pour définir la capacité d'un cluster dans lequel vous prévoyez d'utiliser des instances de base de données Aurora Serverless v2 à l'aide d'AWS CLI, exécutez la commande modify-db-cluster d'AWS CLI. Spécifiez l'option `--serverless-v2-scaling-configuration`. Il est possible que le cluster contienne déjà une ou plusieurs instances de base de données Aurora Serverless v2. Vous pouvez également ajouter les instances de base de données ultérieurement. Les valeurs valides pour les champs `MinCapacity` et `MaxCapacity` sont les suivantes :

- `0`, `0.5`, `1`, `1.5`, `2`, etc. par paliers de 0,5, jusqu'à un maximum de 256. La valeur d'ACU minimale 0 n'est disponible que pour les versions d'Aurora qui prennent en charge la fonctionnalité de pause automatique.

La capacité maximale disponible dépend à la fois de la version de votre moteur de base de données et de la version de la plateforme.

Dans cet exemple, vous définissez la plage d'ACU d'un cluster de bases de données Aurora nommé`sample-cluster` sur une valeur minimale de `48.5` et une valeur maximale de 64.

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster \
  --serverless-v2-scaling-configuration MinCapacity=48.5,MaxCapacity=64
```

La modification de la capacité a lieu immédiatement, que vous choisissiez de l'appliquer immédiatement ou au cours du prochain créneau de maintenance planifié.

Vous pouvez ensuite ajouter les instances de base de données Aurora Serverless v2 au cluster et chaque nouvelle instance de base de données peut être mise à l'échelle entre 48,5 et 64 ACU. La nouvelle plage de capacité s'applique également à toutes les instances de base de données Aurora Serverless v2 qui se trouvaient déjà dans le cluster. Les instances de base de données font l'objet d'une augmentation ou d'une réduction d'échelle si nécessaire pour se situer dans la nouvelle plage de capacité.

Pour obtenir d'autres exemples de définition de la plage de capacité à l'aide de l'interface de ligne de commande, consultez Choix de la plage de capacité Aurora Serverless v2 pour un cluster Aurora.

Pour modifier la configuration de mise à l'échelle d'un cluster de bases de données Aurora Serverless à partir de l'AWS CLI, exécutez la commande modify-db-cluster de l'AWS CLI. Spécifiez l'option `--serverless-v2-scaling-configuration` pour configurer la capacité minimale et la capacité maximale. Les valeurs de capacité valides sont notamment les suivantes :

- Aurora MySQL : `0`, `0.5`, `1`, `1.5`, `2`, etc. par incréments de 0,5 ACU jusqu'à un maximum de `256`.
- Aurora PostgreSQL : `0`, `0.5`, `1`, `1.5`, `2`, etc. par incréments de 0,5 ACU jusqu'à un maximum de `256`.
- La valeur d'ACU minimale 0 n'est disponible que pour les versions d'Aurora qui prennent en charge la fonctionnalité de pause automatique.

Dans l'exemple suivant, vous modifiez la configuration de mise à l'échelle d'une instance de base de données Aurora Serverless v2 nommée `sample-instance` qui fait partie d'un cluster nommé `sample-cluster`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster \
--serverless-v2-scaling-configuration MinCapacity=8,MaxCapacity=64
```

Pour Windows :

```
aws rds modify-db-cluster --db-cluster-identifier sample-cluster ^
--serverless-v2-scaling-configuration MinCapacity=8,MaxCapacity=64
```

API RDS

Vous pouvez définir la capacité d'une instance de base de données Aurora avec l'opération d'API ModifyDBCluster. Spécifiez le paramètre `ServerlessV2ScalingConfiguration`. Les valeurs valides pour les champs `MinCapacity` et `MaxCapacity` sont les suivantes :

- `0`, `0.5`, `1`, `1.5`, `2`, etc. par paliers de 0,5, jusqu'à un maximum de 256. La valeur d'ACU minimale 0 n'est disponible que pour les versions d'Aurora qui prennent en charge la fonctionnalité de pause automatique.

Vous pouvez modifier la configuration de mise à l'échelle d'un cluster contenant des instances de base de données Aurora Serverless v2 avec l'opération d'API ModifyDBCluster. Spécifiez le paramètre `ServerlessV2ScalingConfiguration` pour configurer la capacité minimale et la capacité maximale. Les valeurs de capacité valides sont notamment les suivantes :

- Aurora MySQL : `0`, `0.5`, `1`, `1.5`, `2`, etc. par incréments de 0,5 ACU jusqu'à un maximum de `256`.
- Aurora PostgreSQL : `0`, `0.5`, `1`, `1.5`, `2`, etc. par incréments de 0,5 ACU jusqu'à un maximum de `256`.
- La valeur d'ACU minimale 0 n'est disponible que pour les versions d'Aurora qui prennent en charge la fonctionnalité de pause automatique.

La capacité maximale disponible dépend à la fois de la version de votre moteur de base de données et de la version de la plateforme.

La modification de la capacité a lieu immédiatement, que vous choisissiez de l'appliquer immédiatement ou au cours du prochain créneau de maintenance planifié.

Mise à niveau de votre cluster de bases de données Aurora Serverless v2 pour permettre une mise à l'échelle atteignant jusqu'à 256 ACU

Dans certains cas, lorsque vous essayez de mettre à l'échelle votre cluster de bases de données Aurora Serverless v2 afin d'atteindre des capacités supérieures à 128 ACU, vous recevez un message d'erreur. Le message d'erreur vous indique quelles instances sont incompatibles avec la nouvelle plage de mise à l'échelle. Cela met en évidence la relation importante entre votre plage de capacité, la version du moteur de base de données et la version de plateforme.

L'impossibilité d'aller au-delà de 128 ACU peut se produire pour l'une des deux raisons suivantes :

- Ancienne version du moteur de base de données : mettez à niveau la version du moteur de base de données vers une version prenant en charge 256 ACU. Pour plus d'informations, consultez Capacité Aurora Serverless v2.
- Ancienne version de plateforme : mettez à niveau la plateforme pour votre cluster de bases de données Aurora Serverless v2. Vous pouvez effectuer cette opération de différentes manières :
  - Arrêtez et redémarrez le cluster de bases de données. Lorsque le cluster redémarrera, il utilisera la dernière version de plateforme compatible, qui peut autoriser un nombre maximum d'ACU plus élevé.

Cependant, l'arrêt et le démarrage d'un cluster de bases de données entraînent une durée d'indisponibilité, généralement de plusieurs minutes. Pour plus d'informations, consultez Arrêt et démarrage d'un cluster de bases de données Amazon Aurora.

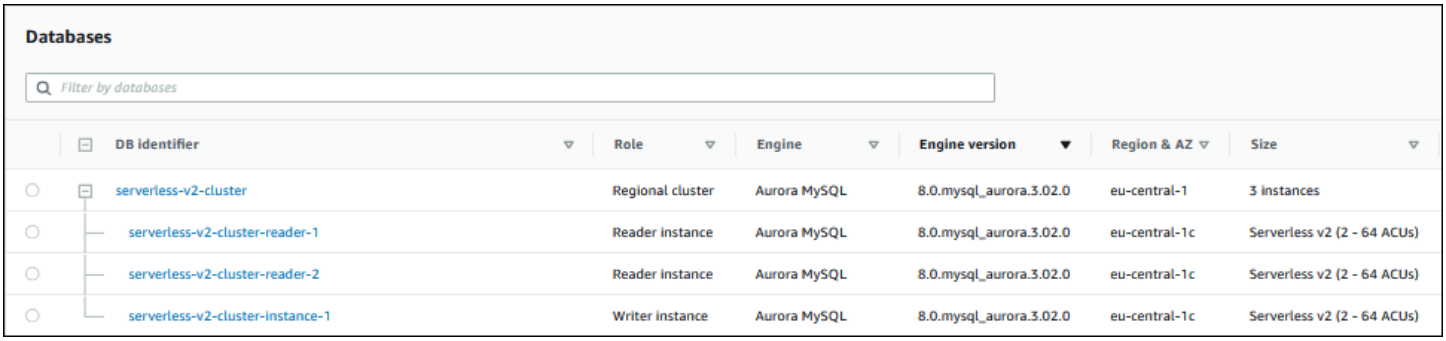
- Utilisez un déploiement bleu/vert. Pour plus d'informations, consultez Présentation des (Amazon Aurora Blue/Green).
  1. Créez un déploiement bleu/vert. Pour plus d'informations, consultez Création d'un blue/green déploiement dans).
  2. Vérifiez que vous pouvez définir la capacité maximale de l'environnement intermédiaire (vert) sur 256 ACU.
  3. Passez à l'environnement vert. Pour plus d'informations, consultez Changer de blue/green déploiement dans).
  4. Supprimez le cluster de bases de données d'origine.

  > **Note**
  >
  > Si vous conservez les informations d'identification de votre base de données dans AWS Secrets Manager, vous ne pouvez pas utiliser les déploiements bleu/vert. Aurora Global Database ne prend pas en charge les déploiements bleu/vert.

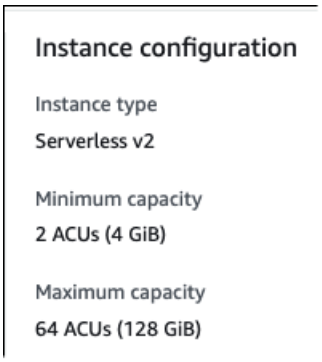
Vérification de la plage de capacité pour Aurora Serverless v2

La procédure de vérification de la plage de capacité de votre cluster Aurora Serverless v2 exige de commencer par définir une plage de capacité. Si vous ne l'avez pas fait, suivez la procédure de la rubrique Définition de la plage de capacité Aurora Serverless v2 d'un cluster.

Quelle que soit la plage de capacité que vous définissez au niveau du cluster, elle s'applique à toutes les instances de base de données Aurora Serverless v2 de votre cluster. L'image suivante représente un cluster contenant plusieurs instances de base de données Aurora Serverless v2. Chacune possède une plage de capacité identique.



Vous pouvez également afficher la page de détails de n'importe quelle instance de base de données Aurora Serverless v2 du cluster. La plage de capacité des instances de base de données s'affiche dans l'onglet Configuration.



Vous pouvez également consulter la plage de capacité actuelle du cluster sur la page Modify (Modifier) du cluster. À ce stade, vous pouvez modifier la plage de capacité. Pour connaître toutes les façons de définir ou modifier la plage de capacité, consultez Définition de la plage de capacité Aurora Serverless v2 d'un cluster.

Vérification de la plage de capacité actuelle d'un cluster Aurora

Vous pouvez vérifier la plage de capacité configurée pour les instances de base de données Aurora Serverless v2 d'un cluster en examinant l'attribut `ServerlessV2ScalingConfiguration` du cluster. L'exemple d'AWS CLI suivant illustre un cluster dont la capacité minimale est de 0,5 unités de capacité Aurora (ACU) et la capacité maximale est de 16 ACU.

```
$ aws rds describe-db-clusters --db-cluster-identifier serverless-v2-64-acu-cluster \
  --query 'DBClusters[*].[ServerlessV2ScalingConfiguration]'
[
    [
        {
            "MinCapacity": 0.5,
            "MaxCapacity": 16.0
```

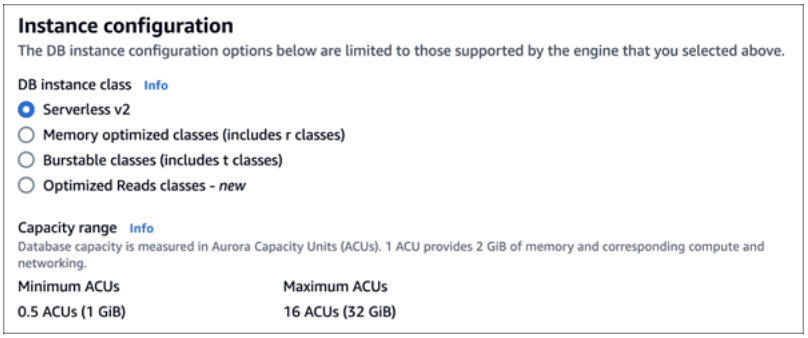
```
        }
    ]
]
```

Ajout d’un lecteur Aurora Serverless v2

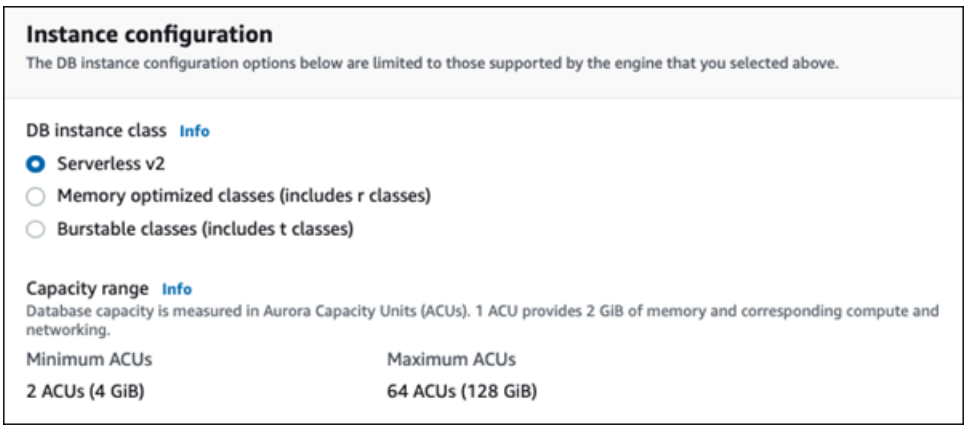
Pour ajouter une instance de base de données de lecteur Aurora Serverless v2 à votre cluster, suivez la même procédure générale que celle de la rubrique Ajout de réplicas Aurora à un cluster de bases de données. Choisissez la classe d’instance Sans serveur v2 pour la nouvelle instance de base de données.

Si l’instance de base de données de lecteur est la première instance de base de données Aurora Serverless v2 du cluster, vous choisissez également la plage de capacité. Ce paramètre s’applique à cette instance de base de données de lecteur et à toutes les autres instances de base de données Aurora Serverless v2 que vous ajoutez au cluster. Chaque instance de base de données Aurora Serverless v2 peut être mise à l’échelle entre les valeurs minimale et maximale d’ACU.

Si d’autres instances Aurora Serverless v2 se trouvent déjà dans le cluster, la boîte de dialogue Ajouter un lecteur indique la plage de capacité actuelle du cluster. Dans ce cas, vous ne pouvez pas modifier la capacité. L’image suivante présente le rapport de la capacité actuelle du cluster.



Si vous avez déjà ajouté des instances de base de données Aurora Serverless v2 au cluster, l’ajout d’une autre instance de base de données de lecteur Aurora Serverless v2 affiche la plage de capacité actuelle. L’image suivante illustre ces paramètres en lecture seule.



Si vous souhaitez modifier la plage de capacité du cluster, suivez la procédure de la rubrique Définition de la plage de capacité Aurora Serverless v2 d'un cluster.

Pour les clusters contenant plusieurs instances de base de données de lecteur, la priorité de basculement de chaque instance de base de données de lecteur Aurora Serverless v2 joue un rôle important dans l'augmentation et la réduction d'échelle de cette instance de base de données. Vous ne pouvez pas spécifier la priorité lors de la création initiale du cluster. Gardez cette propriété à l'esprit lorsque vous ajoutez une deuxième instance de base de données de lecteur à votre cluster. Pour plus d'informations, consultez Choix du niveau de promotion pour un lecteur Aurora Serverless v2.

Pour savoir comment afficher la plage de capacité actuelle d'un cluster, consultez Vérification de la plage de capacité pour Aurora Serverless v2.

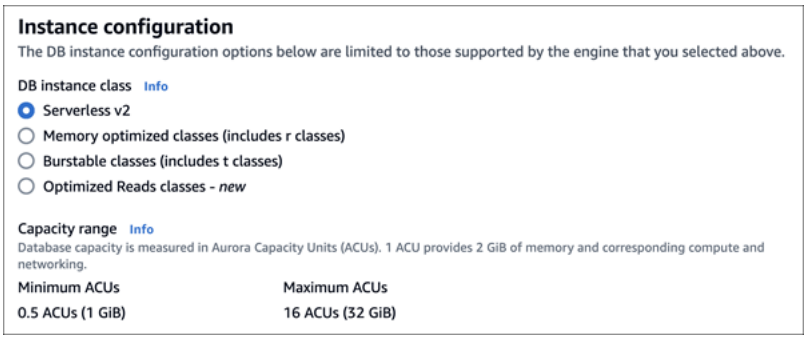
Conversion d'un lecteur ou d'un enregistreur approvisionné en Aurora Serverless v2

Vous pouvez convertir une instance de base de données approvisionnée de sorte qu'elle utilise Aurora Serverless v2. Pour ce faire, suivez la procédure décrite à la rubrique Modification d'une instance de base de données dans un cluster de bases de données. Le cluster doit satisfaire aux Exigences et limites relatives à Aurora Serverless v2. Par exemple, les instances de base de données Aurora Serverless v2 exigent que le cluster exécute certaines versions minimales du moteur.

Supposons que vous convertissiez un cluster approvisionné en cours d'exécution pour tirer parti d'Aurora Serverless v2. Dans ce cas, vous pouvez réduire la durée d'indisponibilité en convertissant une instance de base de données en Aurora Serverless v2 dans le cadre de la première étape du

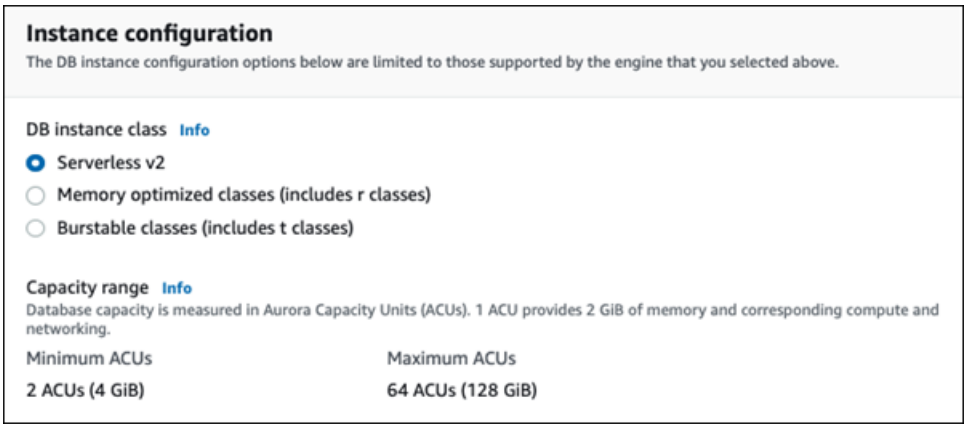
processus de bascule. Pour obtenir la procédure complète, consultez Basculement d'un cluster approvisionné vers Aurora Serverless v2.

Si l'instance de base de données que vous convertissez est la première instance de base de données Aurora Serverless v2 du cluster, vous choisissez la plage de capacité du cluster dans le cadre de l'opération Modify (Modifier). Cette plage de capacité s'applique à chaque instance de base de données Aurora Serverless v2 que vous ajoutez au cluster. L'image suivante illustre la page sur laquelle vous spécifiez les nombres minimal et maximal d'unités de capacité Aurora (ACU).



Pour plus de détails sur l'importance de la plage de capacité, consultez Capacité Aurora Serverless v2.

Si le cluster contient déjà une ou plusieurs instances de base de données Aurora Serverless v2, la plage de capacité existante s'affiche pendant l'opération Modify (Modifier). L'image suivante illustre un exemple de ce panneau d'informations.



Si vous souhaitez modifier la plage de capacité du cluster après avoir ajouté plusieurs instances de base de données Aurora Serverless v2, suivez la procédure de la rubrique Définition de la plage de capacité Aurora Serverless v2 d'un cluster.

Conversion d'un lecteur ou d'un enregistreur Aurora Serverless v2 en mode approvisionné

Vous pouvez convertir une instance de base de données Aurora Serverless v2 en instance de base de données approvisionnée. Pour ce faire, suivez la procédure décrite à la rubrique Modification d'une instance de base de données dans un cluster de bases de données. Choisissez une classe d'instance de base de données autre que Sans serveur.

Vous pouvez convertir une instance de base de données Aurora Serverless v2 en mode approvisionné si elle a besoin d'une capacité supérieure à celle disponible avec le nombre maximal d'unités de capacité Aurora (ACU) d'une instance de base de données Aurora Serverless v2. Par exemple, les classes d'instance de base de données db.r5 et db.r6g les plus grandes ont une capacité de mémoire supérieure à celle à laquelle une instance de base de données Aurora Serverless v2 peut être mise à l'échelle.

> **Tip**
>
> Certaines des anciennes classes d'instance de base de données telles que db.r3 et db.t2 ne sont pas disponibles pour les versions d'Aurora que vous utilisez avec Aurora Serverless v2. Pour savoir quelles classes d'instance de base de données vous pouvez utiliser lors de la conversion d'une instance de base de données Aurora Serverless v2 en mode approvisionné, consultez Moteurs de base de données pris en charge pour les classes d'instance de base de données.

Si vous convertissez l'instance de base de données d'enregistreur de votre cluster d'Aurora Serverless v2 en mode approvisionné, vous pouvez suivre la procédure de la rubrique Basculement d'un cluster approvisionné vers Aurora Serverless v2, mais en sens inverse. Basculez l'une des instances de base de données de lecteur du cluster d'Aurora Serverless v2 vers le mode approvisionné. Effectuez ensuite un basculement pour intégrer cette instance de base de données approvisionnée dans l'enregistreur.

Toute plage de capacité que vous avez précédemment spécifiée pour le cluster reste en place, même si toutes les instances de base de données Aurora Serverless v2 sont retirées du cluster. Si vous souhaitez modifier la plage de capacité, vous pouvez modifier le cluster, comme expliqué à la rubrique Définition de la plage de capacité Aurora Serverless v2 d'un cluster.

Mise en pause des instances de base de données Aurora Serverless v2

Vous pouvez configurer les clusters Aurora pour mettre en pause et reprendre automatiquement leurs instances de base de données Aurora Serverless v2. Pour plus d'informations, consultez Réduction verticale à zéro ACU avec pause et reprise automatiques pour Aurora Serverless v2.

Choix du niveau de promotion pour un lecteur Aurora Serverless v2

Pour les clusters contenant plusieurs instances de base de données Aurora Serverless v2 ou un mélange d'instances de base de données approvisionnées et Aurora Serverless v2, prêtez attention au paramètre de niveau de promotion pour chaque instance de base de données Aurora Serverless v2. Ce paramètre contrôle davantage le comportement des instances de base de données Aurora Serverless v2 que celui des instances de base de données approvisionnées.

Dans AWS Management Console, vous spécifiez cette valeur à l'aide du paramètre Failover priority (Priorité de basculement) sous Additional configuration (Configuration supplémentaire) pour les pages Create database (Créer une base de données), Modify instance (Modifier une instance) et Add reader (Ajouter un lecteur). Cette propriété s'affiche pour les instances de base de données existantes dans la colonne facultative Niveau de priorité sur la page Bases de données. Cette propriété est également disponible sur la page de détails d'un cluster de bases de données ou d'une instance de base de données.

Pour les instances de base de données approvisionnées, le choix du niveau 0–15 détermine uniquement l'ordre dans lequel Aurora choisit l'instance de base de données de lecteur à promouvoir en enregistreur lors d'une opération de basculement. Pour les instances de base de données de lecteur Aurora Serverless v2, le numéro de niveau détermine également si l'instance de base de données fait l'objet d'une augmentation d'échelle pour correspondre à la capacité de l'instance de base de données d'enregistreur ou si elle est mise à l'échelle indépendamment en fonction de sa propre charge de travail. Les instances de base de données de lecteur Aurora Serverless v2 de niveau 0 ou 1 restent à une capacité minimale au moins égale à celle de l'instance de base de données d'enregistreur. De cette façon, elles sont prêtes à prendre le relais de l'instance de base de données d'enregistreur en cas de basculement. Si l'instance de base de données d'enregistreur est une instance de base de données approvisionnée, Aurora estime la capacité Aurora Serverless v2 équivalente. Il utilise cette estimation comme capacité minimale pour l'instance de base de données de lecteur Aurora Serverless v2.

Les instances de base de données de lecteur Aurora Serverless v2 des niveaux 2 à 15 n'ont pas la même contrainte de capacité minimale. Lorsqu'elles sont inactives, elles peuvent faire l'objet d'une

réduction d'échelle à la valeur minimale d'unité de capacité Aurora (ACU) spécifiée dans la plage de capacité du cluster.

L'exemple d'AWS CLI suivant sous Linux montre comment examiner les niveaux de promotion de toutes les instances de base de données de votre cluster. Le dernier champ comporte la valeur `True` pour l'instance de base de données d'enregistreur et la valeur `False` pour toutes les instances de base de données de lecteur.

```
$ aws rds describe-db-clusters --db-cluster-identifier promotion-tier-demo \
  --query 'DBClusters[*].DBClusterMembers[*].
[PromotionTier,DBInstanceIdentifier,IsClusterWriter]' \
  --output text

1   instance-192  True
1   tier-01-4840  False
10  tier-10-7425  False
15  tier-15-6694  False
```

L'exemple d'AWS CLI suivant sous Linux montre comment modifier le niveau de promotion d'une instance de base de données spécifique de votre cluster. Les commandes commencent par modifier l'instance de base de données avec un nouveau niveau de promotion. Elles attendent ensuite que l'instance de base de données soit à nouveau disponible et confirment le nouveau niveau de promotion pour l'instance de base de données.

```
$ aws rds modify-db-instance --db-instance-identifier instance-192 --promotion-tier 0
$ aws rds wait db-instance-available --db-instance-identifier instance-192
$ aws rds describe-db-instances --db-instance-identifier instance-192 \
  --query '*[].[PromotionTier]' --output text
0
```

Pour plus d'informations sur la spécification de niveaux de promotion pour différents cas d'utilisation, consultez Mise à l'échelle d'Aurora Serverless v2.

Utilisation de TLS/SSL avec Aurora Serverless v2

Aurora Serverless v2 peut utiliser le protocole TLS/SSL (Transport Layer Security/Secure Sockets Layer) pour chiffrer les communications entre les clients et vos instances de base de données Aurora Serverless v2. Il prend en charge les versions TLS/SSL 1.0, 1.1 et 1.2. Pour obtenir des informations générales sur l'utilisation de TLS/SSL avec Aurora, consultez Connexions TLS aux clusters de bases de données Aurora MySQL.

Pour en savoir plus sur la connexion à la base de données Aurora MySQL avec le client MySQL, consultez Connexion à une instance de base de données exécutant le moteur de base de données MySQL.

Aurora Serverless v2 prend en charge tous les modes TLS/SSL disponibles pour le client MySQL (`mysql`) et le client PostgreSQL (`psql`), y compris les modes répertoriés dans le tableau suivant.

Description du mode TLS/SSL	mysql	psql
Se connecte sans utiliser TLS/SSL.	DISABLED	désactiver
Essaie de se connecter à l'aide de TLS/SSL, mais revient à non-SSL, si nécessaire.	PREFERRED	prefer (par défaut)
Imposer à l'aide de TLS/SSL.	REQUIRED	require
TLS/SSL est obligatoire et une vérification de l'autorité de certification (CA) est effectuée.	VERIFY_CA	verify-ca
Impose TLS/SSL, vérifie l'autorité de certification et son nom d'hôte.	VERIFY_IDENTITY	verify-full

Aurora Serverless v2 utilise des certificats à caractères génériques. Si vous spécifiez l'option « Vérifier l'autorité de certification » ou « Vérifier l'autorité de certification et son nom d'hôte » lors de l'utilisation de TLS/SSL, commencez par télécharger le référentiel d'approbations Amazon Root CA 1 à partir d'Amazon Trust Services. Vous pouvez ensuite identifier ce fichier au format PEM dans votre commande client. Pour ce faire à l'aide du client PostgreSQL, procédez comme suit.

Pour Linux, macOS ou Unix :

```
psql 'host=endpoint user=user sslmode=require sslrootcert=amazon-root-CA-1.pem
 dbname=db-name'
```

Pour en savoir plus sur l'utilisation de la base de données Aurora PostgreSQL à l'aide du client Postgres, consultez Connexion à une instance de base de données exécutant le moteur de base de données PostgreSQL.

Pour plus d'informations sur la connexion aux clusters de bases de données Aurora, consultez Connexion à un cluster de bases de données Amazon Aurora.

Suites de chiffrement prises en charge pour les connexions aux clusters de bases de données Aurora Serverless v2

L'utilisation de suites de chiffrement configurables vous permet d'avoir plus de contrôle sur la sécurité des connexions de vos bases de données. Vous pouvez spécifier une liste de suites de chiffrement que vous souhaitez autoriser pour la sécurisation des connexions TLS/SSL client à votre base de données. Avec les suites de chiffrement configurables, vous pouvez contrôler le chiffrement de connexion accepté par votre serveur de base de données. Cela évite d'utiliser des chiffrements qui ne sont pas sécurisés ou qui ne sont plus utilisés.

Les clusters de bases de données Aurora Serverless v2 basés sur Aurora MySQL prennent en charge les mêmes suites de chiffrement que les clusters de bases de données provisionnés Aurora MySQL. Pour plus d'informations sur ces suites de chiffrement, consultez Configuration de suites de chiffrement pour les connexions aux clusters de bases de données Aurora MySQL.

Les clusters de bases de données Aurora Serverless v2 basés sur Aurora PostgreSQL prennent en charge les mêmes suites de chiffrement que les clusters de bases de données provisionnés Aurora PostgreSQL. Pour plus d'informations sur ces suites de chiffrement, consultez Configuration de suites de chiffrement pour les connexions aux clusters de bases de données Aurora PostgreSQL.

Affichage d'enregistreurs et de lecteurs Aurora Serverless v2

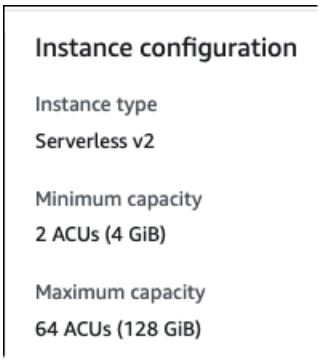
Vous pouvez afficher les détails des instances de base de données Aurora Serverless v2 de la même manière que pour les instances de base de données approvisionnées. Pour ce faire, suivez la procédure générale de la rubrique Affichage d'un cluster de bases de données Amazon Aurora. Un cluster peut contenir toutes les instances de base de données Aurora Serverless v2, toutes les instances de base de données approvisionnées ou quelques-unes de chaque type.

Après avoir créé une ou plusieurs instances de base de données Aurora Serverless v2, vous pouvez voir les instances de base de données qui sont de type Sans serveur et celles qui sont de type Instance. Vous pouvez également afficher les nombres minimal et maximal d'unités de capacité

Aurora (ACU) que l'instance de base de données Aurora Serverless v2 peut utiliser. Chaque unité de capacité est une combinaison de traitement (UC) et de capacité mémoire (RAM). Cette plage de capacité s'applique à chaque instance de base de données Aurora Serverless v2 dans le cluster. Pour obtenir la procédure de vérification de la plage de capacité d'un cluster ou d'une instance de base de données Aurora Serverless v2 dans le cluster, consultez Vérification de la plage de capacité pour Aurora Serverless v2.

Dans AWS Management Console, les instances de base de données Aurora Serverless v2 sont marquées sous la colonne Size (Taille) sur la page Databases (Bases de données). Les instances de base de données approvisionnées affichent le nom d'une classe d'instance de base de données telle que r6g.xlarge. Les instances de base de données Aurora Serverless indiquent Sans serveur pour la classe d'instance de base de données, ainsi que les capacités minimale et maximale de l'instance de base de données. Par exemple, Sans serveur v2 (4–64 ACUs) ou Sans serveur v2 (1–40 ACUs) peuvent s'afficher.

Vous trouverez les mêmes informations dans l'onglet Configuration de chaque instance de base de données Aurora Serverless v2 dans la console. Par exemple, une section Instance type (Type d'instance) qui ressemble à ce qui suit peut s'afficher. Ici, la valeur de Type d'instance est Sans serveur v2, la valeur de Capacité minimale est 2 ACU (4 Gio) et la valeur de Capacité maximale est 64 ACU (128 Gio).



Vous pouvez surveiller la capacité de chaque instance de base de données Aurora Serverless v2 au fil du temps. De cette façon, vous pouvez vérifier les nombres minimal, maximal et moyen d'ACU consommées par chaque instance de base de données. Vous pouvez également vérifier à quel point l'instance de base de données s'est rapprochée de sa capacité minimale ou maximale. Pour afficher ces détails dans AWS Management Console, examinez les graphiques des métriques Amazon CloudWatch dans l'onglet Monitoring (Surveillance) de l'instance de base de données. Pour plus d'informations sur les métriques à surveiller et comment les interpréter, consultez Métriques Amazon CloudWatch importantes pour Aurora Serverless v2.

Journalisation pour Aurora Serverless v2

Pour activer la journalisation de la base de données, spécifiez les journaux à activer à l'aide des paramètres de configuration dans votre groupe de paramètres personnalisé.

Pour Aurora MySQL, vous pouvez activer les journaux suivants.

Aurora MySQL	Description
`general_log`	Crée le journal général. Paramétrez sur 1 pour activer. La valeur par défaut est désactivée (0).
`log_queries_not_using_indexes`	Journalise les requêtes dans le journal des requêtes lentes qui n'utilisent pas d'index. La valeur par défaut est désactivée (0). Paramétrez sur 1 pour activer ce journal.
`long_query_time`	Empêche l'enregistrement des requêtes rapides dans le journal des requêtes lentes. Peut être réglé sur une rangée comprise entre 0 et 31 536 000. La valeur par défaut est 0 (non active).
`server_audit_events`	Liste des événements à capturer dans les journaux. Les valeurs prises en charge sont `CONNECT`, `QUERY`, `QUERY_DCL` , `QUERY_DDL` , `QUERY_DML` , et `TABLE`.
`server_audit_logging`	Paramétrez sur 1 pour activer la journalisation d'audit de serveur. Si vous activez cette option, vous pouvez spécifier les événements d'audit à envoyer à CloudWatch en les répertoriant dans le paramètre `server_audit_events` .
`slow_query_log`	Crée un journal des requêtes lentes. Paramétrez sur 1 pour activer le journal des requêtes lentes. La valeur par défaut est désactivée (0).

Pour plus d'informations, consultez Utilisation de l'Audit avancé avec un cluster de bases de données Amazon Aurora MySQL.

Pour Aurora PostgreSQL, vous pouvez activer les journaux suivants sur vos instances de base de données Aurora Serverless v2.

Aurora PostgreSQL	Description
`log_connections`	Enregistre toutes les connexions réussies.
`log_disconnections`	Journalise la fin d'une session, y compris sa durée.
`log_lock_waits`	La valeur par défaut est 0 (désactivée). Paramétrez sur 1 pour journaliser les attentes de verrouillage.
`log_min_duration_statement`	Durée minimale (en millisecondes) d'exécution d'une instruction avant qu'elle ne soit journalisée.
`log_min_messages`	Définit les niveaux des messages qui sont enregistrés. Les valeurs prises en charge sont `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, `panic`. Pour consigner les données de performances dans le journal `postgres` , définissez la valeur sur `debug1`.
`log_temp_files`	Journalise l'utilisation de fichiers temporaires qui sont au-dessus des kilo-octets (Ko) spécifiés.
`log_statement`	Contrôle les instructions SQL spécifiques qui sont journalisées. Les valeurs prises en charge sont `none`, `ddl`, `mod` et `all`. La valeur par défaut est `none`.

Rubriques

Journalisation avec Amazon CloudWatch

Après avoir utilisé la procédure de Journalisation pour Aurora Serverless v2 pour choisir les journaux de base de données à activer, vous pouvez choisir les journaux à charger (« publier ») sur Amazon CloudWatch.

Vous pouvez utiliser Amazon CloudWatch pour analyser les données des journaux, créer des alarmes et afficher des métriques. Par défaut, les journaux d’erreurs pour Aurora Serverless v2 sont activés et automatiquement chargés sur CloudWatch. Vous pouvez également charger d’autres journaux depuis les instances de base de données Aurora Serverless v2 sur CloudWatch.

Vous choisissez ensuite les journaux à charger sur CloudWatch, en utilisant les paramètres Log exports (Exportations de journaux) dans AWS Management Console ou l’option `--enable-cloudwatch-logs-exports` dans AWS CLI.

Vous pouvez choisir les journaux Aurora Serverless v2 à charger sur CloudWatch. Pour plus d’informations, consultez Utilisation de l’Audit avancé avec un cluster de bases de données Amazon Aurora MySQL.

Comme n’importe quel autre type de cluster de bases de données Aurora, vous ne pouvez pas modifier le groupe de paramètres de cluster de bases de données par défaut. Créez votre propre groupe de paramètres de cluster de bases de données basé sur un paramètre par défaut pour votre cluster de bases de données et votre type de moteur. Nous vous recommandons de créer votre groupe de paramètres de cluster de bases de données personnalisé avant de créer votre cluster de bases de données Aurora Serverless v2 et ce, de sorte qu’il soit disponible lorsque vous créez une base de données sur la console.

> **Note**
>
> Pour Aurora Serverless v2, vous pouvez créer un cluster de bases de données et des groupes de paramètres de base de données. Cela contraste avec Aurora Serverless v1, où vous ne pouvez créer que des groupes de paramètres de cluster de bases de données.

Affichage de journaux Aurora Serverless v2 dans Amazon CloudWatch

Après avoir utilisé la procédure de la rubrique Journalisation avec Amazon CloudWatch pour choisir les journaux de base de données à activer, vous pouvez afficher le contenu des journaux.

Pour plus d'informations sur l'utilisation de CloudWatch avec les journaux Aurora MySQL et Aurora PostgreSQL, consultez Surveillance des événements de journaux dans Amazon CloudWatch et Publication de journaux Aurora PostgreSQL sur Amazon CloudWatch Logs.

Pour afficher les journaux de votre cluster de bases de données Aurora Serverless v2

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Choisissez votre Région AWS.
3. Choisissez Groupes de journaux.
4. Choisissez votre journal de cluster de bases de données Aurora Serverless v2 dans la liste. Le modèle de nommage des journaux est le suivant.

   ```
   /aws/rds/cluster/cluster-name/log_type
   ```

> **Note**
>
> Pour les clusters de bases de données Aurora Aurora Serverless v2 compatibles MySQL, le journal des erreurs inclut les événements de mise à l'échelle du pool de mémoire tampon même en l'absence d'erreurs.

Capacité de surveillance avec Amazon CloudWatch

Avec Aurora Serverless v2, vous pouvez utiliser CloudWatch pour surveiller la capacité et l'utilisation de l'ensemble des instances de base de données Aurora Serverless v2 de votre cluster. Vous

pouvez afficher les métriques au niveau de l'instance pour vérifier la capacité de chaque instance de base de données Aurora Serverless v2 en fonction de leur augmentation/réduction d'échelle. Vous pouvez également comparer les métriques de capacité avec d'autres métriques pour voir comment les modifications apportées aux charges de travail affectent la consommation des ressources. Par exemple, vous pouvez comparer `ServerlessDatabaseCapacity` avec `DatabaseUsedMemory`, `DatabaseConnections` et `DMLThroughput` pour évaluer la manière dont votre cluster de bases de données répond lors des opérations. Pour plus de détails sur les métriques de capacité qui s'appliquent à Aurora Serverless v2, consultez Métriques Amazon CloudWatch importantes pour Aurora Serverless v2.

Pour surveiller la capacité de votre cluster de bases de données Aurora Serverless v2

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Choisissez Métriques. Dans la console, toutes les métriques disponibles apparaissent sous forme de cartes regroupées par nom de service.
3. Choisissez RDS.
4. (Facultatif) Utilisez la zone Search (Recherche) pour trouver les métriques qui sont particulièrement importantes pour Aurora Serverless v2 : `ServerlessDatabaseCapacity`, `ACUUtilization`, `CPUUtilization` et `FreeableMemory`.

Nous vous recommandons de configurer un tableau de bord CloudWatch pour surveiller la capacité de votre cluster de bases de données Aurora Serverless v2 à l'aide des métriques de capacité. Pour en savoir plus, consultez Création de tableaux de bord avec CloudWatch.

Pour en savoir plus sur l'utilisation d'Amazon CloudWatch avec Amazon Aurora, consultez Publication de journaux Amazon Aurora MySQL dans Amazon CloudWatch Logs.

Surveillance de l'activité de mise en pause et de reprise d'Aurora Serverless v2

Aurora écrit un fichier journal distinct pour les instances de base de données Aurora Serverless v2 pour lesquelles la pause automatique est activée. Aurora écrit dans le journal pour chaque intervalle de 10 minutes pendant lequel l'instance n'est pas mise en pause. Aurora conserve jusqu'à sept de ces journaux, qui font l'objet d'une rotation quotidienne. Le fichier journal actuel est nommé `instance.log`, et les anciens journaux sont nommés selon le modèle `instance.`*`YYYY-MM-DD`*`.`*`N`*`.log`.

Ce journal est activé par défaut pour les instances de base de données Aurora Serverless v2 pour lesquelles la pause automatique est activée. Vous pouvez consulter le contenu de ce journal dans la

AWS Management Console ou en utilisant l'AWS CLI ou l'API RDSP. Actuellement, vous ne pouvez pas charger ce journal sur CloudWatch.

Les événements répertoriés dans Événements d'instance de base de données fournissent une vue d'ensemble globale des activités de pause et de reprise, tels que les suivants :

- Quand l'instance commence à se mettre en pause et quand elle finit de le faire.
- Quand l'instance commence à reprendre et quand elle finit de reprendre.
- Cas où l'instance a tenté de se mettre en pause, mais certaines conditions l'en ont empêchée.

`instance.log` fournit des informations plus détaillées sur les raisons pour lesquelles une instance Aurora Serverless v2 a pu être mise en pause ou non.

Le journal peut indiquer qu'une instance a repris pour différentes raisons :

- activité utilisateur : demande de connexion à une base de données. Cela peut provenir d'une session client interactive, d'un appel d'API RDS Data ou d'une demande de téléchargement de journaux à partir de l'instance.
- activité en arrière-plan : catégorie générale qui inclut toutes les raisons pour lesquelles Aurora reprend une instance. Par exemple, lorsqu'une demande de connexion à une instance de lecteur entraîne la reprise de l'instance d'enregistreur, le journal du lecteur indique l'activité de l'utilisateur, tandis que le journal de l'enregistreur classe cette demande de reprise comme activité en arrière-plan. Pour connaître les raisons pour lesquelles Aurora peut reprendre une instance autre qu'une demande de connexion utilisateur, consultez Reprise d'une instance Aurora Serverless v2 mise en pause automatique.

Lorsqu'Aurora n'a connaissance d'aucune condition susceptible d'empêcher l'instance de se mettre en pause à l'expiration de l'intervalle de pause automatique, un message d'information est régulièrement écrit dans le journal. Pour les clusters dotés d'une seule instance de base de données, le journal contient le message suivant :

```
[INFO] No auto-pause blockers registered since time
```

Pour les clusters comportant plusieurs instances de base de données, le message est légèrement différent. Cela est dû au fait qu'un enregistreur peut être incapable de se mettre en pause en raison de l'activité sur l'une des instances du lecteur. Si l'activité du lecteur se termine avant l'expiration de

l'intervalle de pause automatique pour l'enregistreur, ce dernier pourra se mettre en pause à l'heure prévue.

```
[INFO] No auto-pause blockers registered since time.
Database might be required to maintain compute capacity for high availability.
```

Si une opération de pause démarre, mais qu'une nouvelle demande de connexion à la base de données arrive avant la fin de la pause de l'instance, le journal contient le message suivant :

```
[INFO] Unable to pause database due to a new database activity
```

Si Aurora découvre des conditions qui empêchent définitivement l'instance de se mettre en pause, le journal contient le message suivant qui répertorie toutes ces conditions :

```
[INFO] Auto-pause blockers registered since time: list_of_conditions
```

De cette façon, Aurora ne vous empêche pas d'activer la réplication, l'intégration zéro ETL, Aurora Global Database, etc. en combinaison avec la fonction de pause automatique. Le journal vous informe lorsque l'utilisation de telles fonctionnalités peut empêcher la mise en pause automatique de prendre effet.

Les raisons suivantes peuvent expliquer pourquoi une instance Aurora Serverless v2 peut dépasser le délai d'expiration de la pause automatique, sans toutefois pouvoir se mettre en pause :

- activité de base de données avant l'expiration du délai de pause automatique : l'instance de base de données a reçu une demande de connexion avant l'expiration du délai d'expiration.
- membre de la base de données globale : si le cluster de bases de données fait partie d'une base de données globale Aurora, les instances Aurora Serverless v2 de ce cluster ne sont pas mises en pause. Un cluster peut passer d'un cluster autonome à un cluster de bases de données global. Ainsi, des instances précédemment mises en pause automatique peuvent arrêter de le faire et indiquer cette raison dans le journal. Une fois qu'un cluster devient membre d'une base de données globale, il ne redevient un cluster autonome que lorsque vous le détachez explicitement. Le cluster principal est toujours considéré comme faisant partie de la base de données globale même si vous détachez tous les clusters secondaires.
- capacité de réplication configurée : la réplication spécifique au moteur est activée sur l'instance de base de données de l'enregistreur, qu'il s'agisse de la réplication du journal binaire pour MySQL ou de la réplication logique pour PostgreSQL. Cette condition peut également être due à l'utilisation

d'une autre fonctionnalité Aurora nécessitant l'activation de la réplication, telle que les intégrations zéro ETL ou Database Activity Streams (DAS).

- délai de sauvegarde continu : si le système de stockage Aurora n'a pas fini d'appliquer les modifications de stockage jusqu'à la date actuelle, l'instance de l'enregistreur ne s'arrête pas tant qu'elle n'a pas rattrapé son retard. Cette condition n'affecte que l'instance de l'enregistreur et devrait être relativement brève.
- action de maintenance du service ou du client : si une opération de maintenance démarre, l'instance de base de données ne se mettra pas en pause à nouveau tant que cette opération ne sera pas terminée. Cette condition inclut une grande variété d'opérations qui peuvent être lancées par vous ou par Aurora, telles que les mises à jour, le clonage, la modification des paramètres de configuration, les mises à niveau, le téléchargement de fichiers journaux, etc. Cet événement se produit également lorsque vous demandez la suppression d'une instance et qu'Aurora reprend brièvement l'instance dans le cadre du mécanisme de suppression.
- problème de communication transitoire : si Aurora ne parvient pas à déterminer si le paramètre de capacité minimale correspond à 0 ACU dans la configuration de la mise à l'échelle, il ne met pas l'instance en pause. Ce type de scénario est sensé être très rare.

Performances et mise à l'échelle pour Aurora Serverless v2

Les procédures et exemples suivants montrent comment définir la plage de capacité pour les clusters Aurora Serverless v2 et leurs instances de base de données associées. Vous pouvez également utiliser les procédures suivantes pour surveiller le niveau d'occupation de vos instances de base de données. Vous pouvez ensuite utiliser vos résultats pour déterminer si vous devez augmenter ou réduire la plage de capacité.

Avant d'utiliser ces procédures, assurez-vous de bien savoir comment fonctionne la mise à l'échelle d'Aurora Serverless v2. Le mécanisme de mise à l'échelle est différent de celui d'Aurora Serverless v1. Pour en savoir plus, consultez Mise à l'échelle d'Aurora Serverless v2.

Table des matières

Choix de la plage de capacité Aurora Serverless v2 pour un cluster Aurora

Avec les instances de base de données Aurora Serverless v2, vous définissez la plage de capacité qui s'applique à toutes les instances de base de données de votre cluster de bases de données en même temps que vous ajoutez la première instance de base de données Aurora Serverless v2 au cluster de bases de données Pour savoir comment procéder, consultez Définition de la plage de capacité Aurora Serverless v2 d'un cluster.

Vous pouvez également modifier la plage de capacité d'un cluster existant. Les sections suivantes expliquent plus en détail comment choisir les valeurs minimales et maximales appropriées et ce qui se passe lorsque vous modifiez la plage de capacité. Par exemple, la modification de la plage de capacité peut modifier les valeurs par défaut de certains paramètres de configuration. L'application de toutes les modifications des paramètres peut exiger de redémarrer chaque instance de base de données Aurora Serverless v2.

Rubriques

- Exemple : Modification de la plage de capacité Aurora Serverless v2 d'un cluster Aurora PostgreSQL

Choix de la valeur minimale de capacité Aurora Serverless v2 pour un cluster

Il peut s'avérer tentant de toujours choisir 0,5 comme valeur minimale de capacité Aurora Serverless v2. Cette valeur permet à l'instance de base de données de procéder réduire verticalement sa capacité à sa valeur la plus faible lorsqu'elle est complètement inactive, tout en restant active. Vous pouvez également activer le comportement de pause automatique en spécifiant une capacité minimale de 0 ACU, comme expliqué dans Réduction verticale à zéro ACU avec pause et reprise automatiques pour Aurora Serverless v2. Toutefois, selon la façon dont vous utilisez ce cluster et les autres paramètres que vous configurez, une autre capacité minimale peut s'avérer plus efficace. Tenez compte des facteurs suivants lors du choix de la valeur minimale de capacité :

- Le taux de mise à l'échelle d'une instance de base de données Aurora Serverless v2 dépend de sa capacité actuelle. Plus sa capacité actuelle est élevée, plus son augmentation d'échelle est rapide. Si vous avez besoin d'augmenter rapidement l'échelle de l'instance de base de données jusqu'à une capacité très élevée, envisagez de définir la capacité minimale sur une valeur où le taux de mise à l'échelle satisfait à vos exigences.
- Si vous modifiez généralement la classe d'instance de base de données de vos instances de base de données en prévision d'une charge de travail particulièrement élevée ou faible, vous pouvez utiliser cette expérience pour effectuer une estimation approximative de la plage de capacité Aurora Serverless v2 équivalente. Pour déterminer la taille de mémoire à utiliser en période de faible trafic, consultez Spécifications matérielles pour les classes d'instance de base de données pour Aurora.

  Par exemple, supposons que vous utilisiez la classe d'instance de base de données db.r6g.xlarge lorsque la charge de travail de votre cluster est faible. Cette classe d'instance de base de données dispose de 32 Gio de mémoire. Vous pouvez donc spécifier un nombre minimal d'unités de capacité Aurora (ACU) de 16 pour configurer une instance de base de données Aurora Serverless v2 pouvant faire l'objet d'une réduction d'échelle à cette même capacité environ. En effet, chaque ACU correspond à environ 2 Gio de mémoire. Vous pouvez spécifier une valeur légèrement inférieure pour prolonger la réduction d'échelle de l'instance de base de données au cas où votre instance de base de données db.r6g.xlarge soit parfois sous-exploitée.
- Si votre application fonctionne le plus efficacement lorsque les instances de base de données contiennent une certaine quantité de données dans le cache de mémoire tampon, envisagez de spécifier un nombre d'ACU minimal pour lequel la mémoire est suffisamment volumineuse pour

contenir les données fréquemment consultées. Sinon, certaines données sont expulsées du cache de mémoire tampon lorsque les instances de base de données Aurora Serverless v2 font l'objet d'une réduction d'échelle jusqu'à une taille de mémoire inférieure. Ensuite, lorsque les instances de base de données font à nouveau l'objet d'une augmentation d'échelle, les informations sont relues dans le cache de mémoire tampon au fil du temps. Si la quantité d'E/S nécessaire pour remettre les données dans le cache de mémoire tampon est importante, il peut être plus efficace de choisir un nombre minimal d'ACU plus élevé.

- Si vos instances de base de données Aurora Serverless v2 s'exécutent la plupart du temps à une capacité particulière, envisagez de spécifier une valeur minimale de capacité inférieure à cette valeur de référence, sans la définir sur une valeur trop faible. Les instances de base de données Aurora Serverless v2 peuvent estimer le plus efficacement dans quelle mesure et avec quelle rapidité procéder à l'augmentation d'échelle lorsque la capacité actuelle n'est pas nettement inférieure à la capacité requise.
- Si votre charge de travail approvisionnée présente des exigences en mémoire trop élevées pour les petites classes d'instance de base de données telles que T3 ou T4g, choisissez un nombre minimal d'ACU qui fournit une quantité de mémoire comparable à une instance de base de données R5 ou R6g.

  Nous recommandons particulièrement d'utiliser la capacité minimale suivante avec les fonctionnalités spécifiées (ces recommandations peuvent être modifiées) :

  - Performance Insights : 2 ACU
  - Bases de données globales Aurora : 8 ACU (s'applique uniquement à la Région AWS principale)
- Dans Aurora, la réplication s'effectue au niveau de la couche de stockage, de sorte que la capacité du lecteur n'affecte pas directement la réplication. Toutefois, pour les instances de base de données de lecteur Aurora Serverless v2 qui se mettent à l'échelle indépendamment, assurez-vous que la capacité minimale est suffisante pour gérer les charges de travail pendant les périodes d'écriture intensive afin d'éviter la latence des requêtes. Si les instances de base de données de lecteur aux niveaux de promotion 2 à 15 rencontrent des problèmes de performance, envisagez d'augmenter la capacité minimale du cluster. Pour savoir comment choisir si les instances de base de données de lecteur sont mises à l'échelle en même temps que l'enregistreur ou indépendamment, consultez Choix du niveau de promotion pour un lecteur Aurora Serverless v2.
- Si vous avez un cluster de bases de données avec des instances de base de données de lecteur Aurora Serverless v2, les lecteurs ne se mettent pas à l'échelle en même temps que l'instance de base de données d'enregistreur lorsque le niveau de promotion des lecteurs n'est pas de 0 ou 1. Dans ce cas, la définition d'une valeur minimale de capacité faible peut entraîner un retard de réplication excessif. En effet, la capacité des lecteurs peut ne pas être suffisante pour

appliquer les modifications depuis l'enregistreur lorsque la base de données est occupée. Nous vous recommandons de définir la capacité minimale sur une valeur qui représente une quantité de mémoire et de processeur comparable à celle de l'instance de base de données d'enregistreur.

- La valeur du paramètre `max_connections` pour les instances de base de données Aurora Serverless v2 est basée sur la taille de la mémoire dérivée du nombre maximal d'unités ACU. Toutefois, quand vous spécifiez une capacité minimale de 0 ou 0,5 ACU sur les instances de base de données compatibles avec PostgreSQL, la valeur maximale de `max_connections` est limitée à 2 000.

  Si vous avez l'intention d'utiliser le cluster Aurora PostgreSQL pour une charge de travail à connexion élevée, envisagez d'utiliser un nombre minimal d'ACU de 1 ou plus. Pour plus de détails sur la façon dont Aurora Serverless v2 gère le paramètre de configuration `max_connections`, consultez Nombre maximal de connexions pour Aurora Serverless v2.
- La durée nécessaire à une instance de base de données Aurora Serverless v2 pour être mise à l'échelle de sa capacité minimale à sa capacité maximale dépend de la différence entre ses valeurs d'ACU minimale et maximale. Lorsque la capacité actuelle de l'instance de base de données est élevée, Aurora Serverless v2 fait l'objet d'une augmentation d'échelle par incréments plus importants que lorsque l'instance de base de données part d'une faible capacité. Par conséquent, si vous spécifiez une capacité maximale relativement élevée et que l'instance de base de données se rapproche la plupart du temps de cette capacité, envisagez d'augmenter le nombre minimal d'ACU. De cette façon, une instance de base de données inactive peut rétablir sa capacité maximale plus rapidement.

Choix de la valeur maximale de capacité Aurora Serverless v2 pour un cluster

Il peut s'avérer tentant de toujours choisir une valeur élevée pour la valeur maximale de capacité Aurora Serverless v2. Une capacité maximale élevée permet à l'instance de base de données de faire l'objet d'une augmentation d'échelle maximale lorsqu'elle exécute une charge de travail exigeant beaucoup de ressources. Une valeur faible évite la possibilité de frais inattendus. Selon votre utilisation de ce cluster et des autres paramètres que vous configurez, la valeur la plus efficace peut être supérieure ou inférieure à celle à laquelle vous pensiez initialement. Tenez compte des facteurs suivants lors du choix de la valeur maximale de capacité :

- La capacité maximale doit être au moins aussi élevée que la capacité minimale. Vous pouvez définir la capacité minimale et la capacité maximale pour qu'elles soient identiques. Cependant, dans ce cas, la capacité ne fait jamais l'objet d'une augmentation ou d'une réduction d'échelle. Par

conséquent, l'utilisation de valeurs identiques pour les capacités minimale et maximale n'est pas appropriée en dehors des situations de test.

- La capacité maximale doit être supérieure à 0,5 ACU. Vous pouvez définir la capacité minimale et la capacité maximale pour qu'elles soient identiques dans la plupart des cas. Toutefois, vous ne pouvez pas spécifier 0,5 à la fois pour les capacités minimale et maximale. Utilisez une valeur supérieure ou égale à 1 pour la capacité maximale.
- Si vous modifiez généralement la classe d'instance de base de données de vos instances de base de données en prévision d'une charge de travail particulièrement élevée ou faible, vous pouvez utiliser cette expérience pour estimer la plage de capacité Aurora Serverless v2 équivalente. Pour déterminer la taille de mémoire à utiliser en période de trafic élevé, consultez Spécifications matérielles pour les classes d'instance de base de données pour Aurora.

  Par exemple, supposons que vous utilisiez la classe d'instance de base de données db.r6g.4xlarge lorsque la charge de travail de votre cluster est élevée. Cette classe d'instance de base de données dispose de 128 Gio de mémoire. Vous pouvez donc spécifier un nombre maximal d'ACU de 64 pour configurer une instance de base de données Aurora Serverless v2 pouvant faire l'objet d'une augmentation d'échelle à cette même capacité environ. En effet, chaque ACU correspond à environ 2 Gio de mémoire. Vous pouvez spécifier une valeur légèrement supérieure pour prolonger l'augmentation d'échelle de l'instance de base de données au cas où votre instance de base de données db.r6g.4xlarge manque parfois de capacité pour gérer efficacement la charge de travail.
- Si votre budget d'utilisation de la base de données est plafonné, choisissez une valeur inférieure à ce plafond, même si toutes vos instances de base de données Aurora Serverless v2 s'exécutent en permanence à leur capacité maximale. N'oubliez pas que lorsque votre cluster comporte n instances de base de données Aurora Serverless v2, la capacité maximale théorique pour Aurora Serverless v2 que le cluster peut consommer à tout moment correspond à n fois le nombre maximal d'ACU pour le cluster. (La quantité réelle consommée peut être inférieure, par exemple si certains lecteurs sont mis à l'échelle indépendamment de l'enregistreur.)
- Si vous utilisez des instances de base de données de lecteur Aurora Serverless v2 pour décharger une partie de la charge de travail en lecture seule de l'instance de base de données d'enregistreur, vous pourrez peut-être choisir une valeur maximale de capacité inférieure. Cela permet de refléter que chaque instance de base de données de lecteur n'a pas besoin de faire l'objet d'une mise à l'échelle aussi importante que si le cluster ne contenait qu'une seule instance de base de données.
- Supposons que vous souhaitiez vous protéger contre une utilisation excessive due à une mauvaise configuration des paramètres de base de données ou à l'inefficacité des requêtes de votre application. Dans ce cas, vous pouvez éviter une surutilisation accidentelle en choisissant une valeur maximale de capacité inférieure à la valeur absolue la plus élevée que vous pourriez définir.

- Si les pics dus à l'activité réelle de l'utilisateur sont rares mais existent, vous pouvez les prendre en compte lorsque vous choisissez la valeur maximale de capacité. Si la priorité est que l'application continue de s'exécuter à des performances et une capacité de mise à l'échelle optimales, vous pouvez spécifier une valeur maximale de capacité supérieure à celle que vous constatez dans le cas d'une utilisation normale. S'il est admis que l'application s'exécute à un débit réduit pendant les pics d'activité très élevés, vous pouvez choisir une valeur maximale de capacité légèrement inférieure. Assurez-vous de choisir une valeur dont les ressources de mémoire et de processeur sont suffisantes pour maintenir l'application en cours d'exécution.
- Si vous activez les paramètres de votre cluster qui augmentent l'utilisation de la mémoire pour chaque instance de base de données, tenez compte de cette mémoire lorsque vous choisissez le nombre maximal d'ACU. Ces paramètres incluent les paramètres de Performance Insights, les requêtes parallèles Aurora MySQL, le schéma de performances Aurora MySQL et la réplication des journaux binaires Aurora MySQL. Assurez-vous que le nombre maximal d'ACU permet aux instances de base de données Aurora Serverless v2 de faire l'objet d'une augmentation d'échelle suffisante pour gérer la charge de travail lorsque ces fonctionnalités sont utilisées. Pour plus d'informations sur le dépannage des problèmes provoqués par la combinaison d'un nombre maximal d'ACU faible et de fonctionnalités Aurora qui imposent une surcharge de mémoire, consultez Éviter les erreurs de mémoire insuffisante.

Exemple : Modification de la plage de capacité Aurora Serverless v2 d'un cluster Aurora MySQL

L'exemple AWS CLI suivant montre comment mettre à jour la plage d'ACU pour les instances de base de données Aurora Serverless v2 d'un cluster Aurora MySQL existant. Initialement, la plage de capacité pour le cluster est de 8 à 32 ACU.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 8.0,
    "MaxCapacity": 32.0
}
```

L'instance de base de données est inactive et son échelle est réduite à 8 ACU. Les paramètres de capacité suivants s'appliquent à l'instance de base de données à ce stade. Pour représenter la taille du groupe de mémoires tampons en unités facilement lisibles, nous la divisons par 2 puissance 30,

ce qui donne une mesure en gibioctets (Gio). En effet, les mesures liées à la mémoire pour Aurora utilisent des unités basées sur des puissances de 2 et non des puissances de 10.

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              3000 |
+-------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|                9294577664 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+-----------+
| gibibytes |
+-----------+
|   8.65625 |
+-----------+
1 row in set (0.00 sec)
```

Ensuite, nous modifions la plage de capacité du cluster. Une fois la commande `modify-db-cluster` terminée, la plage d'ACU du cluster est comprise entre 12,5 et 80.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=12.5,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 12.5,
    "MaxCapacity": 80.0
}
```

La modification de la plage de capacité a modifié les valeurs par défaut de certains paramètres de configuration. Aurora peut appliquer immédiatement certaines de ces nouvelles valeurs par défaut.

Cependant, certaines modifications de paramètre ne prennent effet qu'après un redémarrage. Le statut `pending-reboot` indique qu'un redémarrage est nécessaire pour appliquer certaines modifications de paramètre.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "pending-reboot"
        }
    ]
}
```

À ce stade, le cluster est inactif et l'instance de base de données `serverless-v2-instance-1` consomme 12,5 ACU. Le paramètre `innodb_buffer_pool_size` est déjà ajusté en fonction de la capacité actuelle de l'instance de base de données. Le paramètre `max_connections` reflète toujours la valeur de l'ancienne capacité maximale. La réinitialisation de cette valeur exige de redémarrer l'instance de base de données.

> **Note**
>
> Si vous définissez le paramètre `max_connections` directement dans un groupe de paramètres de base de données personnalisé, aucun redémarrage n'est nécessaire.

```
mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              3000 |
+-------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
```

```
|               15572402176 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+---------------+
| gibibytes     |
+---------------+
| 14.5029296875 |
+---------------+
1 row in set (0.00 sec)
```

À présent, nous redémarrons l'instance de base de données et nous attendons qu'elle soit de nouveau disponible.

```
aws rds reboot-db-instance --db-instance-identifier serverless-v2-instance-1
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBInstanceStatus": "rebooting"
}

aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
```

L'état `pending-reboot` est effacé. La valeur `in-sync` confirme qu'Aurora a appliqué l'ensemble des modifications de paramètre en attente.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "in-sync"
        }
    ]
}
```

Le paramètre `innodb_buffer_pool_size` a augmenté jusqu'à sa taille finale pour une instance de base de données inactive. Le paramètre `max_connections` a augmenté pour refléter une valeur

dérivée du nombre maximal d'ACU. La formule utilisée par Aurora pour `max_connections` entraîne une augmentation de 1 000 lorsque la taille de mémoire double.

```
mysql> select @@innodb_buffer_pool_size;
+---------------------------+
| @@innodb_buffer_pool_size |
+---------------------------+
|               16139681792 |
+---------------------------+
1 row in set (0.00 sec)

mysql> select @@innodb_buffer_pool_size / pow(2,30) as gibibytes;
+-----------+
| gibibytes |
+-----------+
|  15.03125 |
+-----------+
1 row in set (0.00 sec)

mysql> select @@max_connections;
+-------------------+
| @@max_connections |
+-------------------+
|              4000 |
+-------------------+
1 row in set (0.00 sec)
```

Nous fixons la plage de capacité à 0,5-128 ACU, et redémarrons l'instance de base de données. À présent, l'instance de base de données inactive a une taille de cache de mémoire tampon inférieure à 1 Gio. Nous la mesurons donc en mébioctets (Mio). La valeur 5 000 de `max_connections` est dérivée de la taille de mémoire du paramètre de capacité maximale.

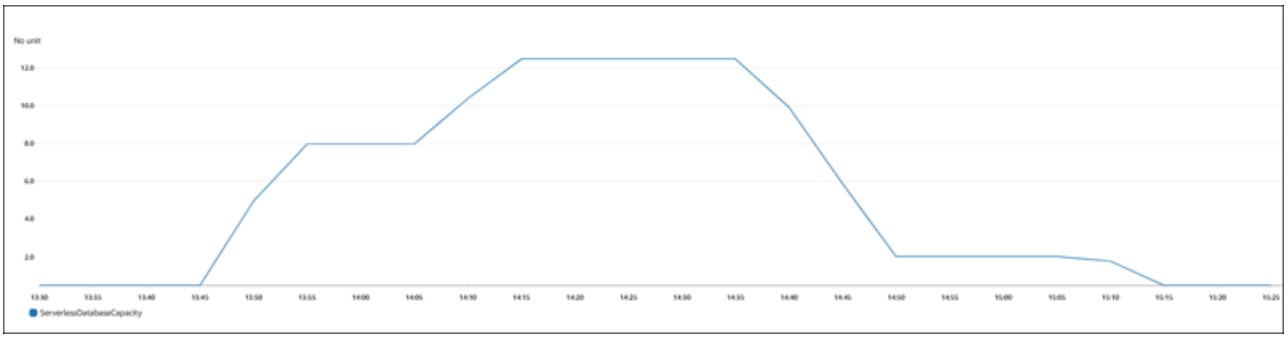
```
mysql> select @@innodb_buffer_pool_size / pow(2,20) as mebibytes, @@max_connections;
+-----------+-------------------+
| mebibytes | @@max_connections |
+-----------+-------------------+
|       672 |              5000 |
+-----------+-------------------+
1 row in set (0.00 sec)
```

Exemple : Modification de la plage de capacité Aurora Serverless v2 d'un cluster Aurora PostgreSQL

Les exemples d'interface de ligne de commande suivants montrent comment mettre à jour la plage d'ACU pour les instances de base de données Aurora Serverless v2 d'un cluster Aurora PostgreSQL existant.

1. La plage de capacité du cluster commence entre 0,5 et 1 ACU.
2. Modifiez la plage de capacité entre 8 et 32 ACU.
3. Modifiez la plage de capacité entre 12,5 et 80 ACU.
4. Modifiez la plage de capacité entre 0,5 et 128 ACU.
5. Ramenez la capacité à sa plage initiale de 0,5 à 1 ACU.

La figure suivante montre les changements de capacité dans Amazon CloudWatch.



L'instance de base de données est inactive et réduite à 0,5 ACU. Les paramètres de capacité suivants s'appliquent à l'instance de base de données à ce stade.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

Ensuite, nous modifions la plage de capacité du cluster. Une fois la commande `modify-db-cluster` terminée, la plage ACU pour le cluster est de 8 à 32.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 8.0,
    "MaxCapacity": 32.0
}
```

La modification de la plage de capacité modifie les valeurs par défaut de certains paramètres de configuration. Aurora peut appliquer immédiatement certaines de ces nouvelles valeurs par défaut. Cependant, certaines modifications de paramètre ne prennent effet qu'après un redémarrage. Le statut `pending-reboot` indique qu'un redémarrage est nécessaire pour appliquer certaines modifications de paramètre.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "pending-reboot"
        }
    ]
}
```

À ce stade, le cluster est inactif et l'instance de base de données `serverless-v2-instance-1` consomme 8 ACU. Le paramètre `shared_buffers` est déjà ajusté en fonction de la capacité actuelle de l'instance de base de données. Le paramètre `max_connections` reflète toujours la valeur de l'ancienne capacité maximale. La réinitialisation de cette valeur exige de redémarrer l'instance de base de données.

> **Note**
>
> Si vous définissez le paramètre `max_connections` directement dans un groupe de paramètres de base de données personnalisé, aucun redémarrage n'est nécessaire.

```
postgres=> show max_connections;
 max_connections
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 1425408
(1 row)
```

Nous redémarrons l'instance de base de données et nous attendons qu'elle soit de nouveau disponible.

```
aws rds reboot-db-instance --db-instance-identifier serverless-v2-instance-1
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
  "DBInstanceStatus": "rebooting"
}

aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
```

Maintenant que l'instance de base de données est redémarrée, le statut `pending-reboot` est effacé. La valeur `in-sync` confirme qu'Aurora a appliqué l'ensemble des modifications de paramètre en attente.

```
aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query '*[].{DBClusterMembers:DBClusterMembers[*].
{DBInstanceIdentifier:DBInstanceIdentifier,DBClusterParameterGroupStatus:DBClusterParameterGrou
[0]'
{
    "DBClusterMembers": [
        {
            "DBInstanceIdentifier": "serverless-v2-instance-1",
            "DBClusterParameterGroupStatus": "in-sync"
        }
    ]
}
```

Après le redémarrage, `max_connections` indique la valeur de la nouvelle capacité maximale.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 1425408
(1 row)
```

Ensuite, nous modifions la plage de capacité du cluster entre 12,5 et 80 ACU.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=12.5,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 12.5,
    "MaxCapacity": 80.0
}
```

À ce stade, le cluster est inactif et l'instance de base de données `serverless-v2-instance-1` consomme 12,5 ACU. Le paramètre `shared_buffers` est déjà ajusté en fonction de la capacité actuelle de l'instance de base de données. La valeur `max_connections` est toujours de 5 000.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 2211840
(1 row)
```

Nous redémarrons à nouveau, mais les valeurs des paramètres restent les mêmes. C'est parce que `max_connections` présente une valeur maximale de 5 000 pour un cluster de bases de données Aurora Serverless v2 exécutant Aurora PostgreSQL.

```
postgres=> show max_connections;
 max_connections
-----------------
 5000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 2211840
(1 row)
```

À présent, nous définissons la plage de capacité entre 0,5 et 128 ACU. Le cluster de bases de données passe à 10 ACU, puis à 2. Nous redémarrons l'instance de base de données.

```
postgres=> show max_connections;
 max_connections
-----------------
 2000
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

La valeur `max_connections` pour les instances de base de données Aurora Serverless v2 est basée sur la taille de la mémoire dérivée du nombre maximal d'unités ACU. Toutefois, quand vous spécifiez une capacité minimale de 0 ou 0,5 ACU sur les instances de base de données compatibles avec PostgreSQL, la valeur maximale de `max_connections` est limitée à 2 000.

Maintenant, nous remettons la capacité à sa plage initiale de 0,5 0 1 ACU et nous redémarrons l'instance de base de données. Le paramètre `max_connections` a retrouvé sa valeur d'origine.

```
postgres=> show max_connections;
 max_connections
```

```
-----------------
 189
(1 row)

postgres=> show shared_buffers;
 shared_buffers
----------------
 16384
(1 row)
```

Utilisation des groupes de paramètres pour Aurora Serverless v2

Lorsque vous créez votre cluster de bases de données Aurora Serverless v2, vous choisissez un moteur de bases de données Aurora spécifique et un groupe de paramètres de cluster de bases de données associé. Si vous n'êtes pas familier avec la façon dont Aurora utilise les groupes de paramètres pour appliquer les paramètres de configuration de manière cohérente entre les clusters, consultez Groupes de paramètres pour Amazon Aurora. Toutes ces procédures de création, de modification, d'application et d'autres actions pour les groupes de paramètres s'appliquent à Aurora Serverless v2.

La fonction de groupe de paramètres fonctionne généralement de la même manière entre les clusters approvisionnés et les clusters contenant des instances de base de données Aurora Serverless v2 :

- Les valeurs de paramètre par défaut pour l'ensemble des instances de base de données du cluster sont définies par le groupe de paramètres du cluster.
- Vous pouvez remplacer certains paramètres pour des instances de base de données spécifiques en spécifiant un groupe de paramètres de base de données personnalisé pour ces instances de base de données. Vous pouvez le faire pendant le débogage ou le réglage des performances pour certaines instances de base de données. Par exemple, supposons que vous disposez d'un cluster contenant des instances de base de données Aurora Serverless v2 et des instances de base de données approvisionnées. Dans ce cas, vous pouvez spécifier des paramètres différents pour les instances de base de données approvisionnées à l'aide d'un groupe de paramètres de base de données personnalisé.
- Pour Aurora Serverless v2, vous pouvez utiliser tous les paramètres ayant la valeur `provisioned` dans l'attribut `SupportedEngineModes` du groupe de paramètres. Dans Aurora Serverless v1, vous ne pouvez utiliser que le sous-ensemble de paramètres ayant `serverless` dans l'attribut `SupportedEngineModes`.

Rubriques

Valeurs des paramètres par défaut

La différence cruciale entre les instances de base de données approvisionnées et les instances de base de données Aurora Serverless v2 réside dans le fait qu'Aurora remplace toutes les valeurs de paramètre personnalisées pour certains paramètres liés à la capacité d'instance de base de données. Les valeurs de paramètre personnalisées s'appliquent toujours à l'ensemble des instances de base de données approvisionnées de votre cluster. Pour en savoir plus sur la façon dont les instances de base de données Aurora Serverless v2 interprètent les paramètres des groupes de paramètres Aurora, consultez Paramètres de configuration des clusters Aurora. Pour obtenir les paramètres qui sont remplacés par Aurora Serverless v2, consultez Paramètres ajustés par Aurora en fonction de l'augmentation et de la réduction d'échelle d'Aurora Serverless v2 et Paramètres calculés par Aurora en fonction de la capacité maximale d'Aurora Serverless v2.

Vous pouvez obtenir une liste de valeurs par défaut pour les groupes de paramètres par défaut pour les différents moteurs de bases de données Aurora en utilisant la commande CLI describe-db-cluster-parameters et en interrogeant la Région AWS. Les valeurs suivantes peuvent être utilisées pour les options `--db-parameter-group-family` et `-db-parameter-group-name` pour les versions de moteur compatibles avec Aurora Serverless v2.

Moteur de base de données et version	Famille du groupe de paramètres	Nom du groupe de paramètres par défaut
Aurora MySQL version 3	`aurora-mysql8.0`	`default.aurora-mysql8.0`
Aurora PostgreSQL version 13.x	`aurora-postgresql13`	`default.aurora-postgresql13`

Moteur de base de données et version	Famille du groupe de paramètres	Nom du groupe de paramètres par défaut
Aurora PostgreSQL version 14.x	`aurora-postgresql14`	`default.aurora-postgresql14`
Aurora PostgreSQL version 15.x	`aurora-postgresql15`	`default.aurora-postgresql15`
Aurora PostgreSQL version 16.x	`aurora-postgresql16`	`default.aurora-postgresql16`
Aurora PostgreSQL version 17.x	`aurora-postgresql17`	`default.aurora-postgresql17`

L'exemple suivant illustre l'obtention d'une liste de paramètres depuis le groupe de clusters de bases de données par défaut pour Aurora MySQL version 3 et Aurora PostgreSQL version 13. Ce sont les versions d'Aurora MySQL et d'Aurora PostgreSQL que vous utilisez avec Aurora Serverless v2.

Pour Linux, macOS ou Unix :

```
aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-mysql8.0 \
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' \
  --output text

aws rds describe-db-cluster-parameters \
  --db-cluster-parameter-group-name default.aurora-postgresql13 \
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' \
  --output text
```

Pour Windows :

```
aws rds describe-db-cluster-parameters ^
  --db-cluster-parameter-group-name default.aurora-mysql8.0 ^
```

```
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' ^
  --output text

aws rds describe-db-cluster-parameters ^
  --db-cluster-parameter-group-name default.aurora-postgresql13 ^
  --query 'Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} |
    [?contains(SupportedEngineModes, `provisioned`) == `true`] | [*].[ParameterName]' ^
  --output text
```

Nombre maximal de connexions pour Aurora Serverless v2

Pour Aurora MySQL et Aurora PostgreSQL, les instances de base de données Aurora Serverless v2 maintiennent une valeur constante pour le paramètre `max_connections` afin que les connexions ne soient pas interrompues lorsque l'instance de base de données fait l'objet d'une réduction d'échelle. La valeur par défaut de ce paramètre est dérivée d'une formule basée sur la taille de la mémoire de l'instance de base de données. Pour plus d'informations sur la formule et les valeurs par défaut des classes d'instance de base de données approvisionnée, consultez Nombre maximal de connexions à une instance de base de données Aurora MySQL et Nombre maximal de connexions à une instance de base de données Aurora PostgreSQL.

Lorsque Aurora Serverless v2 évalue la formule, il utilise la taille de mémoire en fonction du nombre maximal d'unités de capacité Aurora (ACU) de l'instance de base de données, et non la valeur d'ACU actuelle. Si vous modifiez la valeur par défaut, nous vous recommandons d'utiliser une variation de la formule plutôt que de spécifier une valeur constante. De cette façon, Aurora Serverless v2 peut utiliser un réglage approprié en fonction de la capacité maximale.

Lorsque vous modifiez la capacité maximale d'un cluster de bases de données Aurora Serverless v2, vous devez redémarrer les instances de base de données Aurora Serverless v2 pour mettre à jour la valeur `max_connections`. Cela est dû au fait que `max_connections` est un paramètre statique pour Aurora Serverless v2.

Le tableau suivant indique les valeurs par défaut de `max_connections` pour Aurora Serverless v2 en fonction de la valeur maximale d'ACU.

Nombre maximal d'ACU	Nombre maximal de connexions par défaut sur Aurora MySQL	Nombre maximal de connexions par défaut sur Aurora PostgreSQL
1	90	189
4	135	823
8	1 000	1 669
16	2 000	3 360
32	3 000	5 000
64	4 000	5 000
128	5 000	5 000
192	6 000	5 000
256	6 000	5 000

> **Note**
>
> La valeur `max_connections` pour les instances de base de données Aurora Serverless v2 est basée sur la taille de la mémoire dérivée du nombre maximal d'unités ACU. Toutefois, quand vous spécifiez une capacité minimale de 0 ou 0,5 ACU sur les instances de base de données compatibles avec PostgreSQL, la valeur maximale de `max_connections` est limitée à 2 000.

Pour obtenir des exemples spécifiques montrant comment `max_connections` évolue avec la valeur maximale d'ACU, consultez Exemple : Modification de la plage de capacité Aurora Serverless v2 d'un cluster Aurora MySQL et Exemple : Modification de la plage de capacité Aurora Serverless v2 d'un cluster Aurora PostgreSQL.

Paramètres ajustés par Aurora en fonction de l’augmentation et de la réduction d’échelle d’Aurora Serverless v2

Lors de la mise à l’échelle automatique, Aurora Serverless v2 doit être en mesure de modifier les paramètres pour que chaque instance de base de données fonctionne mieux en fonction de l’augmentation ou de la diminution de la capacité. Par conséquent, vous ne pouvez pas remplacer certains paramètres liés à la capacité. Pour les paramètres que vous pouvez remplacer, évitez de coder en dur les valeurs fixes. Les remarques suivantes s’appliquent aux paramètres liés à la capacité.

Pour Aurora MySQL, Aurora Serverless v2 redimensionne certains paramètres dynamiquement pendant la mise à l’échelle. Pour les paramètres suivants, Aurora Serverless v2 n’utilise aucune valeur de paramètre personnalisée que vous spécifiez :

- `innodb_buffer_pool_size`
- `innodb_purge_threads`
- `table_definition_cache`
- `table_open_cache`

Pour Aurora PostgreSQL, Aurora Serverless v2 redimensionne dynamiquement le paramètre suivant pendant la mise à l’échelle. Pour les paramètres suivants, Aurora Serverless v2 n’utilise aucune valeur de paramètre personnalisée que vous spécifiez :

- `shared_buffers`

Pour tous les paramètres autres que ceux énumérés ici, les instances de base de données Aurora Serverless v2 fonctionnent de la même manière que les instances de base de données provisionnées. La valeur de paramètre par défaut est héritée du groupe de paramètres de cluster. Vous pouvez modifier la valeur par défaut pour l’ensemble du cluster à l’aide d’un groupe de paramètres de cluster personnalisé. Sinon, vous pouvez modifier la valeur par défaut de certaines instances de base de données à l’aide d’un groupe de paramètres de base de données personnalisé. Les paramètres dynamiques sont immédiatement mis à jour. Les modifications apportées aux paramètres statiques ne prennent effet qu’après le redémarrage de l’instance de base de données.

Paramètres calculés par Aurora en fonction de la capacité maximale d'Aurora Serverless v2

Pour les paramètres suivants, Aurora PostgreSQL utilise des valeurs par défaut dérivées de la taille de mémoire basée sur le nombre maximal d'ACU, comme avec `max_connections` :

- `autovacuum_max_workers`
- `autovacuum_vacuum_cost_limit`
- `autovacuum_work_mem`
- `effective_cache_size`
- `maintenance_work_mem`

Éviter les erreurs de mémoire insuffisante

Si l'une de vos instances de base de données Aurora Serverless v2 atteint systématiquement la limite de sa capacité maximale, Aurora indique cette condition en définissant le statut de l'instance de base de données sur `incompatible-parameters`. Lorsque l'instance de base de données a le statut `incompatible-parameters`, certaines opérations sont bloquées. Par exemple, vous ne pouvez pas mettre à niveau la version du moteur.

En règle générale, votre instance de base de données a ce statut lorsqu'elle redémarre fréquemment en raison d'erreurs de mémoire insuffisante. Aurora enregistre un événement lorsque ce type de redémarrage se produit. Vous pouvez afficher l'événement en suivant la procédure de la rubrique Affichage d'événements Amazon RDS. Une utilisation exceptionnellement élevée de la mémoire peut se produire en raison de la surcharge provoquée par l'activation de paramètres tels que Performance Insights et l'authentification IAM. Elle peut également se produire en raison d'une charge de travail importante sur votre instance de base de données ou de la gestion des métadonnées associées à un grand nombre d'objets de schéma.

Si la pression de la mémoire diminue de sorte que l'instance de base de données n'atteint pas très souvent sa capacité maximale, Aurora rétablit automatiquement le statut `available` de l'instance de base de données.

Pour vous remettre de cet état, vous pouvez prendre tout ou partie des mesures suivantes :

- Augmentez la limite inférieure de capacité pour les instances de base de données Aurora Serverless v2 en modifiant le nombre minimal d'ACU pour le cluster. Cela évite les situations

problématiques où une base de données inactive fait l'objet d'une réduction d'échelle jusqu'à une capacité où la mémoire est inférieure à la mémoire nécessaire pour les fonctionnalités activées dans votre cluster. Après avoir modifié les paramètres d'ACU du cluster, redémarrez l'instance de base de données Aurora Serverless v2. Cela permet d'évaluer si Aurora peut réinitialiser le statut sur `available`.

- Augmentez la limite supérieure de capacité pour les instances de base de données Aurora Serverless v2 en modifiant le nombre maximal d'ACU pour le cluster. Cela évite les situations problématiques où une base de données occupée ne peut pas faire l'objet d'une augmentation d'échelle jusqu'à une capacité où la mémoire est suffisante pour les fonctionnalités activées dans votre cluster et pour la charge de travail de la base de données. Après avoir modifié les paramètres d'ACU du cluster, redémarrez l'instance de base de données Aurora Serverless v2. Cela permet d'évaluer si Aurora peut réinitialiser le statut sur `available`.
- Désactivez les paramètres de configuration exigeant une surcharge de mémoire. Par exemple, supposons que vos fonctionnalités Gestion des identités et des accès AWS(IAM), Performance Insights ou de réplication des journaux binaires Aurora MySQL soient activées mais que vous ne les utilisez pas. Si c'est le cas, vous pouvez les désactiver. Vous pouvez également augmenter les valeurs de capacité minimale et maximale du cluster pour tenir compte de la mémoire utilisée par ces fonctionnalités. Pour obtenir des directives sur le choix des valeurs de capacité minimale et maximale, consultez Choix de la plage de capacité Aurora Serverless v2 pour un cluster Aurora.
- Réduisez la charge de travail sur l'instance de base de données. Par exemple, vous pouvez ajouter des instances de base de données de lecteur au cluster afin de répartir la charge issue des requêtes en lecture seule sur d'autres instances de base de données.
- Réglez le code SQL utilisé par votre application pour utiliser moins de ressources. Par exemple, vous pouvez examiner vos plans de requête, vérifier le journal des requêtes lentes ou ajuster les index de vos tables. Vous pouvez également effectuer d'autres types de réglages SQL traditionnels.

Métriques Amazon CloudWatch importantes pour Aurora Serverless v2

Pour commencer à utiliser Amazon CloudWatch pour votre instance de base de données Aurora Serverless v2, consultez Affichage de journaux Aurora Serverless v2 dans Amazon CloudWatch. Pour en savoir plus sur la surveillance des clusters de bases de données Aurora via CloudWatch, consultez Surveillance des événements de journaux dans Amazon CloudWatch.

Vous pouvez consulter vos instances de base de données Aurora Serverless v2 dans CloudWatch pour surveiller la capacité consommée par chaque instance de base de données avec la métrique

`ServerlessDatabaseCapacity`. Vous pouvez également surveiller toutes les métriques Aurora CloudWatch standard, comme `DatabaseConnections` et `Queries`. Pour obtenir la liste complète des métriques CloudWatch que vous pouvez surveiller pour Aurora, consultez CloudWatch Métriques Amazon pour Amazon Aurora. Les métriques, de niveau cluster et de niveau instance, sont décrites aux rubriques Métriques de niveau cluster pour Amazon Aurora et Métriques de niveau instance pour Amazon Aurora.

Il est important de surveiller les métriques CloudWatch de niveau instance suivantes pour comprendre comment vos instances de base de données Aurora Serverless v2 font l'objet d'une augmentation et d'une réduction d'échelle. Toutes ces métriques sont calculées toutes les secondes. De cette façon, vous pouvez surveiller le statut actuel de vos instances de base de données Aurora Serverless v2. Vous pouvez définir des alarmes qui vous avertissent si une instance de base de données Aurora Serverless v2 se rapproche d'un seuil pour les métriques liées à la capacité. Vous pouvez déterminer si les valeurs de capacité minimale et maximale sont appropriées ou si vous devez les ajuster. Vous pouvez déterminer où concentrer vos efforts pour optimiser l'efficacité de votre base de données.

- `ServerlessDatabaseCapacity`. En tant que métrique de niveau instance, elle indique le nombre d'ACU représentées par la capacité actuelle de l'instance de base de données. En tant que métrique de niveau cluster, elle représente la moyenne des valeurs `ServerlessDatabaseCapacity` de toutes les instances de base de données Aurora Serverless v2 du cluster. Cette métrique est uniquement une métrique de niveau cluster dans Aurora Serverless v1. Dans Aurora Serverless v2, elle est disponible au niveau de l'instance de base de données et au niveau du cluster.
- `ACUUtilization`. Il s'agit d'une nouvelle métrique dans Aurora Serverless v2. Cette valeur est représentée sous forme de pourcentage. Elle est calculée comme la valeur de la métrique `ServerlessDatabaseCapacity` divisée par le nombre maximal d'ACU du cluster de bases de données. Tenez compte des directives suivantes pour interpréter cette métrique et prendre les mesures nécessaires :
  - Si cette métrique se rapproche de la valeur `100.0`, l'instance de base de données a fait l'objet d'une augmentation d'échelle aussi élevée que possible. Envisagez d'augmenter le nombre maximal d'ACU pour le cluster. De cette façon, les instances de base de données de lecteur et d'enregistreur peuvent être mis à l'échelle jusqu'à une capacité supérieure.
  - Supposons qu'une charge de travail en lecture seule force une instance de base de données de lecteur à se rapprocher de la valeur `100.0` pour `ACUUtilization`, alors que l'instance de base de données d'enregistreur n'est pas proche de sa capacité maximale. Dans ce cas, envisagez d'ajouter des instances de base de données de lecteur supplémentaires au cluster. De cette

façon, vous pouvez répartir la partie en lecture seule de la charge de travail sur un plus grand nombre d'instances de base de données, réduisant ainsi la charge sur chaque instance de base de données de lecteur.

- Supposons que vous exécutiez une application de production, où les performances et la capacité de mise à l'échelle sont les principaux facteurs à prendre en compte. Dans ce cas, vous pouvez définir le nombre maximal d'ACU du cluster sur une valeur élevée. Votre objectif est que la métrique `ACUUtilization` soit toujours inférieure à `100.0`. Avec un nombre maximal d'ACU élevé, vous avez la garantie que l'espace est suffisant en cas de pics d'activité inattendus de la base de données. Seule la capacité de base de données réellement consommée vous est facturée.

- `CPUUtilization`. Cette métrique est interprétée différemment dans Aurora Serverless v2 par rapport aux instances de base de données approvisionnées. Pour Aurora Serverless v2, cette valeur est un pourcentage calculé comme la quantité de processeur actuellement utilisée, divisée par la capacité de processeur disponible sous le nombre maximal d'ACU du cluster de bases de données Aurora surveille automatiquement cette valeur et augmente l'échelle de votre instance de base de données Aurora Serverless v2 lorsque cette dernière utilise systématiquement une proportion élevée de sa capacité de processeur.

  Si cette métrique se rapproche de la valeur `100.0`, l'instance de base de données a atteint sa capacité de processeur maximale. Envisagez d'augmenter le nombre maximal d'ACU pour le cluster. Si cette métrique se rapproche de la valeur `100.0` sur une instance de base de données de lecteur, envisagez d'ajouter des instances de base de données de lecteur supplémentaires au cluster. De cette façon, vous pouvez répartir la partie en lecture seule de la charge de travail sur un plus grand nombre d'instances de base de données, réduisant ainsi la charge sur chaque instance de base de données de lecteur.

- `FreeableMemory`. Cette valeur représente la quantité de mémoire inutilisée disponible lorsque l'instance de base de données Aurora Serverless v2 est mise à l'échelle jusqu'à sa capacité maximale. Pour chaque ACU dont la capacité actuelle est inférieure à la capacité maximale, cette valeur augmente d'environ 2 Gio. Par conséquent, cette métrique ne se rapproche pas de zéro tant que l'instance de base de données ne fait pas l'objet d'une augmentation d'échelle aussi élevée que possible.

  Si cette métrique se rapproche de la valeur `0`, l'instance de base de données a fait l'objet d'une augmentation d'échelle aussi élevée que possible et se rapproche de la limite de sa mémoire disponible. Envisagez d'augmenter le nombre maximal d'ACU pour le cluster. Si cette métrique se rapproche de la valeur `0` sur une instance de base de données de lecteur, envisagez d'ajouter des

instances de base de données de lecteur supplémentaires au cluster. De cette façon, vous pouvez répartir la partie en lecture seule de la charge de travail sur un plus grand nombre d'instances de base de données, réduisant ainsi l'utilisation de la mémoire sur chaque instance de base de données de lecteur.

- `TempStorageIOPS`. Nombre d'E/S par seconde réalisées sur le stockage local attaché à l'instance de base de données. Il inclut les E/S par seconde pour les lectures et les écritures. Cette métrique représente un nombre et est mesurée une fois par seconde. Il s'agit d'une nouvelle mesure pour Aurora Serverless v2. Pour en savoir plus, consultez Métriques de niveau instance pour Amazon Aurora.
- `TempStorageThroughput`. Volume de données transférées depuis et vers le stockage local associé à l'instance de base de données. Cette métrique représente des octets et est mesurée une fois par seconde. Il s'agit d'une nouvelle mesure pour Aurora Serverless v2. Pour en savoir plus, consultez Métriques de niveau instance pour Amazon Aurora.

Généralement, la plupart des augmentations d'échelle pour Aurora Serverless v2 est engendrée par l'utilisation de la mémoire et l'activité du processeur. Les métriques `TempStorageIOPS` et `TempStorageThroughput` peuvent vous aider à diagnostiquer les rares cas où l'activité du réseau pour les transferts entre votre instance de base de données et vos périphériques de stockage locaux est responsable d'augmentations de capacité inattendues. Pour surveiller d'autres activités du réseau, vous pouvez utiliser ces métriques existantes :

- `NetworkReceiveThroughput`
- `NetworkThroughput`
- `NetworkTransmitThroughput`
- `StorageNetworkReceiveThroughput`
- `StorageNetworkThroughput`
- `StorageNetworkTransmitThroughput`

Vous pouvez faire en sorte qu'Aurora publie une partie ou la totalité des journaux de base de données sur Amazon CloudWatch Logs. Pour en savoir plus, consultez les sections suivantes en fonction de votre moteur de base de données :

- the section called "Publication de journaux Aurora PostgreSQL sur CloudWatch Logs"
- the section called "Publication de journaux Aurora MySQL dans CloudWatch Logs"

Application des métriques Aurora Serverless v2 à votre facture AWS

Les frais imputés à Aurora Serverless v2 sur votre facture AWS sont calculés sur la base de la même métrique `ServerlessDatabaseCapacity` que vous pouvez surveiller. Le mécanisme de facturation peut différer de la moyenne CloudWatch calculée pour cette métrique dans les cas où vous utilisez la capacité Aurora Serverless v2 pendant moins d'une heure. Il peut également différer si des problèmes système rendent la métrique CloudWatch indisponible pendant de courtes périodes. Par conséquent, la valeur correspondant aux heures ACU peut être légèrement différente sur votre facture que si vous calculez vous-même le nombre à partir de la valeur moyenne de `ServerlessDatabaseCapacity`.

Exemples de commandes CloudWatch pour les métriques Aurora Serverless v2

Les exemples AWS CLI suivants montrent comment surveiller les métriques CloudWatch les plus importantes relatives à Aurora Serverless v2. Dans chaque cas, remplacez la chaîne `Value=` du paramètre `--dimensions` par l'identifiant de votre propre instance de base de données Aurora Serverless v2.

L'exemple Linux suivant affiche les valeurs de capacité minimale, maximale et moyenne d'une instance de base de données, mesurées toutes les 10 minutes sur une heure. La commande Linux `date` indique les heures de début et de fin par rapport à la date et à l'heure actuelles. La fonction `sort_by` du paramètre `--query` trie les résultats par ordre chronologique en fonction du champ `Timestamp`.

```
aws cloudwatch get-metric-statistics --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Les exemples Linux suivants illustrent la surveillance de la capacité de chaque instance de base de données d'un cluster. Ils mesurent l'utilisation de capacité minimale, maximale et moyenne de chaque instance de base de données. Les mesures sont effectuées une fois par heure sur une période de trois heures. Ces exemples utilisent la métrique `ACUUtilization` qui représente un pourcentage de la limite supérieure sur les ACU, plutôt que la métrique `ServerlessDatabaseCapacity` qui représente un nombre fixe d'ACU. De cette façon, vous n'avez pas besoin de connaître les chiffres réels pour les nombres d'ACU minimal et maximal dans la plage de capacité. Les pourcentages sont compris entre 0 et 100.

```
aws cloudwatch get-metric-statistics --metric-name "ACUUtilization" \
  --start-time "$(date -d '3 hours ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_writer_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table

aws cloudwatch get-metric-statistics --metric-name "ACUUtilization" \
  --start-time "$(date -d '3 hours ago')" --end-time "$(date -d 'now')" --period 3600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_reader_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

L'exemple Linux suivant effectue des mesures similaires aux précédentes. Dans ce cas, les mesures concernent la métrique `CPUUtilization`. Les mesures sont effectuées toutes les 10 minutes sur une période d'une heure. Les chiffres représentent le pourcentage de processeur disponible utilisé, en fonction des ressources de processeur disponibles pour la valeur de capacité maximale de l'instance de base de données.

```
aws cloudwatch get-metric-statistics --metric-name "CPUUtilization" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

L'exemple Linux suivant effectue des mesures similaires aux précédentes. Dans ce cas, les mesures concernent la métrique `FreeableMemory`. Les mesures sont effectuées toutes les 10 minutes sur une période d'une heure.

```
aws cloudwatch get-metric-statistics --metric-name "FreeableMemory" \
  --start-time "$(date -d '1 hour ago')" --end-time "$(date -d 'now')" --period 600 \
  --namespace "AWS/RDS" --statistics Minimum Maximum Average \
  --dimensions Name=DBInstanceIdentifier,Value=my_instance \
  --query 'sort_by(Datapoints[*].
{min:Minimum,max:Maximum,avg:Average,ts:Timestamp},&ts)' --output table
```

Surveillance des performances d’Aurora Serverless v2 avec Performance Insights

Vous pouvez utiliser Performance Insights pour surveiller les performances des instances de base de données Aurora Serverless v2. Pour connaître les procédures relatives à Performance Insights, consultez Surveillance de la charge de la base de données avec Performance Insights sur .

Les nouveaux compteurs Performance Insights suivants s’appliquent aux instances de base de données Aurora Serverless v2.

- `os.general.serverlessDatabaseCapacity` : capacité actuelle de l’instance de base de données en ACU. La valeur correspond à la métrique CloudWatch `ServerlessDatabaseCapacity` pour l’instance de base de données.
- `os.general.acuUtilization` : pourcentage de la capacité actuelle par rapport à la capacité maximale configurée. La valeur correspond à la métrique CloudWatch `ACUUtilization` pour l’instance de base de données.
- `os.general.maxConfiguredAcu` : capacité maximale que vous avez configurée pour cette instance de base de données Aurora Serverless v2. Elle est mesurée en ACU.
- `os.general.minConfiguredAcu` – Capacité minimale que vous avez configurée pour cette instance de base de données Aurora Serverless v2. Elle est mesurée en ACU.

Pour obtenir la liste complète des compteurs Performance Insights, consultez Métrique de compteur de Performance Insights.

Lorsque les valeurs `vCPU` sont affichées pour une instance de base de données Aurora Serverless v2 dans Performance Insights, ces valeurs représentent des estimations basées sur le nombre d’ACU pour l’instance de base de données. À l’intervalle par défaut d’une minute, toutes les valeurs vCPU fractionnelles sont arrondies au nombre entier le plus proche. Pour les intervalles plus longs, la valeur vCPU affichée correspond à la moyenne des valeurs entières de vCPU pour chaque minute.

Résolution des problèmes de capacité d’Aurora Serverless v2

Dans certains cas, Aurora Serverless v2 ne se réduit pas à la capacité minimale, même si la base de données n’est pas chargée. Plusieurs raisons sont possibles :

- Certaines fonctionnalités peuvent augmenter l’utilisation des ressources et empêcher la réduction de la base de données à sa capacité minimale. Les principales fonctions sont notamment :

  - Bases de données globales Aurora
  - Exportation des journaux CloudWatch Logs
  - Activation de `pg_audit` sur les clusters de bases de données compatibles avec Aurora PostgreSQL
  - Surveillance améliorée
  - Performance Insights

  Pour plus d'informations, consultez Choix de la valeur minimale de capacité Aurora Serverless v2 pour un cluster.
- Si une instance de lecture n'est pas réduite au minimum et reste à une capacité identique ou supérieure à celle de l'instance d'enregistreur, vérifiez le niveau de priorité de l'instance de lecture. Les instances de base de données Aurora Serverless v2 de lecture de niveau 0 ou 1 sont maintenues à une capacité minimale au moins aussi élevée que l'instance de base de données d'écriture. Changez le niveau de priorité du processus de lecture à 2 ou plus pour qu'il augmente et diminue indépendamment du processus d'écriture. Pour plus d'informations, consultez Choix du niveau de promotion pour un lecteur Aurora Serverless v2.
- Définissez tous les paramètres de la base de données qui ont un impact sur la taille de la mémoire partagée à leurs valeurs par défaut. La définition d'une valeur supérieure à la valeur par défaut augmente les besoins en mémoire partagée et empêche la base de données de se réduire à la capacité minimale. Par exemple, `max_connections` et `max_locks_per_transaction`.

  > **Note**
  >
  > La mise à jour des paramètres de mémoire partagée nécessite un redémarrage de la base de données pour que les changements prennent effet.

- De lourdes charges de travail de base de données peuvent augmenter l'utilisation des ressources.
- D'importants volumes de base de données peuvent augmenter l'utilisation des ressources.

  Amazon Aurora utilise des ressources de mémoire et de processeur pour la gestion des clusters de bases de données. Aurora nécessite plus d'UC et de mémoire pour gérer des clusters de bases de données comportant d'importants volumes de base de données. Si la capacité minimale de votre cluster est inférieure à la capacité minimale requise pour la gestion du cluster, votre cluster ne sera pas réduit à la capacité minimale.
- Les processus d'arrière-plan, tels que la purge, peuvent également augmenter l'utilisation des ressources.

- Les limites de la version de plateforme peuvent affecter les fonctionnalités de mise à l’échelle. La plage de mise à l’échelle disponible pour un cluster donné dépend à la fois de la version du moteur et du matériel (version de plateforme). Il est possible qu’une version de moteur plus performante fonctionne sur une version de plateforme moins performante, et vice-versa.

Si la base de données n’est toujours pas réduite à la capacité minimale configurée, arrêtez et redémarrez la base de données pour récupérer les fragments de mémoire qui ont pu s’accumuler au fil du temps. L’arrêt et le démarrage d’une base de données entraînent une durée d’indisponibilité, il est donc recommandé de le faire avec parcimonie.

Réduction verticale à zéro ACU avec pause et reprise automatiques pour Aurora Serverless v2

Vous pouvez spécifier que les instances de base de données Aurora Serverless v2 sont réduites verticalement à 0 ACU et qu'elles se mettent automatiquement en pause si aucune connexion n'est initiée par l'activité des utilisateurs au cours d'une période spécifiée. Pour ce faire, spécifiez une valeur ACU minimale correspondant à zéro pour votre cluster de bases de données. La capacité d'une instance ne vous est pas facturée lorsque l'instance est à l'état de pause. L'activation de la fonctionnalité de pause et de reprise automatiques (pause automatique) pour les clusters Aurora peu utilisés ou présentant des périodes d'inactivité prolongées peut vous aider à gérer les coûts de votre flotte de bases de données.

> **Note**
>
> La fonctionnalité de pause automatique est disponible pour Aurora Serverless v2 avec Aurora PostgreSQL et Aurora MySQL. Vous devrez peut-être mettre à niveau la version de votre moteur de base de données Aurora pour tirer parti de cette fonctionnalité. Pour déterminer les versions de moteur où une capacité minimale de 0 ACU est disponible, consultez Capacité Aurora Serverless v2.

Rubriques

Présentation de la fonctionnalité de pause automatique Aurora Serverless v2

Les instances de base de données Aurora Serverless v2 peuvent être mises en pause automatiquement après une période sans connexion utilisateur et reprendre automatiquement lorsqu'une demande de connexion arrive. La fonctionnalité de pause et de reprise automatiques d'Aurora Serverless v2 permet de gérer les coûts pour les systèmes qui ne sont pas soumis à un objectif de niveau de service (SLO) strict. Par exemple, vous pouvez activer cette fonctionnalité pour les clusters utilisés pour le développement et les tests, ou pour les applications internes où une courte pause est acceptable pendant le redémarrage de la base de données. Si votre charge de travail implique des périodes d'inactivité et peut tolérer de légers retards de connexion lors de la reprise de l'instance, envisagez de mettre en pause automatiquement vos instances Aurora Serverless v2 afin de réduire les coûts.

Pour contrôler ce comportement, spécifiez si les instances de base de données Aurora Serverless v2 d'un cluster peuvent être mises en pause automatiquement ou non, et pendant combien de temps chaque instance doit rester inactive avant de se mettre en pause. Pour activer le comportement de pause automatique pour toutes les instances de base de données Aurora Serverless v2 d'un cluster Aurora, définissez la valeur correspondant à la capacité minimale du cluster sur 0 ACU.

Si vous avez déjà tiré parti de la fonctionnalité d'Aurora Serverless v1 qui appliquait 0 ACU après une période d'inactivité, vous pouvez passer à Aurora Serverless v2 et utiliser sa fonctionnalité de pause automatique correspondante.

Les avantages en termes d'économies de coûts de la fonctionnalité de pause automatique sont similaires à ceux de l'utilisation de la fonctionnalité d'arrêt et de démarrage de cluster. La mise en pause automatique d'Aurora Serverless v2 présente les avantages supplémentaires d'une reprise plus rapide par rapport au démarrage d'un cluster arrêté et à l'automatisation du processus de détermination du moment de la pause et de la reprise de chaque instance de base de données.

La fonctionnalité de pause automatique fournit également une granularité supplémentaire dans le contrôle des coûts des ressources de calcul au sein de votre cluster. Vous pouvez activer la pause de certaines instances de lecteur même si l'instance de l'enregistreur et les autres lecteurs du cluster restent actifs en permanence. Pour ce faire, attribuez aux instances de lecteur qui peuvent être mises en pause indépendamment des autres instances une priorité de basculement comprise entre 2 et 15.

Les instances de l'enregistreur et toutes les instances de lecteur dont les priorités de basculement sont de 0 et 1 se mettent en pause et reprennent toujours en même temps. Ainsi, les instances de

ce groupe se mettent en pause après qu'aucune connexion n'a eu lieu pendant l'intervalle de temps spécifié.

Les clusters de bases de données Aurora peuvent contenir une combinaison d'instances de base de données d'enregistreur et de lecteur, ainsi que d'instances de base de données et d'instances de base de données Aurora Serverless v2 provisionnées. Par conséquent, pour utiliser cette fonctionnalité de manière efficace, il est utile de comprendre les aspects suivants du mécanisme de pause automatique :

- Circonstances dans lesquelles une instance de base de données peut se mettre automatiquement en pause.
- Cas dans lesquels une instance de base de données ne peut pas se mettre en pause. Par exemple, l'activation de certaines fonctionnalités d'Aurora ou l'exécution de certains types d'opérations sur le cluster peuvent empêcher les instances de se mettre en pause, même en l'absence de connexion à ces instances.
- Conséquences en matière de surveillance et de facturation lorsqu'une instance est mise en pause.
- Actions pouvant entraîner la reprise du traitement par une instance de base de données.
- Modification de la capacité d'une instance de base de données au moment des événements de pause et de reprise.
- Comment contrôler l'intervalle d'inactivité avant la pause d'une instance de base de données.
- Comment coder la logique de l'application pour gérer la période pendant laquelle une instance de base de données reprend le traitement.

Conditions préalables et limitations de la fonctionnalité Aurora Serverless v2 de pause automatique

Avant d'utiliser la fonctionnalité de pause automatique, vérifiez les versions de moteur à utiliser. Vérifiez également si la pause automatique fonctionne en combinaison avec les autres fonctionnalités Aurora que vous avez l'intention d'utiliser. Vous ne pouvez pas activer la pause automatique si vous utilisez une version de moteur qui ne la prend pas en charge. Pour les fonctionnalités incompatibles, aucune erreur ne s'affichera si vous les utilisez en combinaison avec la pause automatique. Si le cluster utilise des fonctionnalités ou des paramètres incompatibles, les instances Aurora Serverless v2 ne seront pas automatiquement mises en pause.

- Si vous utilisez Aurora PostgreSQL, le moteur de base de données doit exécuter au moins la version 16.3, 15.7, 14.12 ou 13.15.

- Si vous utilisez Aurora MySQL, le moteur de base de données doit exécuter la version 3.08.0 ou une version supérieure.
- Pour obtenir la liste complète des versions de moteur et des régions AWS dans lesquelles cette fonctionnalité est disponible, consultez Régions et moteurs de base de données Aurora pris en charge pour Aurora Serverless v2.
- Lorsqu'une instance Aurora Serverless v2 reprend, sa capacité peut être inférieure à ce qu'elle était lorsque l'instance a été mise en pause. Pour en savoir plus, consultez Différences du comportement de pause automatique entre Aurora Serverless v2 et Aurora Serverless v1.

Certaines conditions ou certains paramètres empêchent les instances Aurora Serverless v2 de se mettre en pause automatiquement. Pour plus d'informations, consultez Situations dans lesquelles Aurora Serverless v2 n'applique pas de pause automatique.

Activation et désactivation de la fonctionnalité de pause automatique

Vous pouvez activer et désactiver la fonctionnalité de pause automatique au niveau du cluster. Pour cela, vous devez utiliser les mêmes procédures que lorsque vous ajustez la capacité minimale et maximale du cluster. La fonctionnalité de pause automatique est représentée par une capacité minimale de 0 ACU.

Rubriques

- Activation de la pause automatique pour les instances Aurora Serverless v2 d'un cluster
- Intervalle d'expiration configurable de la fonctionnalité Aurora Serverless v2 de pause automatique
- Reprise d'une instance Aurora Serverless v2 mise en pause automatique
- Désactivation de la pause automatique pour les instances Aurora Serverless v2 d'un cluster

Activation de la pause automatique pour les instances Aurora Serverless v2 d'un cluster

Suivez la procédure décrite dans Définition de la plage de capacité Aurora Serverless v2 d'un cluster. Pour la capacité minimale, choisissez 0 ACU. Lorsque vous choisissez une capacité minimale de 0 ACU, vous pouvez également spécifier la durée pendant laquelle l'instance doit rester inactive avant qu'elle ne soit automatiquement mise en pause.

L'exemple d'interface de ligne de commande suivant montre comment créer un cluster Aurora en activant la fonctionnalité de pause automatique et en définissant l'intervalle de pause automatique sur 10 minutes (600 secondes).

```
aws rds create-db-cluster \
    --db-cluster-identifier my-serverless-v2-cluster \
    --region eu-central-1 \
    --engine aurora-mysql \
    --engine-version 8.0 \
    --serverless-v2-scaling-configuration
 MinCapacity=0,MaxCapacity=4,SecondsUntilAutoPause=600 \
    --master-username myuser \
    --manage-master-user-password
```

L'exemple d'interface de ligne de commande suivant montre comment activer la fonctionnalité de pause automatique pour un cluster Aurora existant. Cet exemple définit l'intervalle de pause automatique sur une heure (3 600 secondes).

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration
 MinCapacity=0,MaxCapacity=80,SecondsUntilAutoPause=3600

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 0,
    "MaxCapacity": 80.0,
    "SecondsUntilAutoPause": 3600
}
```

Intervalle d'expiration configurable de la fonctionnalité Aurora Serverless v2 de pause automatique

L'intervalle d'expiration est représenté dans l'attribut `ServerlessV2ScalingConfiguration` de votre cluster Aurora. Vous pouvez spécifier cet intervalle dans la AWS Management Console lors de la création ou de la modification d'un cluster Aurora, si la capacité minimale est définie sur 0 ACU. Vous pouvez le spécifier dans l'AWS CLI en utilisant le paramètre `--serverless-v2-scaling-configuration` lors de la création ou de la modification d'un cluster Aurora. Vous pouvez le spécifier dans l'API RDS en utilisant le paramètre `ServerlessV2ScalingConfiguration` lors de la création ou de la modification d'un cluster Aurora.

L’intervalle minimum que vous pouvez définir est de 300 secondes (5 minutes). C’est la valeur par défaut si vous ne spécifiez aucun intervalle. L’intervalle maximum que vous pouvez définir est 86 400 secondes (1 journée).

Supposons que vous désactiviez la fonctionnalité de pause automatique pour un cluster en remplaçant la capacité minimale du cluster par une valeur différente de zéro. Dans ce cas, la propriété d’intervalle sera supprimée de l’attribut `ServerlessV2ScalingConfiguration`. L’absence de cette propriété fournit une confirmation supplémentaire que la fonctionnalité de pause automatique est désactivée pour ce cluster. Si vous réactivez ultérieurement la pause automatique, vous pourrez à nouveau spécifier un intervalle personnalisé à ce moment-là.

Reprise d’une instance Aurora Serverless v2 mise en pause automatique

- Lorsque vous vous connectez à une instance Aurora Serverless v2 mise en pause, celle-ci reprend automatiquement et accepte la connexion.
- Une tentative de connexion qui n’inclut pas d’informations d’identification valides entraîne tout de même la reprise de l’instance de base de données.
- Si vous vous connectez via le point de terminaison de l’enregistreur, Aurora entame la reprise de l’instance de base de données de l’enregistreur si elle a été automatiquement mise en pause. Dans le même temps, Aurora reprend toutes les instances de lecteur mises en pause automatique dont la priorité de basculement est de 0 ou 1, ce qui signifie que leur capacité est liée à la capacité de l’instance de l’enregistreur.
- Si vous vous connectez via le point de terminaison du lecteur, Aurora choisit une instance de lecteur de manière aléatoire. Si cette instance de lecteur est mise en pause, Aurora la reprend. Aurora reprend également l’instance de l’enregistreur en premier, car elle doit toujours être active si des instances de lecteur sont actives. Lorsqu’Aurora reprend cette instance d’enregistreur, cela entraîne également la reprise de toutes les instances de lecteur des niveaux 0 et 1 de la promotion de basculement.
- Si vous envoyez une demande à votre cluster via l’API de données RDS, Aurora reprend l’instance de l’enregistreur si elle est mise en pause. Aurora traite ensuite la demande de l’API de données.
- Lorsque vous modifiez la valeur d’un paramètre de configuration dans un groupe de paramètres de cluster de bases de données, Aurora reprend automatiquement toutes les instances Aurora Serverless v2 mises en pause dans tous les clusters qui utilisent ce groupe de paramètres de cluster. De même, lorsque vous modifiez la valeur d’un paramètre dans un groupe de paramètres de base de données, Aurora reprend automatiquement toutes les instances Aurora Serverless v2 mises en pause qui utilisent ce groupe de paramètres de base de données. Le même comportement de reprise automatique s’applique lorsque vous modifiez un cluster pour attribuer

un autre groupe de paramètres de cluster, ou lorsque vous modifiez une instance pour attribuer un autre groupe de paramètres de base de données.

- L'exécution d'une demande de retour en arrière permet de reprendre automatiquement l'instance d'enregistreur Aurora Serverless v2 si elle est mise en pause. Aurora traite la demande de retour en arrière une fois que l'instance d'enregistreur a repris ses activités. Vous pouvez revenir à une période pendant laquelle une instance Aurora Serverless v2 a été mise en pause.
- La création d'un instantané de cluster ou la suppression d'un instantané n'entraîne la reprise d'aucune instance Aurora Serverless v2.
- La création d'un clone Aurora oblige Aurora à reprendre l'instance d'enregistreur du cluster en cours de clonage.
- Si une instance mise en pause reçoit un grand nombre de demandes de connexion avant la fin de sa reprise, il est possible que certaines sessions ne puissent pas se connecter. Nous recommandons d'implémenter une logique de nouvelle tentative pour les connexions aux clusters Aurora ayant certaines instances Aurora Serverless v2 pour lesquelles la pause automatique est activée. Par exemple, vous pouvez réessayer trois fois une connexion ayant échoué.
- Aurora peut effectuer certains types de maintenance interne mineure sans mettre une instance hors veille. Cependant, certains types de maintenance qui ont lieu pendant la fenêtre de maintenance du cluster nécessitent qu'Aurora reprenne l'instance. Lorsque la maintenance est terminée, l'instance est automatiquement remise en pause s'il n'y a plus d'activité après l'intervalle spécifié.

  > **Note**
  >
  > Aurora ne reprend pas automatiquement une instance mise en pause pour les tâches planifiées spécifiques au moteur, telles que celles de l'extension PostgreSQL `pg_cron` ou du planificateur d'événements MySQL. Pour garantir l'exécution de ces tâches, établissez une connexion manuelle avec l'instance avant l'heure planifiée. Aurora ne met en file d'attente aucune tâche lorsque l'heure planifiée a lieu alors que l'instance de base de données est mise en pause. Ces tâches seront ignorées lorsque l'instance reprendra ultérieurement.

- Si le cluster Aurora subit un basculement alors qu'une instance Aurora Serverless v2 est mise en pause automatique, Aurora peut reprendre une instance, puis la promouvoir cette instance en tant qu'enregistreur. La même chose peut se produire si une ou plusieurs instances de base de données sont supprimées du cluster alors qu'une instance est mise en pause. Dans ce cas, l'instance deviendra l'enregistreur dès sa reprise.

- Les opérations qui modifient les propriétés du cluster entraînent également la reprise des instances Aurora Serverless v2 mises en pause automatique. Par exemple, une instance mise en pause automatique reprend son activité pour des opérations telles que les suivantes :
  - Modification de la plage de mise à l'échelle du cluster.
  - Mise à niveau de la version du moteur du cluster.
  - Description ou téléchargement des fichiers journaux à partir d'une instance mise en pause. Vous pouvez examiner les données de journal historiques des instances en pause en activant le chargement des journaux vers CloudWatch et en analysant ces journaux via CloudWatch.

Désactivation de la pause automatique pour les instances Aurora Serverless v2 d'un cluster

Suivez la procédure décrite dans Définition de la plage de capacité Aurora Serverless v2 d'un cluster. Pour la capacité minimale, choisissez une valeur égale ou supérieure à 0,5. Lorsque vous désactivez la fonctionnalité de pause automatique, l'intervalle d'inactivité de l'instance est réinitialisé. Si vous réactivez la pause automatique, vous devez spécifier un nouvel intervalle d'expiration.

L'exemple d'interface de ligne de commande suivant montre comment désactiver la fonctionnalité de pause automatique pour un cluster Aurora existant. La sortie `describe-db-clusters` indique que l'attribut `SecondsUntilAutoPause` est supprimé lorsque la capacité minimale est définie sur une valeur différente de zéro.

```
aws rds modify-db-cluster --db-cluster-identifier serverless-v2-cluster \
  --serverless-v2-scaling-configuration MinCapacity=2,MaxCapacity=80

aws rds describe-db-clusters --db-cluster-identifier serverless-v2-cluster \
  --query 'DBClusters[*].ServerlessV2ScalingConfiguration|[0]'
{
    "MinCapacity": 2,
    "MaxCapacity": 80.0
}
```

Fonctionnement de la pause automatique dans Aurora Serverless v2

Vous pouvez utiliser les informations suivantes pour planifier votre utilisation de la fonctionnalité de pause automatique. Comprendre les circonstances dans lesquelles les instances se mettent en pause, reprennent ou restent actives peut vous aider à trouver un juste milieu entre disponibilité, réactivité et économies de coûts.

Rubriques

Que se passe-t-il en cas de mise en pause d'instances Aurora Serverless v2

Lorsqu'une instance de base de données Aurora Serverless v2 se met en pause après une période sans connexion :

- Aurora commence à mettre en pause l'instance une fois que l'intervalle spécifié s'est écoulé sans aucune connexion à l'instance, quel que soit le nombre d'ACU dont elle dispose à ce moment-là.
- Le mécanisme de pause n'est pas instantané. Une instance Aurora Serverless v2 sur le point d'être mise en pause automatique peut attendre un court instant pour prendre en compte toutes les modifications apportées au stockage Aurora.
- Les frais pour cette instance sont suspendus. La métrique `ServerlessV2Usage` a la valeur 0 lorsque l'instance est en pause.
- La valeur d'état de l'instance ne change pas. L'état est toujours affiché comme « disponible ».
- L'instance arrête d'écrire dans les fichiers journaux de la base de données. Elle arrête d'envoyer des métriques à CloudWatch, à part 0 % pour `CPUUtilization` et `ACUUtilization`, et 0 pour `ServerlessDatabaseCapacity`.
- Aurora émet des événements lorsqu'une instance de base de données Aurora Serverless v2 commence à se mettre en pause ou met fin à sa pause et si le mécanisme de pause est interrompu ou échoue. Pour plus d'informations sur ces événements, consultez Événements d'instance de base de données.

Que se passe-t-il lorsque les instances Aurora Serverless v2 mises en pause automatiquement reprennent

Lorsqu'une instance de base de données Aurora Serverless v2 reprend après avoir été automatiquement mise en pause, les conditions suivantes s'appliquent :

- Toutes les modifications de paramètres incluses dans les modifications `pending-reboot` sont appliquées lorsque l'instance reprend.
- Aurora émet des événements au niveau de l'instance lorsque chaque instance de base de données Aurora Serverless v2 commence à reprendre son activité ou termine sa reprise et si l'instance ne peut pas reprendre pour une raison quelconque. Pour plus d'informations sur ces événements, consultez Événements d'instance de base de données.
- Toutes les connexions demandées sont établies une fois que l'instance de base de données a fini de reprendre son activité. Le délai de reprise étant généralement d'environ 15 secondes, nous vous recommandons de régler les paramètres de délai d'expiration des clients pour qu'ils soient supérieurs à 15 secondes. Par exemple, dans les paramètres de votre pilote JDBC, vous pouvez ajuster les valeurs des paramètres `connectTimeout` et `sslResponseTimeout` pour qu'elles soient supérieures à 15 secondes.

> **Note**
>
> Si une instance Aurora Serverless v2 reste en pause pendant plus de 24 heures, Aurora peut la mettre en veille prolongée, ce qui lui prendra plus de temps pour reprendre. Dans ce cas, le temps de reprise peut être de 30 secondes ou plus, ce qui équivaut à peu près à un redémarrage de l'instance. Si votre base de données présente des périodes d'inactivité qui durent plus d'une journée, nous vous recommandons de définir des délais de connexion de 30 secondes ou plus.

Fonctionnement de la facturation des instances pour les clusters Aurora Serverless v2 mis en pause automatique

Lorsqu'une instance Aurora Serverless v2 est mise en pause automatiquement, ses frais d'instance sont nuls. La métrique `ServerlessV2Usage` est nulle pendant cette période. AWS facture toujours les frais de stockage Aurora et d'autres aspects du cluster qui ne sont pas liés à cette instance de base de données spécifique.

Situations dans lesquelles Aurora Serverless v2 n’applique pas de pause automatique

- Si la valeur de capacité minimale de votre cluster de bases de données est supérieure à 0 ACU, les instances Aurora Serverless v2 du cluster ne sont pas automatiquement mises en pause. Si vous avez des clusters avec des instances Aurora Serverless v2 datant qui existaient avant que la fonctionnalité de pause automatique ne soit disponible, le paramètre de capacité minimale le plus bas était de 0,5 ACU. Pour utiliser la fonctionnalité de pause automatique avec ces clusters, remplacez le paramètre de capacité minimale par 0 ACU.
- Si des connexions initiées par l’utilisateur sont ouvertes sur une instance Aurora Serverless v2, celle-ci ne sera pas mise en pause. L’instance ne se met pas non plus en pause pendant des activités telles que l’application de correctifs et les mises à niveau. Les connexions administratives utilisées par Aurora pour les surveillances de l’état ne sont pas considérées comme des activités et n’empêchent pas l’instance de se mettre en pause.
- Dans un cluster Aurora PostgreSQL sur lequel la réplication logique est activée, ou dans un cluster Aurora MySQL sur lequel la réplication du journal binaire est activée, l’instance d’enregistreur et toutes les instances de lecteur dont le niveau de promotion de basculement correspond à 0 ou à 1 ne sont pas automatiquement mises en pause. Aurora effectue constamment un minimum d’activités pour vérifier l’état de la connexion de réplication.

  Pour les clusters sur lesquels la réplication est activée, vous pouvez toujours avoir des instances de lecteur Aurora dans le cluster source qui se mettent en pause automatiquement. Pour cela, définissez leur priorité de basculement sur une valeur autre que 0 ou 1. De cette façon, elles pourront être mises en pause indépendamment de l’instance d’enregistreur.
- Si votre cluster Aurora est associé à un proxy RDS, ce dernier maintient une connexion ouverte avec chaque instance de base de données du cluster. Dès lors, aucune instance Aurora Serverless v2 d’un tel cluster n’est automatiquement mise en pause.
- Si votre cluster est le cluster principal d’une base de données globale Aurora, Aurora ne met pas automatiquement en pause l’instance d’enregistreur Aurora Serverless v2. Cela est dû au fait qu’un niveau d’activité constant est nécessaire sur l’instance d’enregistreur pour pouvoir gérer les autres clusters de la base de données globale. Comme l’instance d’enregistreur reste active, toutes les instances de lecteur Aurora Serverless v2 dont la priorité de basculement est de 0 ou 1 ne sont pas mises en pause automatiquement.
- Les instances Aurora Serverless v2 des clusters secondaires d’une base de données Aurora Global Database ne se mettent pas en pause automatiquement. Si un cluster de bases de données est promu qu statut de cluster autonome, la fonctionnalité de pause automatique devient effective si toutes les autres conditions sont remplies.

- Dans un cluster associé à une intégration zéro ETL à Redshift, l'instance d'enregistreur et toutes les instances de lecteur des niveaux de promotion de basculement 0 et 1 en cas ne se mettent pas en pause automatiquement.
- Outre l'activité sur le port de base de données principal de l'instance, si la fonctionnalité Babelfish est activée sur une instance Aurora PostgreSQL, toutes les connexions et activités sur le port T-SQL empêchent la mise en pause automatique de l'instance.

Fonctionnement de la pause automatique avec la fonctionnalité d'arrêt et de démarrage du cluster

Vous pouvez arrêter et démarrer un cluster Aurora lorsque la fonctionnalité de pause automatique est activée. Le fait que certaines instances soient mises en pause n'a pas d'importance. Lorsque vous redémarrez le cluster, toutes les instances Aurora Serverless v2 mises en pause reprennent automatiquement.

Lorsqu'un cluster Aurora est arrêté, les instances Aurora Serverless v2 mises en pause ne reprennent pas automatiquement en fonction des tentatives de connexion. Une fois le cluster redémarré, les mécanismes habituels de pause et de reprise des instances Aurora Serverless v2 s'appliquent.

Fonctionnement de la maintenance et des mises à niveau pour les clusters Aurora Serverless v2 mis en pause automatiquement

- Lorsqu'une instance Aurora Serverless v2 est mise en pause automatiquement, si vous tentez de mettre à niveau le cluster Aurora, Aurora reprend l'instance et la met à niveau.
- Aurora reprend régulièrement toutes les instances Aurora Serverless v2 mises en pause automatiquement pour effectuer des opérations de maintenance, telles que des mises à niveau de versions mineures et des modifications de propriétés telles que les groupes de paramètres.
- Lorsqu'une instance Aurora Serverless v2 sort de son état de veille pour une opération administrative telle qu'une mise à niveau ou l'application d'une maintenance, Aurora attend au moins 20 minutes avant de mettre à nouveau cette instance en pause. L'objectif est de permettre à toutes les opérations en arrière-plan de se terminer. Cette période de 20 minutes permet également d'éviter de mettre en pause et de reprendre l'instance plusieurs fois si celle-ci subit plusieurs opérations administratives successives.

Différences du comportement de pause automatique entre Aurora Serverless v2 et Aurora Serverless v1

- Le temps de reprise est amélioré dans Aurora Serverless v2 par rapport à Aurora Serverless v1. Le délai de reprise est généralement d'environ 15 secondes si l'instance a été mise en pause pendant moins de 24 heures. Si l'instance a été mise en pause pendant plus de 24 heures, le délai de reprise peut être plus long.
- Avec la méthode appliquée par Aurora Serverless v2 aux clusters multi-AZ, certaines instances de base de données du cluster peuvent être mise en pause tandis que d'autres sont actives. L'instance de l'enregistreur reprend chaque fois qu'un lecteur est exécuté, car l'enregistreur est nécessaire pour coordonner certaines activités au sein du cluster. Comme Aurora Serverless v1 n'utilise pas d'instances de lecteur, l'ensemble du cluster serait toujours mis en pause ou actif.
- Lorsque le point de terminaison du lecteur choisit au hasard une instance de lecteur à laquelle se connecter, cette instance peut déjà être active ou peut avoir été mise en pause automatiquement. Ainsi, le temps d'accès à l'instance de lecteur peut varier et être plus difficile à prévoir. Les clusters multi-AZ qui utilisent Aurora Serverless v2 et la mise en pause automatique peuvent donc bénéficier de la configuration de points de terminaison personnalisés pour des cas d'utilisation spécifiques en lecture seule, au lieu de diriger toutes les sessions en lecture seule vers le point de terminaison du lecteur.
- Les instances Aurora Serverless v2 sont soumises à des opérations de maintenance à la même fréquence que les instances provisionnées. Étant donné qu'Aurora reprend automatiquement les instances lorsque ce type de maintenance est nécessaire, vous constaterez peut-être que les instances Aurora Serverless v2 reprennent plus fréquemment que les clusters Aurora Serverless v1.

Fonctionnement de la pause automatique Aurora Serverless v2 pour différents types de clusters Aurora

Les considérations relatives à la fonctionnalité de pause automatique dépendent du nombre d'instances présentes dans votre cluster Aurora, des niveaux de promotion du basculement des instances de lecteur et du fait que toutes les instances sont Aurora Serverless v2 ou une combinaison des Aurora Serverless v2 et d'instances provisionnées.

Rubriques

- Fonctionnement de la pause automatique Aurora Serverless v2 pour l'instance d'enregistreur d'un cluster de bases de données
- Fonctionnement de la pause automatique Aurora Serverless v2 pour les clusters multi-AZ
- Fonctionnement de la pause automatique Aurora Serverless v2 pour les clusters dotés d'instances provisionnées

Disposition des clusters Aurora recommandée lors de l'utilisation de la pause automatique

Lorsque la fonction de pause automatique est activée, vous pouvez organiser votre cluster Aurora de manière à trouver le juste équilibre entre haute disponibilité, réponse rapide et capacité de mise à l'échelle en fonction de votre cas d'utilisation. Pour ce faire, vous devez choisir la combinaison d'instances Aurora Serverless v2, d'instances provisionnées et de niveaux de promotion du basculement pour les instances de base de données de votre cluster.

Les types de configurations suivants présentent différents compromis entre haute disponibilité et optimisation des coûts pour votre cluster :

- Pour un système de développement et de test, vous pouvez configurer un cluster de bases de données dans une seule zone de disponibilité avec une instance de base de données Aurora Serverless v2. Cette instance unique traite toutes les demandes de lecture et d'écriture. Lorsque le cluster n'est pas utilisé pendant des intervalles de temps importants, l'instance de base de données se met en pause. À ce stade, les coûts de calcul de la base de données pour votre cluster sont également mis en pause.
- Pour un système exécutant une application où la haute disponibilité est une priorité, mais où le cluster reste totalement inactif pendant des périodes spécifiques, vous pouvez configurer un cluster multi-AZ dans lequel les instances de base de données d'enregistreur et de lecteur sont Aurora Serverless v2. Définissez l'instance de lecteur sur une priorité de basculement de 0 ou 1, de sorte que l'instance de l'enregistreur et celle du lecteur se mettent en pause et reprennent en même temps. Vous bénéficiez ainsi d'un basculement rapide lorsque le cluster est actif. Lorsque le cluster reste inactif pendant une durée supérieure au seuil de pause automatique, les frais d'instance de base de données pour les deux instances sont mis en pause. Lorsque le cluster reprend le traitement, la première session de base de données met peu de temps à se connecter.
- Supposons que votre cluster soit constamment actif avec un minimum d'activité et qu'il nécessite une réponse rapide quelle que soit la connexion. Dans ce cas, vous pouvez créer un cluster avec plusieurs instances de lecteur Aurora Serverless v2 et dissocier les fonctionnalités de certaines

instances de lecteur de celle de l’enregistreur. Spécifiez une priorité de basculement de 0 ou 1 pour l’instance d’enregistreur et pour une instance de lecteur. Spécifiez une priorité supérieure à 1 pour les autres instances de lecteur. Ainsi, les instances de lecteur situées dans les niveaux de priorité les plus élevés pourront être mises en pause automatiquement, même lorsque l’enregistreur et l’un des lecteurs restent actifs.

Dans ce cas, vous pouvez utiliser d’autres techniques pour garantir que le cluster reste disponible en permanence tout en réduisant verticalement sa capacité pendant les périodes d’inactivité :

- Vous pouvez utiliser des instances provisionnées pour l’enregistreur et le lecteur de priorité 0 ou 1. Ainsi, deux instances de base de données ne se mettent jamais en pause automatiquement et sont toujours disponibles pour traiter le trafic de base de données et effectuer des basculements.
- Vous pouvez configurer un point de terminaison personnalisé qui inclut les instances Aurora Serverless v2 des niveaux de priorité les plus élevés, mais pas l’enregistreur ni les lecteurs des niveaux de promotion 0 ou 1. Ainsi, vous pouvez diriger les sessions en lecture seule qui ne sont pas sensibles à la latence vers des lecteurs susceptibles d’être mis en pause automatiquement. Vous pouvez éviter d’utiliser le point de terminaison du lecteur pour de telles demandes, car Aurora peut diriger les connexions de ce point de terminaison vers l’instance de lecteur qui reste active en permanence ou vers l’une des instances mises en pause automatiquement. L’utilisation du point de terminaison personnalisé vous permet de diriger les connexions vers différents groupes d’instances en fonction de vos préférences en matière de réponse rapide ou de mise à l’échelle supplémentaire.

Fonctionnement de la pause automatique Aurora Serverless v2 pour l’instance d’enregistreur d’un cluster de bases de données

Lorsqu’un cluster de bases de données Aurora ne contient qu’une seule instance de base de données, le mécanisme de pause et de reprise automatiques de l’instance de base de données est simple. Cela dépend uniquement de l’activité de l’instance de l’enregistreur. Vous pouvez utiliser cette configuration pour les clusters destinés au développement et aux tests, ou pour exécuter des applications où la haute disponibilité n’est pas cruciale. Notez que dans un cluster à instance unique, Aurora dirige les connexions via le point de terminaison du lecteur vers l’instance de base de données de l’enregistreur. Ainsi, pour un cluster de bases de données à instance unique, la tentative de connexion au point de terminaison du lecteur entraîne la reprise de l’instance de base de données d’enregistreur mise en pause automatiquement.

Les facteurs supplémentaires suivants s’appliquent aux clusters Aurora dotés de plusieurs instances de base de données :

- Dans un cluster de bases de données Aurora, l'accès à l'instance de base de données de l'enregistreur est généralement fréquent. Par conséquent, il se peut que l'instance de base de données de l'enregistreur reste active même lorsqu'une ou plusieurs instances de base de données du lecteur se mettent en pause automatiquement.
- Certaines activités sur les instances de base de données du lecteur nécessitent que cette instance de base de données de l'enregistreur soit disponible. Par conséquent, les instances de base de données de l'enregistreur ne peuvent pas être mises en pause tant que toutes les instances du lecteur ne le sont pas également. La reprise d'une instance de lecteur entraîne automatiquement la reprise de l'instance d'enregistreur, même si votre application n'y accède pas directement.
- Les instances de lecteur Aurora Serverless v2 dont les niveaux de promotion du basculement correspondent à 0 et 1 se mettent à l'échelle pour maintenir leur capacité synchronisée avec l'instance d'enregistreur. Ainsi, lorsqu'une instance d'enregistreur Aurora Serverless v2 reprend, il en va de même pour toutes les instances de lecteur Aurora Serverless v2 dont les niveaux de promotion correspondent à 0 ou 1.

Fonctionnement de la pause automatique Aurora Serverless v2 pour les clusters multi-AZ

Dans un cluster de bases de données Aurora contenant à la fois une instance de base de données d'enregistreur et une ou plusieurs instances de base de données de lecteur, certaines instances de base de données Aurora Serverless v2 peuvent être suspendues tandis que d'autres sont actives. L'instance d'enregistreur et toutes les instances de lecteur dont les priorités de basculement sont de 0 et 1 se mettent en pause et reprennent toujours en même temps. Les instances de lecteur dont la priorité est différente de 0 ou 1 peuvent se mettre en pause et reprendre indépendamment des autres instances.

Lorsque vous utilisez cette fonctionnalité pour des clusters comportant plusieurs instances de lecteur, vous pouvez gérer les coûts sans sacrifier la haute disponibilité. L'instance d'enregistreur et une ou deux autres instances de lecteur peuvent rester actives en permanence, tandis que les autres instances de lecteur peuvent se mettre en pause lorsqu'elles ne sont pas nécessaires pour gérer un trafic de lecture important.

Le fait qu'une instance de lecteur puisse être automatiquement mise en pause dépend de sa capacité à se mettre à l'échelle indépendamment ou du fait que la capacité est liée à celle de l'instance de base de données de l'enregistreur. Cette propriété de mise à l'échelle dépend de la priorité de basculement de l'instance de base de données du lecteur. Lorsque la priorité du lecteur est de 0 ou 1, la capacité du lecteur suit la capacité de l'instance de base de données de l'enregistreur. Par

conséquent, pour permettre aux instances de base de données du lecteur de se mettre en pause automatiquement dans le plus large éventail de situations, définissez leur priorité sur une valeur supérieure à 1.

Le délai de reprise d'une instance de lecteur peut être légèrement plus long que celui d'une instance d'enregistreur. Pour obtenir une réponse rapide en cas de mise en pause d'instances, connectez-vous au point de terminaison du cluster.

Fonctionnement de la pause automatique Aurora Serverless v2 pour les clusters dotés d'instances provisionnées

Aucune instance de base de données provisionnée dans votre cluster de bases de données Aurora ne se met en pause automatiquement. Seules les instances de base de données Aurora Serverless v2, avec la classe d'instance `db.serverless`, peuvent utiliser la fonctionnalité de pause automatique.

Lorsque votre cluster Aurora contient des instances de base de données provisionnées, aucune instance d'enregistreur Aurora Serverless v2 ne se met en pause automatiquement. Cela est dû au fait que l'instance d'enregistreur doit rester disponible tant que toutes les instances de lecteur sont actives. Le fait que l'enregistreur Aurora Serverless v2 reste actif signifie également que les instances de lecteur Aurora Serverless v2 dont les priorités de basculement sont de 0 ou 1 ne se mettent pas en pause automatiquement dans un cluster hybride contenant des instances provisionnées.

Surveillance des clusters Aurora utilisant la pause automatique

Pour surveiller Aurora, vous devez déjà connaître les procédures de surveillance indiquées dans Surveillance des métriques Amazon Aurora avec Amazon CloudWatch et les métriques CloudWatch répertoriées dans Référence des métriques pour Amazon Aurora. Sachez que certaines considérations particulières doivent être prises en compte lorsque vous surveillez des clusters Aurora qui utilisent la fonctionnalité de pause automatique :

- Il peut y avoir des périodes pendant lesquelles les instances Aurora Serverless v2 n'enregistrent pas les données de journal ni la plupart des métriques, car elles sont en pause. Les seules métriques envoyées à CloudWatch lorsqu'une instance est en pause sont 0 % pour `CPUUtilization` et `ACUUtilization`, et 0 pour `ServerlessDatabaseCapacity`.
- Vous pouvez vérifier si les instances Aurora Serverless v2 se mettent en pause plus ou moins fréquemment que prévu. Pour ce faire, vérifiez à quelle fréquence la métrique `ServerlessDatabaseCapacity` passe d'une valeur différente de zéro à zéro, et combien de temps elle reste nulle. Si les instances ne restent pas en pause aussi longtemps que prévu, vous

n’économiserez pas autant que vous le pourriez en termes de coûts. Si les instances se mettent en pause et reprennent plus fréquemment que prévu, il se peut que votre cluster subisse une latence inutile lorsqu’il répond aux demandes de connexion. Pour plus d’informations sur les facteurs qui déterminent si les instances Aurora Serverless v2peuvent se mettre en pause et à quelle fréquence, consultez Conditions préalables et limitations de la fonctionnalité Aurora Serverless v2 de pause automatique, Situations dans lesquelles Aurora Serverless v2 n’applique pas de pause automatique et Résolution des problèmes liés à la pause automatique Aurora Serverless v2.

- Vous pouvez également examiner un fichier journal qui enregistre les opérations de pause et de reprise automatiques pour une instance Aurora Serverless v2. Si une instance n’a pas été mise en pause après le délai d’expiration, ce fichier journal indique également la raison pour laquelle la pause automatique n’a pas eu lieu. Pour plus d’informations, consultez Surveillance de l’activité de mise en pause et de reprise d’Aurora Serverless v2.

Rubriques

Vérifier si une instance Aurora Serverless v2 est mise en pause

Pour déterminer si une instance Aurora Serverless v2 est en état de pause, vous pouvez observer sa métrique `ACUUtilization`. Cette métrique a la valeur 0 lorsque l’instance est en pause.

Lorsqu’une instance Aurora Serverless v2 est mise en pause, la valeur de son état apparaît toujours comme étant Disponible. Il en va de même lorsqu’une instance Aurora Serverless v2 mise en pause est en cours de reprise. Cela est dû au fait que vous pouvez vous connecter avec succès à ce type d’instance, même si la connexion est légèrement retardée.

Toutes les métriques liées à la disponibilité des instances Aurora prennent en compte la période pendant laquelle l’instance est en pause comme une période de disponibilité.

Événements pour les opérations de pause et de reprise automatiques

Aurora émet des événements pour les instances Aurora Serverless v2 lorsque les opérations de pause et de reprise automatiques démarrent, se terminent ou sont annulées. Les événements liés à

la fonction de pause automatique sont compris entre `RDS-EVENT-0370` et `RDS-EVENT-0374`. Pour plus d'informations sur ces événements, consultez Événements d'instance de base de données.

Fonctionnement de la pause automatique avec Performance Insights et la surveillance améliorée

Lorsqu'une instance Aurora Serverless v2 est mise en pause, Aurora ne collecte pas d'informations de surveillance pour cette instance via Performance Insights ou la surveillance améliorée. Lorsque l'instance reprend, il se peut qu'il y ait un bref délai avant qu'Aurora ne recommence à collecter ces informations de surveillance.

Interactions de la fonctionnalité Aurora Serverless v2 de pause automatique avec les métriques Aurora

Lorsqu'une instance Aurora Serverless v2 est mise en pause, elle n'émet pas la plupart des métriques CloudWatch et n'écrit aucune information dans les journaux de sa base de données. Les seules métriques envoyées à CloudWatch lorsqu'une instance est en pause sont 0 % pour `CPUUtilization` et `ACUUtilization`, et 0 pour `ServerlessDatabaseCapacity`.

Lorsque CloudWatch calcule des statistiques relatives à la disponibilité et à la durée de fonctionnement des instances ou des clusters, les instances Aurora Serverless v2 sont considérées comme disponibles pendant la période de mise en pause.

Lorsque vous lancez une action de l'AWS CLI ou de l'API RDS pour décrire ou télécharger les journaux d'une instance Aurora Serverless v2 mise en pause, cette instance reprend automatiquement pour rendre les informations du journal disponibles.

Exemple de métriques CloudWatch

Les exemples AWS CLI suivants montrent comment observer l'évolution de la capacité d'une instance au fil du temps. Au cours de cette période, l'instance se met automatiquement en pause puis reprend. Lorsqu'elle est en pause, la métrique `ServerlessDatabaseCapacity` indique la valeur 0. Pour déterminer si l'instance a été mise en pause à un moment quelconque au cours de la période, nous vérifions si la capacité minimale pendant cette période a été nulle.

L'exemple Linux suivant représente une instance Aurora Serverless v2 qui a été automatiquement mise en pause pendant un certain temps. Nous échantillonnons `ServerlessDatabaseCapacity` chaque minute, sur une période de trois minutes. Une valeur d'ACU minimale de 0,0 confirme que l'instance a été mise en pause à un moment donné au cours de chaque minute.

```
$ aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '3 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --statistics Minimum \
  --namespace "AWS/RDS" --dimensions Name=DBInstanceIdentifier,Value=my-autopause-
instance \
  --output text | sort -k 3

ServerlessDatabaseCapacity
DATAPOINTS 0.0 2024-08-02T22:11:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:12:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:13:00+00:00 None
```

Ensuite, nous essayons d'établir une connexion avec l'instance Aurora Serverless v2 mise en pause. Dans cet exemple, nous utilisons intentionnellement un mot de passe incorrect afin que la tentative de connexion échoue. Malgré cet échec, la tentative de connexion oblige Aurora à reprendre l'instance mise en pause.

```
$ mysql -h my_cluster_endpoint.rds.amazonaws.com -u admin -pwrong-password

ERROR 1045 (28000): Access denied for user 'admin'@'ip_address' (using password: YES)
```

L'exemple Linux suivant montre que l'instance mise en pause a repris, est restée inactive pendant environ cinq minutes, puis est revenue à son état de pause. L'instance a repris à une capacité de 2,0 ACU. Ensuite, elle a légèrement augmenté verticalement, par exemple pour effectuer un nettoyage interne. Comme elle n'a reçu aucune tentative de connexion utilisateur dans le délai de cinq minutes, elle est directement passée de 4 ACU à l'état de pause.

```
$ aws cloudwatch get-metric-statistics \
  --metric-name "ServerlessDatabaseCapacity" \
  --start-time "$(date -d '8 minutes ago')" --end-time "$(date -d 'now')" --period 60 \
  --statistics Minimum \
  --namespace "AWS/RDS" --dimensions Name=DBInstanceIdentifier,Value=my-autopause-
instance \
  --output text | sort -k 3

ServerlessDatabaseCapacity
DATAPOINTS 0.0 2024-08-02T22:13:00+00:00 None
```

```
DATAPOINTS 2.0 2024-08-02T22:14:00+00:00 None
DATAPOINTS 3.0 2024-08-02T22:15:00+00:00 None
DATAPOINTS 3.0 2024-08-02T22:16:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:17:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:18:00+00:00 None
DATAPOINTS 4.0 2024-08-02T22:19:00+00:00 None
DATAPOINTS 0.0 2024-08-02T22:20:00+00:00 None
```

Si le rapport était destiné à montrer à quelle vitesse l'instance a augmenté verticalement pour gérer la charge de travail, nous pouvons spécifier `Maximum` pour la statistique au lieu de `Minimum`. Pour la planification de la capacité et l'estimation des coûts, nous pouvons spécifier une période plus longue et utiliser la statistique `Average`. De cette façon, nous avons pu déterminer les valeurs de capacité typiques pendant les périodes de forte activité, de faible activité et d'état de pause. Pour examiner le comportement aux moments précis où la pause et la reprise ont lieu, nous pouvons spécifier une période d'une seconde et examiner un intervalle de temps plus court. Les valeurs d'horodatage figurant dans la sortie, par exemple `2024-08-02T22:13:00+00:00`, indiquent le format permettant de spécifier des paramètres précis pour les options `--start-time` et `--end-time`.

Résolution des problèmes liés à la pause automatique Aurora Serverless v2

Si vous constatez que les instances Aurora Serverless v2 ne se mettent pas en pause aussi souvent que prévu, recherchez les causes possibles suivantes :

- Vérifiez que la version d'Aurora que vous utilisez prend en charge une capacité minimale de 0 ACU. Pour en savoir plus sur les plages de capacité des différentes versions d'Aurora, consultez Capacité Aurora Serverless v2.
- Vérifiez que la valeur de capacité minimale du cluster est définie sur 0 ACU.
- Vérifiez que l'instance en question utilise réellement la classe d'instance Aurora Serverless v2 `db.serverless`, et non l'une des classes d'instance provisionnées.
- Vérifiez que le cluster n'utilise aucun des paramètres ou fonctionnalités incompatibles à partir de Conditions préalables et limitations de la fonctionnalité Aurora Serverless v2 de pause automatique.
- Examinez le fichier journal indiquant à quel moment les instances Aurora Serverless v2 ont été mises en pause ou reprises, ou à quel moment Aurora n'a pas pu mettre en pause ou reprendre une instance pour une raison ou une autre. Pour plus d'informations, consultez Surveillance de l'activité de mise en pause et de reprise d'Aurora Serverless v2.
- Vérifiez si des clients ou des applications maintiennent les connexions ouvertes pendant de longues périodes. À l'inverse, vérifiez si des applications utilisant l'API de données RDS ou les

fonctions Lambda envoient des demandes fréquentes afin que l'instance ne soit jamais inactive assez longtemps pour être mise en pause. Vous pouvez examiner les métriques CloudWatch telles que `ConnectionAttempts` et `DatabaseConnections`. Pour plus d'informations, consultez Métriques de niveau instance pour Amazon Aurora.

- Si une instance de lecteur se met rarement en pause, voire jamais, vérifiez sa priorité de basculement. Si le lecteur est utilisé pour la mise à l'échelle de la lecture et non comme lecteur de secours en cas de basculement, définissez sa priorité sur une valeur comprise entre 2 et 15.
- Si l'instance d'enregistreur se met rarement en pause, voire jamais, vérifiez l'utilisation des instances de lecteur. L'enregistreur ne peut se mettre en pause que si l'ensemble du cluster est inactif. Cela est dû au fait qu'une connexion à une instance de lecteur entraîne la reprise de l'enregistreur.
- Si votre application reçoit des délais d'expiration lors des demandes de connexion alors que les instances Aurora Serverless v2 reprennent leur activité, envisagez de prolonger l'intervalle d'expiration utilisé par le code de votre application ou le cadre de base de données sous-jacent. Des délais de connexion plus longs réduisent le risque d'échec des connexions lors de la reprise des instances Aurora Serverless v2. Cependant, les délais d'attente plus longs peuvent également ralentir la détection des problèmes de disponibilité dans votre cluster par votre application.

  À l'inverse, envisagez d'allonger l'intervalle de pause automatique afin que les applications ne trouvent pas aussi souvent d'instances en pause.

  S'il n'existe pas d'équilibre logique entre le comportement de pause automatique et la réactivité du cluster pour votre application, ce cluster n'est peut-être pas adapté à la fonctionnalité de pause automatique.
- Si vous estimez la durée pendant laquelle vos instances Aurora Serverless v2 sont mises en pause, sachez que certains facteurs empêchent de faire des prévisions précises.
  - Les instances peuvent reprendre périodiquement pour effectuer des opérations de maintenance, pour appliquer des mises à niveau de versions mineures ou pour apporter des modifications à des groupes de paramètres.
  - Pour les clusters multi-AZ, il existe des situations dans lesquelles la reprise d'une instance entraîne également la reprise d'autres instances. La reprise d'un lecteur entraîne toujours la reprise de l'enregistreur. La reprise de l'enregistreur entraîne toujours la reprise des lecteurs dont les priorités de basculement sont de 0 ou 1.

Nous recommandons de mesurer le temps de pause réel sur une période de plusieurs jours, avec une charge de travail réaliste. Utilisez ensuite ces mesures pour définir une base de référence pour la durée prévue de pause d'une instance.

- Vous constaterez peut-être que des opérations internes telles que la purge MySQL, le planificateur d'événements MySQL, l'autovacuum de PostgreSQL ou les tâches PostgreSQL planifiées via l'extension `pg_cron` ne s'exécutent pas ou ne se terminent pas. L'instance ne reprend pas automatiquement pour effectuer ce type d'opérations qui n'impliquent pas de connexion utilisateur à la base de données. Si ce type d'opérations internes est en cours à l'expiration de la pause automatique, les tâches de maintenance telles que la purge MySQL et l'autovacuum de PostgreSQL sont annulées. Les tâches planifiées à partir du planificateur d'événements MySQL ou de l'extension PostgreSQL `pg_cron` sont également annulées si elles sont en cours au moment où Aurora lance l'opération de pause.

  Si vous devez vous assurer que l'instance est régulièrement active afin de pouvoir effectuer les opérations planifiées, vous pouvez lancer une connexion pour que l'instance reprenne avant l'heure de début de la tâche. Vous pouvez également augmenter l'intervalle de pause automatique afin que les opérations telles que l'autovacuum puissent s'exécuter plus longtemps une fois l'activité de l'utilisateur terminée. Vous pouvez également utiliser des mécanismes tels que les fonctions Lambda pour effectuer des opérations de base de données selon un calendrier, de manière à ce que l'instance reprenne automatiquement si nécessaire.

Considérations relatives à la conception de la fonctionnalité Aurora Serverless v2 de pause automatique

Lorsqu'une instance de base de données Aurora Serverless v2 reprend après avoir été automatiquement mise en pause, elle commence avec une capacité relativement faible et augmente verticalement à partir de là. Cette capacité de départ s'applique même si l'instance de base de données avait une capacité supérieure juste avant sa mise en pause automatique.

Utilisez cette fonctionnalité avec les applications qui peuvent tolérer un intervalle d'environ 15 secondes lors de l'établissement d'une connexion. Cela tient compte du cas typique où une instance Aurora Serverless v2 reprend en raison d'une seule connexion entrante ou de quelques-unes. Si l'instance a été mise en pause pendant plus de 24 heures, le délai de reprise peut être plus long.

Si votre application utilisait déjà Aurora Serverless v1 et sa fonctionnalité de pause automatique, vous avez peut-être déjà mis en place de tels intervalles d'expiration pour les tentatives de connexion. Si vous utilisez déjà Aurora Serverless v2 en combinaison avec la fonctionnalité Aurora d'arrêt et de démarrage de cluster, le temps de reprise des instances Aurora Serverless v2 mises en pause automatiquement est généralement beaucoup plus court que le temps de démarrage d'un cluster arrêté.

Lorsque vous codez la logique de connexion de votre application, réessayez la connexion si la première tentative renvoie une erreur dont la cause est passagère. (Si l'erreur est due à un échec d'authentification, corrigez les informations d'identification avant de réessayer.) Une erreur qui se produit juste après la reprise peut être due à l'expiration d'un délai ou aux limites de la base de données. Une nouvelle tentative peut résoudre les problèmes dans les cas plus rares où une instance Aurora Serverless v2 reprend en raison d'un nombre élevé de demandes de connexion simultanées. Dans ce cas, le traitement de certaines connexions peut prendre plus de temps que d'habitude ou dépasser la limite de connexions simultanées lors de la première tentative.

Pendant le développement et le débogage de l'application, ne laissez pas les sessions client ni les outils de programmation avec des connexions ouvertes à la base de données. Aurora ne met pas en pause une instance si des connexions initiées par l'utilisateur sont ouvertes, même si ces connexions n'exécutent aucune instruction ou transaction SQL. Lorsqu'une seule instance Aurora Serverless v2 d'un cluster Aurora ne peut pas être mise en pause, la mise en pause d'autres instances de ce cluster peut également échouer. Pour plus d'informations, consultez Situations dans lesquelles Aurora Serverless v2 n'applique pas de pause automatique.

Aurora émet des événements lorsqu'une instance de base de données Aurora Serverless v2 commence à reprendre son activité ou termine sa reprise et si l'instance ne peut pas reprendre pour une raison quelconque. Pour plus d'informations sur ces événements, consultez Événements d'instance de base de données.

Migration vers Aurora Serverless v2

Pour convertir un cluster de bases de données existant afin qu'il utilise Aurora Serverless v2, vous pouvez effectuer les actions suivantes :

- Effectuer une mise à niveau à partir d'un cluster de bases de données Aurora provisionné.
- Effectuer une mise à niveau à partir d'un cluster Aurora Serverless v1.
- Migration d'une base de données sur site vers un cluster Aurora Serverless v2.

Lorsque votre cluster mis à niveau exécute la version appropriée du moteur, comme indiqué à la rubrique Exigences et limites relatives à Aurora Serverless v2, vous pouvez commencer à lui ajouter des instances de base de données Aurora Serverless v2. La première instance de base de données que vous ajoutez au cluster mis à niveau doit être une instance de base de données approvisionnée. Vous pouvez ensuite basculer le traitement de la charge de travail d'écriture et/ou de la charge de travail de lecture vers les instances de base de données Aurora Serverless v2.

Table des matières

> **Note**
>
> Ces rubriques expliquent comment convertir un cluster de bases de données existant. Pour obtenir des informations sur la création d'un nouveau cluster de bases de données Aurora Serverless v2, consultez Création d'un cluster de bases de données qui utilise Aurora Serverless v2.

Mise à niveau ou basculement de clusters existants pour utiliser Aurora Serverless v2

Si votre cluster approvisionné dispose d'une version de moteur qui prend en charge Aurora Serverless v2, le basculement vers Aurora Serverless v2 n'exige pas de mise à niveau. Dans ce cas, vous pouvez ajouter des instances de base de données Aurora Serverless v2 à votre cluster d'origine. Vous pouvez basculer le cluster de sorte qu'il utilise toutes les instances de base de données Aurora Serverless v2. Vous pouvez également utiliser une combinaison d'instances de base de données Aurora Serverless v2 et approvisionnées dans le même cluster de bases de données. Pour obtenir les versions du moteur Aurora prenant en charge Aurora Serverless v2, consultez Régions et moteurs de base de données Aurora pris en charge pour Aurora Serverless v2.

Si vous exécutez une version antérieure du moteur qui ne prend pas en charge Aurora Serverless v2, suivez ces étapes générales :

1. Mettez à niveau le cluster.
2. Créez une instance de base de données d'enregistreur approvisionnée pour le cluster mis à niveau.
3. Modifiez le cluster de sorte qu'il utilise des instances de base de données Aurora Serverless v2.

> **Important**
>
> Lorsque vous effectuez une mise à niveau de version majeure vers une version compatible avec Aurora Serverless v2 en utilisant la restauration ou le clonage d'instantané, la première instance de base de données que vous ajoutez au nouveau cluster doit être une instance de base de données approvisionnée. Cet ajout amorce la dernière étape du processus de mise à niveau.
>
> Tant que cette dernière étape n'intervient pas, le cluster ne dispose pas de l'infrastructure requise pour prendre en charge Aurora Serverless v2. Par conséquent, ces clusters mis à niveau commencent toujours avec une instance de base de données d'enregistreur approvisionnée. Vous pouvez ensuite convertir ou basculer l'instance de base de données approvisionnée vers une instance Aurora Serverless v2.

La mise à niveau d'Aurora Serverless v1 vers Aurora Serverless v2 implique la création d'un cluster approvisionné en tant qu'étape intermédiaire. Vous effectuez ensuite les mêmes étapes de mise à niveau que lorsque vous commencez avec un cluster approvisionné.

Chemins de mise à niveau pour les clusters compatibles avec MySQL pour utiliser Aurora Serverless v2

Si votre cluster d'origine exécute Aurora MySQL, choisissez la procédure appropriée en fonction de la version et du mode du moteur de votre cluster.

Si votre cluster Aurora MySQL d'origine est le suivant	Procédez comme suit pour basculer vers Aurora Serverless v2
Cluster approvisionné exécutant Aurora MySQL version 3, compatible avec MySQL 8.0	Il s'agit de la dernière étape pour toutes les conversions à partir de clusters Aurora MySQL existants. Si nécessaire, effectuez une mise à niveau d'une version mineure vers la version 3.02.0 ou ultérieure. Utilisez une instance de base de données approvisionnée pour l'instance de base de données d'enregistreur. Ajoutez une instance de base de données de lecteur Aurora Serverless v2. Procédez à un basculement pour en faire l'instance de base de données d'enregistreur. (Facultatif) Convertissez les autres instances de base de données provisionnées dans le cluster en Aurora Serverless v2. Ou ajoutez de nouvelles instances de base de données Aurora Serverless v2 et supprimez les instances de base de données provisionnées. Pour obtenir la procédure complète et des exemples, consultez Basculement d'un cluster approvisionné vers Aurora Serverless v2.

Si votre cluster Aurora MySQL d'origine est le suivant	Procédez comme suit pour basculer vers Aurora Serverless v2
Cluster approvisionné exécutant Aurora MySQL version 2, compatible avec MySQL 5.7	Effectuez une mise à niveau d'une version majeure vers Aurora MySQL version 3.02.0 ou ultérieure. Suivez ensuite la procédure propre à Aurora MySQL version 3 pour basculer le cluster de sorte à utiliser les instances de base de données Aurora Serverless v2.
Cluster Aurora Serverless v1 exécutant Aurora MySQL version 2, compatible avec MySQL 5.7	Pour vous aider à planifier votre conversion depuis Aurora Serverless v1, commencez par consulter Comparaison d'Aurora Serverless v2 avec Aurora Serverless v1. Suivez ensuite la procédure décrite dans Mise à niveau d'un cluster Aurora Serverless v1 vers Aurora Serverless v2.

Chemins de mise à niveau pour les clusters compatibles avec PostgreSQL pour utiliser Aurora Serverless v2

Si votre cluster d'origine exécute Aurora PostgreSQL, choisissez la procédure appropriée en fonction de la version et du mode du moteur de votre cluster.

Si votre cluster Aurora PostgreSQL d'origine est le suivant	Procédez comme suit pour basculer vers Aurora Serverless v2
Cluster approvisionné exécutant Aurora PostgreSQL version 13	Il s'agit de la dernière étape pour toutes les conversions à partir de clusters Aurora PostgreSQL existants. Si nécessaire, effectuez une mise à niveau d'une version mineure vers la version 13.6 ou ultérieure. Ajoutez une instance de base de données approvisionnée pour l'instance de base de données d'enregistreur. Ajoutez une

Si votre cluster Aurora PostgreSQL d'origine est le suivant	Procédez comme suit pour basculer vers Aurora Serverless v2
	instance de base de données de lecteur Aurora Serverless v2. Procédez à un basculement pour faire de cette instance Aurora Serverless v2 l'instance de base de données d'enregistreur. (Facultatif) Convertissez les autres instances de base de données provisionnées dans le cluster en Aurora Serverless v2. Ou ajoutez de nouvelles instances de base de données Aurora Serverless v2 et supprimez les instances de base de données provisionnées. Pour obtenir la procédure complète et des exemples, consultez Basculement d'un cluster approvisionné vers Aurora Serverless v2.
Cluster provisionné exécutant Aurora PostgreSQL version 11 ou 12	Effectuez une mise à niveau d'une version majeure vers Aurora PostgreSQL version 13.6 ou ultérieure. Suivez ensuite la procédure propre à Aurora PostgreSQL version 13 pour basculer le cluster de sorte à utiliser les instances de base de données Aurora Serverless v2.
Cluster Aurora Serverless v1 exécutant Aurora PostgreSQL version 11 ou 13	Pour vous aider à planifier votre conversion depuis Aurora Serverless v1, commencez par consulter Comparaison d'Aurora Serverless v2 avec Aurora Serverless v1. Suivez ensuite la procédure décrite dans Mise à niveau d'un cluster Aurora Serverless v1 vers Aurora Serverless v2.

Basculement d’un cluster approvisionné vers Aurora Serverless v2

Pour basculer un cluster approvisionné pour utiliser Aurora Serverless v2, procédez comme suit :

1. Vérifiez si le cluster approvisionné doit être mis à niveau pour être utilisé avec les instances de base de données Aurora Serverless v2. Pour connaître les versions d’Aurora compatibles avec Aurora Serverless v2, consultez Exigences et limites relatives à Aurora Serverless v2.

   Si le cluster approvisionné exécute une version de moteur qui n’est pas disponible pour Aurora Serverless v2, mettez à niveau la version de moteur du cluster :

   - Si vous disposez d’un cluster provisionné compatible avec MySQL 5.7, suivez les instructions de mise à niveau pour Aurora MySQL version 3. Procédez comme décrit à la rubrique Comment effectuer une mise à niveau sur place.
   - Si vous disposez d’un cluster provisionné compatible avec PostgreSQL exécutant PostgreSQL version 11 ou 12, suivez les instructions de mise à niveau pour Aurora PostgreSQL version 13. Procédez comme décrit à la rubrique Réalisation d’une mise à niveau de version majeure.
2. Configurez toutes les autres propriétés du cluster de sorte qu’elles satisfassent aux exigences Aurora Serverless v2 de la rubrique Exigences et limites relatives à Aurora Serverless v2.
3. Définissez la configuration de mise à l’échelle du cluster. Suivez la procédure décrite dans Définition de la plage de capacité Aurora Serverless v2 d’un cluster.
4. Ajoutez une ou plusieurs instances de base de données Aurora Serverless v2 au cluster. Suivez la procédure générale de la rubrique Ajout de réplicas Aurora à un cluster de bases de données. Pour chaque nouvelle instance de base de données, spécifiez le nom de classe d'instance de base de données spécial Serverless dans ou `db.serverless` dans l' AWS CLI API Amazon RDS. AWS Management Console

   Dans certains cas, le cluster peut déjà comporter une ou plusieurs instances de base de données de lecteur approvisionnées. Si tel est le cas, vous pouvez convertir l’un des lecteurs en instance de base de données Aurora Serverless v2 au lieu de créer une instance de base de données. Pour ce faire, suivez la procédure décrite dans Conversion d’un lecteur ou d’un enregistreur approvisionné en Aurora Serverless v2.
5. Effectuez une opération de basculement pour faire de l’une des instances de base de données Aurora Serverless v2 l’instance de base de données d’enregistreur du cluster.
6. (Facultatif) Convertissez toutes les instances de base de données approvisionnées en Aurora Serverless v2 ou retirez-les du cluster. Suivez la procédure générale décrite à la rubrique

Conversion d’un lecteur ou d’un enregistreur approvisionné en Aurora Serverless v2 ou Suppression d’une instance de base de données d’un cluster de bases de données Aurora.

> **Tip**
>
> Le retrait d’instances de base de données approvisionnées n’est pas obligatoire. Vous pouvez configurer un cluster contenant à la fois des instances de base de données Aurora Serverless v2 et approvisionnées. Cependant, tant que vous n’êtes pas familiarisé avec les caractéristiques de performance et de mise à l’échelle des instances de base de données Aurora Serverless v2, nous vous recommandons de configurer vos clusters avec des instances de base de données du même type.

L' AWS CLI exemple suivant montre le processus de commutation à l'aide d'un cluster provisionné qui exécute Aurora MySQL version 3.02.0. Le cluster se nomme `mysql-80`. Il commence par deux instances de base de données approvisionnées nommées `provisioned-instance-1` et `provisioned-instance-2`, l’une enregistreur, l’autre lecteur. Elles utilisent toutes les deux la classe d’instance de base de données `db.r6g.large`.

```
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' --output text
mysql-80
provisioned-instance-2     False
provisioned-instance-1     True

$ aws rds describe-db-instances --db-instance-identifier provisioned-instance-1 \
  --output text --query '*[].[DBInstanceIdentifier,DBInstanceClass]'
provisioned-instance-1     db.r6g.large

$ aws rds describe-db-instances --db-instance-identifier provisioned-instance-2 \
  --output text --query '*[].[DBInstanceIdentifier,DBInstanceClass]'
provisioned-instance-2     db.r6g.large
```

Créons une table contenant des données. Nous pouvons ainsi confirmer que les données et le fonctionnement du cluster sont identiques avant et après la bascule.

```
mysql> create database serverless_v2_demo;
mysql> create table serverless_v2_demo.demo (s varchar(128));
```

```
mysql> insert into serverless_v2_demo.demo values ('This cluster started with a
 provisioned writer.');
Query OK, 1 row affected (0.02 sec)
```

Commençons par ajouter une plage de capacité au cluster. Dans le cas contraire, nous obtiendrions une erreur lors de l'ajout d'instances de base de données Aurora Serverless v2 au cluster. Si nous utilisons le AWS Management Console pour cette procédure, cette étape est automatique lorsque nous ajoutons la première Aurora Serverless v2 instance de base de données.

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-1 \
  --db-cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-
mysql

An error occurred (InvalidDBClusterStateFault) when calling the CreateDBInstance
 operation:
Set the Serverless v2 scaling configuration on the parent DB cluster before creating a
 Serverless v2 DB instance.

$ # The blank ServerlessV2ScalingConfiguration attribute confirms that the cluster
 doesn't have a capacity range set yet.
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 --query
 'DBClusters[*].ServerlessV2ScalingConfiguration'
[]

$ aws rds modify-db-cluster --db-cluster-identifier mysql-80 \
  --serverless-v2-scaling-configuration MinCapacity=0.5,MaxCapacity=16
{
  "DBClusterIdentifier": "mysql-80",
  "ServerlessV2ScalingConfiguration": {
    "MinCapacity": 0.5,
    "MaxCapacity": 16
  }
}
```

Nous créons deux lecteurs Aurora Serverless v2 pour remplacer les instances de base de données d'origine. Pour cela, nous spécifions la classe d'instance de base de données `db.serverless` pour les nouvelles instances de base de données.

```
$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-1 --db-
cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-mysql
{
  "DBInstanceIdentifier": "serverless-v2-instance-1",
```

```
  "DBClusterIdentifier": "mysql-80",
  "DBInstanceClass": "db.serverless",
  "DBInstanceStatus": "creating"
}

$ aws rds create-db-instance --db-instance-identifier serverless-v2-instance-2 \
  --db-cluster-identifier mysql-80 --db-instance-class db.serverless --engine aurora-
mysql
{
  "DBInstanceIdentifier": "serverless-v2-instance-2",
  "DBClusterIdentifier": "mysql-80",
  "DBInstanceClass": "db.serverless",
  "DBInstanceStatus": "creating"
}

$ # Wait for both DB instances to finish being created before proceeding.
$ aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-1
 && \
  aws rds wait db-instance-available --db-instance-identifier serverless-v2-instance-2
```

Nous effectuons un basculement pour faire de l'une des instances de base de données Aurora Serverless v2 le nouvel enregistreur du cluster.

```
$ aws rds failover-db-cluster --db-cluster-identifier mysql-80 \
  --target-db-instance-identifier serverless-v2-instance-1
{
  "DBClusterIdentifier": "mysql-80",
  "DBClusterMembers": [
    {
      "DBInstanceIdentifier": "serverless-v2-instance-1",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "serverless-v2-instance-2",
      "IsClusterWriter": false,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "provisioned-instance-2",
      "IsClusterWriter": false,
```

```
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    },
    {
      "DBInstanceIdentifier": "provisioned-instance-1",
      "IsClusterWriter": true,
      "DBClusterParameterGroupStatus": "in-sync",
      "PromotionTier": 1
    }
  ],
  "Status": "available"
}
```

Il faut compter quelques secondes pour que cette modification soit appliquée. À ce stade, nous disposons d'un enregistreur Aurora Serverless v2 et d'un lecteur Aurora Serverless v2. Par conséquent, nous n'avons besoin d'aucune des instances de base de données approvisionnées d'origine.

```
$ aws rds describe-db-clusters --db-cluster-identifier mysql-80 \
  --query '*[].[DBClusterIdentifier,DBClusterMembers[*].
[DBInstanceIdentifier,IsClusterWriter]]' \
  --output text
mysql-80
serverless-v2-instance-1        True
serverless-v2-instance-2        False
provisioned-instance-2          False
provisioned-instance-1          False
```

La dernière étape de la procédure de bascule consiste à supprimer les deux instances de base de données approvisionnées.

```
$ aws rds delete-db-instance --db-instance-identifier provisioned-instance-2 --skip-
final-snapshot
{
  "DBInstanceIdentifier": "provisioned-instance-2",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql",
  "EngineVersion": "8.0.mysql_aurora.3.02.0",
  "DBInstanceClass": "db.r6g.large"
}
```

```
$ aws rds delete-db-instance --db-instance-identifier provisioned-instance-1 --skip-
final-snapshot
{
  "DBInstanceIdentifier": "provisioned-instance-1",
  "DBInstanceStatus": "deleting",
  "Engine": "aurora-mysql",
  "EngineVersion": "8.0.mysql_aurora.3.02.0",
  "DBInstanceClass": "db.r6g.large"
}
```

À titre de vérification finale, nous confirmons que la table d'origine est accessible et accessible en écriture à partir de l'instance de base de données d'enregistreur Aurora Serverless v2.

```
mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
+--------------------------------------------------+
1 row in set (0.00 sec)

mysql> insert into serverless_v2_demo.demo values ('And it finished with a Serverless
 v2 writer.');
Query OK, 1 row affected (0.01 sec)

mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
| And it finished with a Serverless v2 writer.     |
+--------------------------------------------------+
2 rows in set (0.01 sec)
```

Nous nous connectons également à l'instance de base de données de lecteur Aurora Serverless v2 et confirmons que les données récemment écrites y sont également disponibles.

```
mysql> select * from serverless_v2_demo.demo;
+--------------------------------------------------+
| s                                                |
+--------------------------------------------------+
| This cluster started with a provisioned writer.  |
```

```
| And it finished with a Serverless v2 writer.      |
+---------------------------------------------------+
2 rows in set (0.01 sec)
```

Comparaison d'Aurora Serverless v2 avec Aurora Serverless v1

Si vous utilisez déjà Aurora Serverless v1, vous pouvez découvrir les principales différences entre Aurora Serverless v1 et Aurora Serverless v2. Les différences d'architecture, telles que la prise en charge des instances de base de données de lecteur, établissent de nouveaux types de cas d'utilisation.

Vous pouvez utiliser les tableaux suivants pour mieux comprendre les différences les plus importantes entre Aurora Serverless v2 et Aurora Serverless v1.

Rubriques

- Comparaison des exigences d'Aurora Serverless v2 et d'Aurora Serverless v1
- Comparaison de la mise à l'échelle et de la disponibilité d'Aurora Serverless v2 et d'Aurora Serverless v1
- Comparaison de la prise en charge des fonctionnalités d'Aurora Serverless v2 et d'Aurora Serverless v1
- Adaptation des cas d'utilisation Aurora Serverless v1 à Aurora Serverless v2

Comparaison des exigences d'Aurora Serverless v2 et d'Aurora Serverless v1

Le tableau ci-dessous récapitule les différentes exigences pour exécuter votre base de données à l'aide d'Aurora Serverless v2 ou d'Aurora Serverless v1. Aurora Serverless v2 offre des versions supérieures des moteurs de base de données Aurora MySQL et Aurora PostgreSQL par rapport à Aurora Serverless v1.

Fonctionnalité	Exigence Aurora Serverless v2	Exigence Aurora Serverless v1
Moteurs de base de données	Aurora MySQL, Aurora PostgreSQL	Aurora MySQL, Aurora PostgreSQL
Versions d'Aurora MySQL prises en charge	Consultez Aurora Serverless v2 avec Aurora MySQL.	Consultez Aurora Serverless v1 avec Aurora MySQL.

Fonctionnalité	Exigence Aurora Serverless v2	Exigence Aurora Serverless v1
Versions d'Aurora PostgreSQL prises en charge	Consultez Aurora Serverless v2 avec Aurora PostgreSQL.	Consultez Aurora Serverless v1 avec Aurora PostgreSQL.
Mise à niveau d'un cluster de bases de données	Comme pour les clusters de bases de données provisionnés, vous pouvez effectuer des mises à niveau manuellement sans attendre qu'Aurora mette à niveau le cluster de bases de données pour vous. Pour de plus amples informations, veuillez consulter Modification d'un cluster de bases de données Amazon Aurora. **Note** Pour effectuer une mise à niveau de version majeure de 13.x vers 14.x ou 15.x pour un cluster de base de données compatible Aurora PostgreSQL, la capacité maximale de votre cluster doit être d'au moins 2. ACUs	Les mises à jour des versions mineures sont appliquées automatiquement dès qu'elles sont disponibles. Pour plus d'informations, consultez Versions de moteur de base de données Aurora Serverless v1 et Aurora. Vous pouvez effectuer des mises à niveau de versions majeures manuellement. Pour plus d'informations, consultez Modification d'un cluster de bases de données Aurora Serverless v1.

Fonctionnalité	Exigence Aurora Serverless v2	Exigence Aurora Serverless v1
Conversion à partir d'un cluster de bases de données approvisionné	Vous pouvez utiliser les méthodes suivantes : • Ajouter une ou plusieurs instances de base de données de lecteur Aurora Serverless v2 à un cluster approvisionné existant. Pour utiliser Aurora Serverless v2 pour l'enregistreur, effectuez un basculement vers l'une des instances de base de données Aurora Serverless v2. Pour que l'ensemble du cluster utilise les instances de base de données Aurora Serverless v2, retirez toutes les instances de base de données d'enregistreur approvisionnées après avoir promu l'instance de base de données Aurora Serverless v2 en enregistreur. • Créer un cluster avec le moteur de base de données et la version du moteur appropriés. Utilisez l'une des méthodes standard. Par exemple, restaurez un instantané de cluster ou créez un clone d'un cluster existant. Choisissez Aurora Serverless v2 pour certaines ou la totalité des instances	Restaurer l'instantané du cluster approvisionné afin de créer un cluster Aurora Serverless v1.

Fonctionnalité	Exigence Aurora Serverless v2	Exigence Aurora Serverless v1
	de base de données du nouveau cluster. Si vous créez le cluster via le clonage, vous ne pouvez pas mettre à niveau la version du moteur en même temps. Assurez-vous que le cluster d'origine exécute déjà une version de moteur compatible avec Aurora Serverless v2.	
Conversion à partir d'un cluster Aurora Serverless v1	Suivez la procédure décrite dans Mise à niveau d'un cluster Aurora Serverless v1 vers Aurora Serverless v2.	Ne s'applique pas
Classes d'instance de base de données disponibles	La classe d'instance de base de données spéciale `db.serverless` . Dans le AWS Management Console, il est étiqueté Serverless.	Non applicable. Aurora Serverless v1 utilise le mode moteur `serverless` .
Port	Tous les ports compatibles avec MySQL ou PostgreSQL	Port MySQL ou PostgreSQL par défaut uniquement
Adresse IP publique autorisée ?	Oui	Non

Fonctionnalité	Exigence Aurora Serverless v2	Exigence Aurora Serverless v1
Cloud privé virtuel (VPC) requis ?	Oui	Oui. Chaque cluster Aurora Serverless v1 consomme 2 points de terminaison d'équilibreur de charge d'interface et de passerelle alloués à votre VPC.

Comparaison de la mise à l'échelle et de la disponibilité d'Aurora Serverless v2 et d'Aurora Serverless v1

Le tableau suivant résume les différences entre Aurora Serverless v2 et Aurora Serverless v1 en termes de capacité de mise à l'échelle et de disponibilité.

La mise à l'échelle d'Aurora Serverless v2 est plus réactive, plus granulaire et moins perturbatrice que la mise à l'échelle d'Aurora Serverless v1. Aurora Serverless v2 peut faire l'objet d'une mise à l'échelle en modifiant la taille de l'instance de base de données et en ajoutant des instances de base de données supplémentaires au cluster de bases de données. Il peut également évoluer en ajoutant des clusters dans d'autres Régions AWS à une base de données globale Aurora. En revanche, la mise à l'échelle d'Aurora Serverless v1 n'est appliquée qu'en augmentant ou en diminuant la capacité de l'enregistreur. L'ensemble de la capacité de calcul d'un cluster Aurora Serverless v1 s'exécute dans une seule zone de disponibilité et une seule Région AWS.

Fonctionnalité de mise à l'échelle et de haute disponibilité	Comportement d'Aurora Serverless v2	Comportement d'Aurora Serverless v1
Unités de capacité minimale d'Aurora (ACUs) (Aurora MySQL)	0,5 lorsque le cluster est en cours d'exécution, 0 lorsque le cluster est en pause.	1 lorsque le cluster est en cours d'exécution, 0 lorsque le cluster est en pause.
Minimum ACUs (Aurora PostgreSQL)	0,5 lorsque le cluster est en cours d'exécution, 0 lorsque le cluster est en pause.	2 lorsque le cluster est en cours d'exécution, 0 lorsque le cluster est en pause.

Fonctionnalité de mise à l'échelle et de haute disponibilité	Comportement d'Aurora Serverless v2	Comportement d'Aurora Serverless v1
Maximum ACUs (Aurora MySQL)	256	256
Maximum ACUs (Aurora PostgreSQL)	256	384
Arrêt d'un cluster	Vous pouvez arrêter et démarrer manuellement le cluster à l'aide de la même fonction d'arrêt et de démarrage du cluster que les clusters approvisionnés.	Le cluster est mis en pause automatiquement après un certain délai. Sa disponibilité prend un certain temps lorsque l'activité reprend.
Mise à l'échelle d'instances de base de données	Augmentez ou diminuez avec un incrément minimum de 0,5 ACUs.	Augmentez ou diminuez en doublant ou en réduisant de moitié le. ACUs
Nombre d'instances de base de données	Identique à un cluster approvisionné : 1 instance de base de données d'enregistreur, jusqu'à 15 instances de base de données de lecteur.	1 instance de base de données gérant à la fois les lectures et les écritures.
La mise à l'échelle peut intervenir pendant l'exécution des instructions SQL ?	Oui. Aurora Serverless v2 n'exige pas d'attendre un point silencieux.	Non. Par exemple, la mise à l'échelle attend que les transactions de longue durée, les tables temporaires et les verrous de table se terminent.
Les instances de base de données de lecteur sont mises à l'échelle en même temps que l'enregistreur	Facultatif	Ne s'applique pas

Fonctionnalité de mise à l'échelle et de haute disponibilité	Comportement d'Aurora Serverless v2	Comportement d'Aurora Serverless v1
Stockage maximum	128 Tio	128 Tio
Cache de mémoire tampon conservé lors de la mise à l'échelle	Oui. Le cache de mémoire tampon est redimensionné dynamiquement.	Non. Le cache de mémoire tampon est rechargé après la mise à l'échelle.
Basculement	Oui, comme pour les clusters approvisionnés.	Seulement dans la mesure du possible, sous réserve de disponibilité de la capacité. Plus lent que dans Aurora Serverless v2.
Fonctionnalité multi-AZ	Oui, comme pour les clusters approvisionnés. Un cluster multi-AZ exige une instance de base de données de lecteur dans une deuxième zone de disponibilité. Pour un cluster multi-AZ, Aurora effectue un basculement multi-AZ en cas de défaillance de la zone de disponibilité.	Les clusters Aurora Serverless v1 exécutent l'ensemble de leurs calculs dans une seule zone de disponibilité. La reprise en cas de défaillance de la zone de disponibilité est effectuée dans la mesure du possible seulement et sous réserve de disponibilité de la capacité.
Bases de données globales Aurora	Oui	Non
Mise à l'échelle en fonction de la sollicitation de la mémoire	Oui	Non
Mise à l'échelle en fonction de la charge du processeur	Oui	Oui

Fonctionnalité de mise à l'échelle et de haute disponibilité	Comportement d'Aurora Serverless v2	Comportement d'Aurora Serverless v1
Mise à l'échelle en fonction du trafic réseau	Oui, en fonction de la charge de mémoire et d'UC du trafic réseau. Le paramètre `max_connections` reste constant pour éviter l'interruption des connexions lors de la réduction d'échelle.	Oui, selon le nombre de connexions.
Action de délai d'attente pour les événements de mise à l'échelle	Non	Oui
Ajouter de nouvelles instances de base de données au cluster via AWS Auto Scaling	Non applicable. Vous pouvez créer des instances de base de données de lecteur Aurora Serverless v2 aux niveaux de promotion 2 à 15 en conservant leur faible capacité.	Non. Les instances de base de données de lecteur ne sont pas disponibles.

Comparaison de la prise en charge des fonctionnalités d'Aurora Serverless v2 et d'Aurora Serverless v1

Le tableau ci-dessous résume les éléments suivants :

- Fonctionnalités qui sont disponibles dans Aurora Serverless v2 mais pas dans Aurora Serverless v1
- Fonctionnalités qui fonctionnent différemment entre Aurora Serverless v1 et Aurora Serverless v2
- Fonctionnalités qui ne sont actuellement pas disponibles dans Aurora Serverless v2

Aurora Serverless v2 inclut de nombreuses fonctionnalités issues de clusters approvisionnés qui ne sont pas disponibles pour Aurora Serverless v1.

Fonctionnalité	Prise en charge de Aurora Serverless v2	Prise en charge de Aurora Serverless v1
Topologie de cluster	Aurora Serverless v2 est une propriété des instances de base de données individuelles. Un cluster peut contenir plusieurs instances de base de données Aurora Serverless v2, ou une combinaison d'instances de base de données Aurora Serverless v2 et approvisionnées.	Les clusters Aurora Serverless v1 n'utilisent pas la notion d'instances de base de données. Vous ne pouvez pas modifier la propriété Aurora Serverless v1 après avoir créé le cluster.
Paramètres de configuration	Presque tous les paramètres peuvent être modifiés comme dans les clusters approvisionnés. Pour en savoir plus, consultez Utilisation des groupes de paramètres pour Aurora Serverless v2.	Seul un sous-ensemble de paramètres peut être modifié.
Groupes de paramètres	Groupe de paramètres de cluster et groupes de paramètres de base de données. Les paramètres ayant la valeur `provisioned` dans l'attribut `SupportedEngineModes` sont disponibles. C'est beaucoup plus de paramètres que dans Aurora Serverless v1.	Groupe de paramètres de cluster uniquement. Les paramètres ayant la valeur `serverless` dans l'attribut `SupportedEngineModes` sont disponibles.
Chiffrement du volume du cluster	Facultatif	Obligatoire. Les Limitations des clusters de bases de données chiffrées Amazon Aurora s'appliquent à

Fonctionnalité	Prise en charge de Aurora Serverless v2	Prise en charge de Aurora Serverless v1
		l'ensemble des clusters Aurora Serverless v1.
Instantanés entre régions	Oui	Le snapshot doit être chiffré avec votre propre clé AWS Key Management Service (AWS KMS).
Sauvegardes automatiques conservées après la suppression du cluster de bases de données	Oui	Non
TLS/SSL	Oui. La prise en charge est la même que pour les clusters approvisionnés. Pour plus d'informations, consultez Utilisation de TLS/SSL avec Aurora Serverless v2.	Oui. Il existe certaines différences de prise en charge de TLS pour les clusters approvisionnés. Pour plus d'informations, consultez Utilisation de TLS/SSL avec Aurora Serverless v1.
Clonage	Uniquement depuis et vers les versions de moteur de base de données compatibles avec Aurora Serverless v2. Vous ne pouvez pas utiliser le clonage pour mettre à niveau Aurora Serverless v1 ou une version antérieure d'un cluster approvisionné.	Uniquement depuis et vers les versions de moteur de base de données compatibles avec Aurora Serverless v1.
Intégration à Amazon S3	Oui	Oui
Intégration avec AWS Secrets Manager	Oui	Non

Fonctionnalité	Prise en charge de Aurora Serverless v2	Prise en charge de Aurora Serverless v1
Exportation d’instantanés de cluster de bases de données vers S3	Oui	Non
Association d’un rôle IAM	Oui	Non
Téléchargement de journaux sur Amazon CloudWatch	Facultatif. Vous choisissez les journaux à activer et les journaux vers lesquels vous souhaitez les télécharger CloudWatch.	Tous les journaux activés sont chargés CloudWatch automatiquement.
API de données disponible	Oui	Oui
Éditeur de requête disponible	Oui	Oui
Performance Insights	Oui	Non
Proxy Amazon RDS disponible	Oui	Non
Babelfish for Aurora PostgreSQL disponible	Oui. Pris en charge pour les versions d’Aurora PostgreSQL compatibles avec Babelfish et Aurora Serverless v2.	Non

Adaptation des cas d’utilisation Aurora Serverless v1 à Aurora Serverless v2

Selon votre cas d’utilisation pour Aurora Serverless v1, vous pouvez adapter cette approche pour tirer parti des fonctionnalités d’Aurora Serverless v2, comme suit.

Supposons que vous disposiez d’un cluster Aurora Serverless v1 à faible charge et que votre priorité est de maintenir une disponibilité continue tout en réduisant les coûts. Avec Aurora Serverless v2, vous pouvez réduire le nombre minimal d’ACU à 0,5, comparé au nombre minimal d’ACU de 1 pour Aurora Serverless v1. Vous pouvez augmenter la disponibilité en créant une configuration multi-AZ, l'instance de base de données du lecteur ayant également un minimum de 0,5 ACUs.

Supposons que vous disposiez d'un cluster Aurora Serverless v1 que vous utilisez dans un scénario de développement et de test. Dans ce cas, le coût est également une priorité élevée, mais le cluster n'a pas besoin d'être disponible en permanence. Aurora Serverless v2 peut automatiquement mettre en pause chaque instance lorsqu'elle est complètement inactive. Pour ce faire, spécifiez une capacité minimale de 0 ACUs pour le cluster, comme expliqué dansRéduction verticale à zéro ACU avec pause et reprise automatiques pour Aurora Serverless v2. Vous pouvez aussi arrêter manuellement le cluster lorsqu'il n'est pas nécessaire, et le démarrer au moment du prochain cycle de test ou de développement.

Supposons que vous disposiez d'un cluster Aurora Serverless v1 dont la charge de travail est importante. Un cluster équivalent qui utilise Aurora Serverless v2 peut être mis à l'échelle avec davantage de granularité. Par exemple, Aurora Serverless v1 elle évolue en doublant la capacité, par exemple de 64 à 128 ACUs. En revanche, votre instance de base de données Aurora Serverless v2 peut être mise à l'échelle par incréments de 0,5 ACU.

Supposons que votre charge de travail exige une capacité totale supérieure à celle disponible dans Aurora Serverless v1. Vous pouvez utiliser plusieurs instances de base de données de lecteur Aurora Serverless v2 pour décharger les parties de la charge de travail exigeantes en lecture de l'instance de base de données d'enregistreur. Vous pouvez également répartir la charge de travail exigeante en lecture entre plusieurs instances de base de données de lecteur.

Pour une charge de travail exigeante en écriture, vous pouvez configurer le cluster avec une instance de base de données approvisionnée volumineuse en tant qu'enregistreur. Vous pouvez le faire en même temps qu'une ou plusieurs instances de base de données de lecteur Aurora Serverless v2.

Mise à niveau d'un cluster Aurora Serverless v1 vers Aurora Serverless v2

> **Important**
>
> AWS a annoncé la end-of-life date du Aurora Serverless v1 : 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

Le processus de mise à niveau d'un cluster de bases de données d'Aurora Serverless v1 vers Aurora Serverless v2 se compose de plusieurs étapes. Cela s'explique par le fait qu'il n'est pas possible de convertir directement de Aurora Serverless v1 vers Aurora Serverless v2. Il existe toujours une étape intermédiaire qui implique de convertir le cluster de bases de données Aurora Serverless v1 en cluster provisionné.

Clusters de bases de données compatibles avec Aurora MySQL

Vous pouvez convertir votre Aurora Serverless v1 cluster de base de données en cluster de base de données provisionné, puis utiliser un blue/green déploiement pour le mettre à niveau et le convertir en Aurora Serverless v2 cluster de base de données. Nous recommandons cette procédure pour les environnements de production. Pour de plus amples informations, veuillez consulter Utilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données.

Pour utiliser un blue/green déploiement pour mettre à niveau un Aurora Serverless v1 cluster exécutant Aurora MySQL version 2 (compatible avec MySQL 5.7)

1. Convertissez le cluster de bases de données Aurora Serverless v1 en un cluster Aurora MySQL version 2 provisionné. Suivez la procédure décrite dans Conversion d'Aurora Serverless v1 en mode approvisionné.
2. Créez un blue/green déploiement. Suivez la procédure décrite dans Création d'un blue/green déploiement dans).
3. Choisissez une version d'Aurora MySQL compatible avec Aurora Serverless v2, par exemple la version 3.04.1, pour le cluster vert.

   Pour les versions compatibles, consultez Aurora Serverless v2 avec Aurora MySQL.
4. Modifiez l'instance de base de données d'enregistreur du cluster vert pour utiliser la classe d'instance de base de données Sans serveur v2 (db.serverless).

   Pour en savoir plus, consultez Conversion d'un lecteur ou d'un enregistreur approvisionné en Aurora Serverless v2.
5. Lorsque votre cluster de bases de données Aurora Serverless v2 mis à niveau est disponible, passez du cluster bleu au cluster vert.

Clusters de bases de données compatibles avec Aurora PostgreSQL

Vous pouvez convertir votre Aurora Serverless v1 cluster de base de données en cluster de base de données provisionné, puis utiliser un blue/green déploiement pour le mettre à niveau et le convertir

en Aurora Serverless v2 cluster de base de données. Nous recommandons cette procédure pour les environnements de production. Pour de plus amples informations, veuillez consulter Utilisation d' (Amazon Aurora Blue/Green Deployments) pour les mises à jour de bases de données.

Pour utiliser un blue/green déploiement pour mettre à niveau un Aurora Serverless v1 cluster exécutant Aurora PostgreSQL version 11

1. Convertissez le cluster de bases de données Aurora Serverless v1 en un cluster Aurora PostgreSQL provisionné. Suivez la procédure décrite dans Conversion d'Aurora Serverless v1 en mode approvisionné.
2. Créez un blue/green déploiement. Suivez la procédure décrite dans Création d'un blue/green déploiement dans).
3. Choisissez une version d'Aurora PostgreSQL compatible avec Aurora Serverless v2, par exemple la version 15.3, pour le cluster vert.

   Pour les versions compatibles, consultez Aurora Serverless v2 avec Aurora PostgreSQL.
4. Modifiez l'instance de base de données d'enregistreur du cluster vert pour utiliser la classe d'instance de base de données Sans serveur v2 (db.serverless).

   Pour en savoir plus, consultez Conversion d'un lecteur ou d'un enregistreur approvisionné en Aurora Serverless v2.
5. Lorsque votre cluster de bases de données Aurora Serverless v2 mis à niveau est disponible, passez du cluster bleu au cluster vert.

Vous pouvez également mettre directement à niveau votre cluster de bases de données Aurora Serverless v1 de la version 11 d'Aurora PostgreSQL vers la version 13, le convertir en cluster de bases de données provisionné, puis convertir le cluster provisionné en cluster de bases de données Aurora Serverless v2.

Pour mettre à niveau, puis convertir un cluster Aurora Serverless v1 exécutant Aurora PostgreSQL version 11

1. Convertissez le cluster de bases de données Aurora Serverless v1 en un cluster Aurora PostgreSQL provisionné. Suivez la procédure décrite dans Conversion d'Aurora Serverless v1 en mode approvisionné.

2. Mettez à niveau le cluster Aurora Serverless v1 vers une version 13 d'Aurora PostgreSQL compatible avec Aurora Serverless v2, par exemple 13.12. Suivez la procédure décrite dans Mise à niveau de la version majeure.

   Pour les versions compatibles, consultez Aurora Serverless v2 avec Aurora PostgreSQL.

3. Ajoutez une instance de base de données de lecteur Aurora Serverless v2 au cluster. Pour plus d'informations, consultez Ajout d'un lecteur Aurora Serverless v2.
4. Basculez vers l'instance de base de données Aurora Serverless v2 :

   a. Choisissez l'instance de base de données d'enregistreur du cluster de bases de données.

   b. Pour Actions, choisissez Failover (Basculement).

   c. Sur la page de confirmation, choisissez Basculement.

Pour les clusters de bases de données Aurora Serverless v1 exécutant Aurora PostgreSQL version 13, convertissez le cluster Aurora Serverless v1 en cluster de bases de données provisionné, puis vous convertissez le cluster provisionné en cluster de bases de données Aurora Serverless v2.

Pour mettre à niveau un cluster Aurora Serverless v1 exécutant Aurora PostgreSQL version 13

1. Convertissez le cluster de bases de données Aurora Serverless v1 en un cluster Aurora PostgreSQL provisionné. Suivez la procédure décrite dans Conversion d'Aurora Serverless v1 en mode approvisionné.
2. Ajoutez une instance de base de données de lecteur Aurora Serverless v2 au cluster. Pour plus d'informations, consultez Ajout d'un lecteur Aurora Serverless v2.
3. Basculez vers l'instance de base de données Aurora Serverless v2 :

   a. Choisissez l'instance de base de données d'enregistreur du cluster de bases de données.

   b. Pour Actions, choisissez Failover (Basculement).

   c. Sur la page de confirmation, choisissez Basculement.

4. Supprimez l'instance de lecteur.

Migration d'une base de données sur site vers Aurora Serverless v2

Vous pouvez migrer vos bases de données sur site vers Aurora Serverless v2, tout comme avec les bases de données Aurora MySQL et Aurora PostgreSQL provisionnées.

- Pour les bases de données MySQL, vous pouvez utiliser la commande `mysqldump`. Pour plus d'informations, consultez Importation de données vers une base de données Amazon RDS for MySQL avec une durée d'indisponibilité réduite dans le Guide de l'utilisateur Amazon Relational Database Service.
- Pour les bases de données PostgreSQL, vous pouvez utiliser les commandes `pg_dump` et `pg_restore`. Pour obtenir plus d'informations, consultez l'article de blog Best practices for migrating PostgreSQL databases to Amazon RDS and Amazon Aurora (Bonnes pratiques pour la migration des bases de données PostgreSQL vers Amazon RDS et Amazon Aurora).

Utilisation d’Amazon Aurora Serverless v1

> **Important**
>
> AWS a annoncé la date de fin de vie d’Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d’informations, consultez Support étendu RDS.

Amazon Aurora Serverless v1 (Amazon Aurora sans serveur version 1) est une configuration de d’autoscaling à la demande pour Amazon Aurora. Un cluster de bases de données Aurora Serverless v1 est un cluster de bases de données qui permet d’augmenter ou de réduire la capacité en fonction des besoins de votre application. Cela contraste avec des clusters de bases de données approvisionnés Aurora dont vous pouvez gérer manuellement la capacité. Aurora Serverless v1 offre une option relativement simple et rentable pour les charges de travail peu fréquentes, intermittentes ou imprévisibles. Il est économique, car il démarre, augmente ou réduit la capacité de calcul en fonction de l’utilisation de votre application et s’arrête lorsqu’il n’est pas utilisé.

Pour en savoir plus sur la tarification, consultez Tarification sans serveur sous Édition compatible avec MySQL ou Édition compatible avec PostgreSQL sur la page Amazon Aurora pricing.

Les clusters Aurora Serverless v1 ont le même type de volume de stockage haute capacité, distribué et hautement disponible que celui utilisé par les clusters de bases de données alloués.

Le volume d’un cluster Aurora Serverless v1 est toujours chiffré. Vous pouvez choisir la clé de chiffrement, mais vous ne pouvez pas désactiver le chiffrement. Dès lors, vous pouvez effectuer les mêmes opérations sur Aurora Serverless v1 que sur des instantanés chiffrés. Pour plus d’informations, consultez Aurora Serverless v1 et instantanés.

Rubriques

Disponibilité des régions et des versions pour Aurora Serverless v1

La disponibilité et la prise en charge des fonctions varient selon les versions spécifiques de chaque moteur de base de données Aurora, et selon les Régions AWS. Pour en savoir plus sur les versions et la disponibilité des régions avec Aurora et Aurora Serverless v1, consultez Aurora Serverless v1.

Avantages d'Aurora Serverless v1

Aurora Serverless v1 offre les avantages suivants :

- Simplicité – Aurora Serverless v1 élimine une grande partie de la complexité de la gestion des instances et de la capacité des bases de données.
- Évolutivité – Aurora Serverless v1 met à l'échelle sans heurt la capacité de calcul et de mémoire en fonction des besoins, sans perturber les connexions client.
- Rentabilité – Quand vous utilisez Aurora Serverless v1, vous payez uniquement pour les ressources de bases de données que vous consommez, à la seconde.
- Stockage hautement disponible – Pour assurer une protection contre la perte de données, Aurora Serverless v1 utilise le même système de stockage distribué et tolérant aux pannes avec réplication à six voies qu'Aurora.

Cas d’utilisation pour Aurora Serverless v1

Aurora Serverless v1 est conçu pour les cas d’utilisation suivants :

- Applications rarement utilisées – Vous avez une application qui n’est utilisée que quelques minutes plusieurs fois par jour ou par semaine, comme un site de blog peu volumineux. Avec Aurora Serverless v1, vous payez uniquement les ressources de base de données que vous consommez, à la seconde.
- Nouvelles applications – Vous déployez une nouvelle application et vous n’êtes pas sûr de la taille de l’instance dont vous avez besoin. Avec Aurora Serverless v1, vous pouvez créer un point de terminaison de base de données et faire en sorte que la base de données se mette à l’échelle automatiquement selon les besoins en capacité de votre application.
- Charge de travail variables – Vous exécutez une application peu utilisée, avec des pics de 30 minutes à plusieurs heures quelques fois chaque jour, ou plusieurs fois par an. Il peut s’agir, par exemple, d’applications pour les ressources humaines, la budgétisation et le reporting opérationnel. Avec Aurora Serverless v1, vous n’avez plus besoin d’allouer de la capacité pour les pics ou la moyenne d’utilisation.
- Charges de travail imprévisibles – Vous exécutez des charges de travail quotidiennes présentant des hausses soudaines et imprévisibles en termes d’activité. Par exemple, un site d’informations sur la circulation routière qui pourrait connaître un pic d’activité lorsqu’il commence à pleuvoir. Avec Aurora Serverless v1, la capacité de votre base de données augmente automatiquement pour répondre aux besoins de la charge de pointe de l’application et revient à la normale lorsque la hausse d’activité est terminée.
- Bases de données de développement et de test – Vos développeurs utilisent les bases de données pendant les heures de travail, mais n’en ont pas besoin la nuit ou le week-end. Avec Aurora Serverless v1, votre base de données s’arrête automatiquement lorsqu’elle n’est pas utilisée.
- Applications multilocataires – Avec Aurora Serverless v1, vous n’avez pas à gérer individuellement la capacité de base de données pour chaque application de votre flotte. Aurora Serverless v1 gère la capacité de base de données individuelle pour vous.

Limites d Aurora Serverless v1

Les limites suivantes s’appliquent à Aurora Serverless v1 :

- > **Important**
  >
  > AWS a annoncé la date de fin de vie d'Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

- Aurora Serverless v1 ne prend pas en charge les fonctionnalités suivantes :
  - Bases de données globales Aurora
  - Réplicas Aurora
  - Gestion des identités et des accès AWSAuthentification de base de données (IAM)
  - Retour sur trace dans Aurora
  - Flux d'activité de base de données.
  - Authentification Kerberos
  - Performance Insights
  - RDS Proxy (Proxy RDS)
  - Affichage des journaux dans la AWS Management Console
- Les connexions à un cluster de bases de données Aurora Serverless v1 sont automatiquement fermées si elles restent ouvertes pendant plus d'un jour.
- Tous les clusters de bases de données Aurora Serverless v1 présentent les limites suivantes :
  - Vous ne pouvez pas exporter d'instantanés Aurora Serverless v1 vers des compartiments Amazon S3.
  - Vous ne pouvez pas utiliser AWS Database Migration Service et la capture des données de modification (CDC) avec les clusters de bases de données Aurora Serverless v1. Seuls les clusters de bases de données Aurora alloués prennent en charge les CDC avec AWS DMS en tant que source.
  - Vous ne pouvez pas enregistrer de données dans des fichiers texte dans Amazon S3 ni charger de données de fichiers texte vers Aurora Serverless v1 depuis S3.
  - Vous ne pouvez pas attacher un rôle IAM à un cluster de bases de données Aurora Serverless v1. Cependant, vous pouvez charger des données dans Aurora Serverless v1 depuis Amazon S3 à l'aide de l'extension `aws_s3` avec la fonction `aws_s3.table_import_from_s3` et le

paramètre `credentials`. Pour plus d'informations, consultez Importation de données Amazon S3 dans une d'un cluster de base de données Aurora PostgreSQL.

- Lorsque vous utilisez l'éditeur de requêtes, un secret Secrets Manager est créé afin que les informations d'identification de la base de données puissent accéder à cette dernière. Si vous supprimez les informations d'identification de l'éditeur de requêtes, le secret associé est également supprimé de Secrets Manager. Vous ne pouvez pas récupérer ce secret une fois que vous l'avez supprimé.

- Les clusters de bases de données basés sur Aurora MySQL exécutant Aurora Serverless v1 ne prennent pas en charge les opérations suivantes :
  - Appel de fonctions AWS Lambda à partir votre cluster de bases de données Aurora MySQL. Cependant, les fonctions AWS Lambda peuvent effectuer des appels à votre cluster de bases de données Aurora Serverless v1.
  - Restauration d'un instantané à partir d'une instance de base de données ne correspondant pas à Aurora MySQL ou RDS for MySQL.
  - Réplication de données à l'aide de la réplication basée sur des journaux binaires (binlogs). Cette limitation est vraie, que votre cluster de bases de données Aurora Serverless v1 basé sur Aurora MySQL soit la source ou la cible de la réplication. Pour répliquer des données dans un cluster de bases de données Aurora Serverless v1 à partir d'une instance de base de données MySQL hors d'Aurora, telle qu'une instance s'exécutant sur Amazon EC2, envisagez d'utiliser AWS Database Migration Service. Pour plus d'informations, consultez le Guide de l'utilisateur AWS Database Migration Service.
  - Création d'utilisateurs avec un accès basé sur l'hôte (`'username'@'IP_address'`). En effet, Aurora Serverless v1 utilise une flotte de routeurs entre le client et l'hôte de la base de données pour une mise à l'échelle transparente. L'adresse IP que le cluster de bases de données Aurora Serverless voit est celle de l'hôte du routeur et non celle de votre client. Pour plus d'informations, consultez Architecture d'Aurora Serverless v1.

    Utilisez plutôt le caractère générique (`'username'@'%'`).

- Les clusters de bases de données basés sur Aurora PostgreSQL exécutant Aurora Serverless v1 présentent les limitations suivantes :
  - La gestion du plan de requête Aurora PostgreSQL (extension `apg_plan_management`) n'est pas prise en charge.
  - La fonction de réplication logique disponible dans Amazon RDS PostgreSQL et Aurora PostgreSQL n'est pas prise en charge.

- Les communications sortantes telles que celles activées par les extensions Amazon RDS pour PostgreSQL ne sont pas prises en charge. Par exemple, vous ne pouvez pas accéder aux données externes avec l'extension `postgres_fdw/dblink`. Pour plus d'informations sur les extensions RDS PostgreSQL, consultez PostgreSQL sur Amazon RDS dans le Guide de l'utilisateur Amazon RDS.
- Certaines requêtes et commandes SQL sont actuellement déconseillées. Il s'agit notamment des verrous consultatifs au niveau de la session, des relations temporaires, des notifications asynchrones (`LISTEN`) et des curseurs avec maintien (`DECLARE` *`name`* `... CURSOR WITH HOLD FOR` *`query`*). En outre, les commandes `NOTIFY` empêchent la mise à l'échelle et sont déconseillées.

  Pour plus d'informations, consultez Mise à l'échelle automatique pour Aurora Serverless v1.

- Vous ne pouvez pas définir la fenêtre de sauvegarde automatisée préférée pour un cluster de bases de données Aurora Serverless v1.
- Vous pouvez définir la fenêtre de maintenance pour un cluster de bases de données Aurora Serverless v1. Pour plus d'informations, consultez Ajustement du créneau de maintenance préféré pour un cluster de bases de données.

Exigences en matière de configuration pour Aurora Serverless v1

Lorsque vous créez un cluster de bases de données Aurora Serverless v1, prêtez une attention particulière aux exigences suivantes :

- Utilisez ces numéros de ports spécifiques pour chaque moteur de base de données :
  - Aurora MySQL – `3306`
  - Aurora PostgreSQL – `5432`
- Créez votre cluster de bases de données Aurora Serverless v1 dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC. Lorsque vous créez un cluster de bases de données Aurora Serverless v1 dans votre VPC, vous consommez deux (2) des cinquante (50) points de terminaison d'interface et d'équilibrage de charge de passerelle alloués à votre VPC. Ces points de terminaison sont créés automatiquement pour vous. Pour augmenter votre quota, vous pouvez contacter Support. Pour plus d'informations, consultez Amazon VPC quotas (Quotas Amazon VPC).
- Vous ne pouvez pas donner d'adresse IP publique à un cluster de bases de données Aurora Serverless v1. Vous pouvez accéder à un cluster de bases de données Aurora Serverless v1 uniquement à partir d'un VPC.

- Créez des sous-réseaux dans différentes zones de disponibilité pour le groupe de sous-réseaux de base de données que vous utilisez pour votre cluster de bases de données Aurora Serverless v1. En d'autres termes, vous ne pouvez pas disposer de plusieurs sous-réseaux dans la même zone de disponibilité.
- Les modifications apportées à un groupe de sous-réseaux utilisé par un cluster de bases de données Aurora Serverless v1 ne sont pas appliquées au cluster.
- Vous pouvez accéder à un cluster de bases de données Aurora Serverless v1 à partir d'AWS Lambda. Pour ce faire, vous devez configurer votre fonction Lambda pour qu'elle s'exécute dans le même VPC que votre cluster de bases de données Aurora Serverless v1. Pour plus d'informations sur l'utilisation de AWS Lambda, consultez Configuration d'une fonction Lambda pour accéder aux ressources d'un VPC Amazon dans le Manuel du développeur AWS Lambda.

Utilisation de TLS/SSL avec Aurora Serverless v1

Par défaut, Aurora Serverless v1 utilise le protocole TLS/SSL (Transport Layer Security/Secure Sockets Layer) pour chiffrer les communications entre les clients et votre cluster de bases de données Aurora Serverless v1. Il prend en charge les versions TLS/SSL 1.0, 1.1 et 1.2. Vous n'êtes pas tenu de configurer votre cluster de bases de données Aurora Serverless v1 pour utiliser TLS/SSL.

Cependant, les limites suivantes s'appliquent :

- La prise en charge de TLS/SSL pour les clusters de bases de données Aurora Serverless v1 n'est actuellement pas disponible dans la Région AWS Chine (Pékin).
- Lorsque vous créez des utilisateurs de base de données pour un cluster de bases de données Aurora Serverless v1 basé sur Aurora MySQL, n'utilisez pas la clause `REQUIRE` pour les autorisations SSL. Cela empêche les utilisateurs de se connecter à l'instance de base de données Aurora.
- Pour les utilitaires de client MySQL et de client PostgreSQL, les variables de session que vous pouvez utiliser dans d'autres environnements n'ont aucun effet sur l'utilisation de TLS/SSL entre le client et Aurora Serverless v1.
- Pour le client MySQL, en cas de connexion avec le mode `VERIFY_IDENTITY` TLS/SSL, vous devez actuellement utiliser la commande `mysql` compatible MySQL 8.0. Pour plus d'informations, consultez Connexion à une instance de base de données exécutant le moteur de base de données MySQL.

Selon le client que vous utilisez pour vous connecter au cluster de bases de données Aurora Serverless v1, vous n'êtes pas nécessairement tenu de spécifier TLS/SSL pour obtenir une connexion chiffrée. Par exemple, pour utiliser le client PostgreSQL pour vous connecter à un cluster de bases de données Aurora Serverless v1 exécutant Aurora Édition compatible avec PostgreSQL, connectez-vous comme d'habitude.

```
psql -h endpoint -U user
```

Après avoir entré votre mot de passe, le client PostgreSQL affiche les détails de la connexion, notamment la version TLS/SSL et le chiffrement.

```
psql (12.5 (Ubuntu 12.5-0ubuntu0.20.04.1), server 10.12)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256,
 compression: off)
Type "help" for help.
```

Important

Aurora Serverless v1 utilise le protocole TLS/SSL (Transport Layer Security/Secure Sockets Layer) pour chiffrer les connexions par défaut, à moins que SSL/TLS ne soit désactivé par l'application cliente. La connexion TLS/SSL se termine au niveau de la flotte de routeurs. La communication entre la flotte de routeurs et votre cluster de bases de données Aurora Serverless v1 s'effectue dans la limite du réseau interne du service.

Vous pouvez consulter le statut de la connexion client pour vérifier si la connexion à Aurora Serverless v1 est chiffrée via TLS/SSL. Les tables PostgreSQL `pg_stat_ssl` et `pg_stat_activity`, ainsi que la fonction `ssl_is_used` n'affichent pas l'état TLS/SSL pour la communication entre l'application cliente et Aurora Serverless v1. De même, l'état TLS/SSL ne peut pas provenir de l'instruction `status` MySQL.

Les paramètres de cluster Aurora `force_ssl` pour PostgreSQL et `require_secure_transport` pour MySQL n'étaient pas pris en charge pour Aurora Serverless v1. Ces paramètres sont maintenant disponibles pour Aurora Serverless v1. Pour obtenir la liste complète des paramètres pris en charge par Aurora sans serveur v1, appelez l'opération d'API DescribeEngineDefaultClusterParameters. Pour plus d'informations sur les groupes de paramètres et Aurora sans serveur version 1, consultez Groupes de paramètres pour Aurora Serverless v1.

Pour utiliser le client MySQL pour vous connecter à un cluster de bases de données Aurora Serverless v1 exécutant Aurora Édition compatible avec MySQL, spécifiez TLS/SSL dans votre requête. L'exemple suivant inclut le référentiel d'approbations Amazon Root CA 1 téléchargé à partir d'Amazon Trust Services et nécessaire pour que cette connexion aboutisse.

```
mysql -h endpoint -P 3306 -u user -p --ssl-ca=amazon-root-CA-1.pem --ssl-mode=REQUIRED
```

Lorsque vous y êtes invité, entrez votre mot de passe. Après quoi, la surveillance MySQL s'ouvre. Vous pouvez vérifier que la session est chiffrée à l'aide de la commande `status`.

```
mysql> status
--------------
mysql  Ver 14.14 Distrib 5.5.62, for Linux (x86_64) using readline 5.1
Connection id:          19
Current database:
Current user:           ***@*******
SSL:                    Cipher in use is ECDHE-RSA-AES256-SHA
...
```

Pour en savoir plus sur la connexion à la base de données Aurora MySQL avec le client MySQL, consultez Connexion à une instance de base de données exécutant le moteur de base de données MySQL.

Aurora Serverless v1 prend en charge tous les modes TLS/SSL disponibles pour le client MySQL (`mysql`) et le client PostgreSQL (`psql`), y compris les modes répertoriés dans le tableau suivant.

Description du mode TLS/SSL	mysql	psql
Se connecte sans utiliser TLS/SSL.	DISABLED	désactiver
Essaie de se connecter à l'aide de TLS/SSL, mais revient à non-SSL, si nécessaire.	PREFERRED	prefer (par défaut)
Imposer à l'aide de TLS/SSL.	REQUIRED	require

Description du mode TLS/SSL	mysql	psql
Impose TLS/SSL et vérifie l'autorité de certification.	VERIFY_CA	verify-ca
Impose TLS/SSL, vérifie l'autorité de certification et son nom d'hôte.	VERIFY_IDENTITY	verify-full

Aurora Serverless v1 utilise des certificats à caractères génériques. Si vous spécifiez l'option « Vérifier l'autorité de certification » ou « Vérifier l'autorité de certification et son nom d'hôte » lors de l'utilisation de TLS/SSL, commencez par télécharger le référentiel d'approbations Amazon Root CA 1 à partir d'Amazon Trust Services. Vous pouvez ensuite identifier ce fichier au format PEM dans votre commande client. Pour ce faire, utilisez le client PostgreSQL :

Pour Linux, macOS ou Unix :

```
psql 'host=endpoint user=user sslmode=require sslrootcert=amazon-root-CA-1.pem
 dbname=db-name'
```

Pour en savoir plus sur l'utilisation de la base de données Aurora PostgreSQL à l'aide du client Postgres, consultez Connexion à une instance de base de données exécutant le moteur de base de données PostgreSQL.

Pour plus d'informations sur la connexion aux clusters de bases de données Aurora, consultez Connexion à un cluster de bases de données Amazon Aurora.

Suites de chiffrement prises en charge pour les connexions aux clusters de bases de données Aurora Serverless v1

L'utilisation de suites de chiffrement configurables vous permet d'avoir plus de contrôle sur la sécurité des connexions de vos bases de données. Vous pouvez spécifier une liste de suites de chiffrement que vous souhaitez autoriser pour la sécurisation des connexions TLS/SSL client à votre base de données. Avec les suites de chiffrement configurables, vous pouvez contrôler le chiffrement de connexion accepté par votre serveur de base de données. Cela évite d'utiliser des chiffrements qui ne sont pas sécurisés ou qui ne sont plus utilisés.

Les clusters de bases de données Aurora Serverless v1 basés sur Aurora MySQL prennent en charge les mêmes suites de chiffrement que les clusters de bases de données provisionnés Aurora MySQL. Pour plus d'informations sur ces suites de chiffrement, consultez Configuration de suites de chiffrement pour les connexions aux clusters de bases de données Aurora MySQL.

Les clusters de bases de données Aurora Serverless v1 basés sur Aurora PostgreSQL ne prennent pas en charge les suites de chiffrement.

Fonctionnement d'Aurora Serverless v1

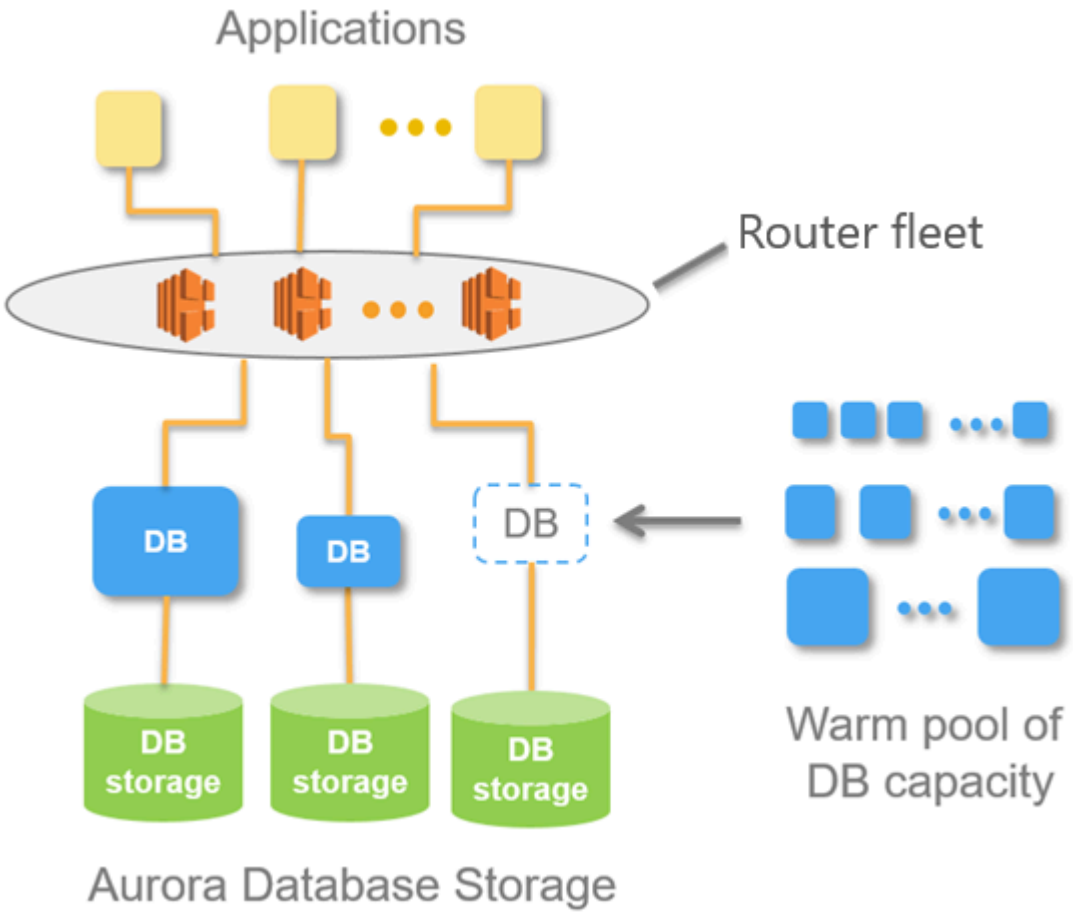
> **Important**
>
> AWS a annoncé la date de fin de vie d'Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

Vous allez pouvoir découvrir comment fonctionne Aurora Serverless v1.

Rubriques

Architecture d’Aurora Serverless v1

L’image suivante illustre l’architecture d’Aurora Serverless v1.



Au lieu d’allouer et de gérer les serveurs de base de données, vous spécifiez des unités de capacité Aurora (ACU). Chaque unité de capacité combine environ 2 gigaoctets (Go) de mémoire, avec une UC et une mise en réseau correspondantes. Le stockage de base de données s’échelonne automatiquement de 10 gibioctets (Gio) jusqu’à 128 tebibytes (Tio), ce qui équivaut au stockage dans un cluster de bases de données Aurora standard.

Vous pouvez définir l’unité de capacité minimale et maximale. L’unité de capacité Aurora minimale est l’unité de capacité la plus petite à laquelle le cluster de bases de données peut être dimensionné. L’unité de capacité Aurora maximale est l’unité de capacité la plus grande à laquelle le cluster de bases de données peut être dimensionné. En fonction de vos paramètres, Aurora Serverless v1 crée automatiquement des règles de mise à l’échelle pour les seuils d’utilisation de l’UC, de connexions et de mémoire disponible.

Aurora Serverless v1 gère le groupe de ressources « à chaud » dans une Région AWS pour minimiser le temps de mise à l'échelle. Quand Aurora Serverless v1 ajoute de nouvelles ressources au cluster de bases de données Aurora, il utilise la flotte de routeurs pour faire basculer les connexions client actives vers les nouvelles ressources. Quelle que soit l'heure, seules les ACU qui sont activement utilisées dans votre cluster de bases de données Aurora vous sont facturées.

Mise à l'échelle automatique pour Aurora Serverless v1

La capacité allouée à votre cluster de bases de données Aurora Serverless v1 est mise à l'échelle de manière transparente en fonction de la charge générée par votre application client. Ici, la charge représente l'utilisation du processeur et le nombre de connexions. Lorsque la capacité est limitée par l'un ou l'autre de ces facteurs, Aurora Serverless v1 se met à l'échelle. Aurora Serverless v1 se met également à l'échelle lorsqu'il détecte des problèmes de performances qui peuvent être résolus en procédant ainsi.

Vous pouvez afficher les événements de mise à l'échelle pour votre cluster Aurora Serverless v1 dans la AWS Management Console. Lors de la mise à l'échelle automatique, Aurora Serverless v1 réinitialise la métrique `EngineUptime`. La valeur de métrique de réinitialisation ne signifie pas que la mise à l'échelle transparente a rencontré des problèmes ou que des connexions Aurora Serverless v1 ont été interrompues. C'est simplement le point de départ de la disponibilité à la nouvelle capacité. Pour en savoir plus sur les métriques, consultez Surveillance des métriques d'un cluster de bases de données Amazon Aurora.

Lorsque votre cluster de bases de données Aurora Serverless v1 n'a pas de connexions actives, il peut se redimensionner jusqu'à la capacité zéro (0 ACU). Pour en savoir plus, consultez Mettre en pause et reprendre pour Aurora Serverless v1.

Lorsqu'il a besoin d'effectuer une opération de mise à l'échelle, Aurora Serverless v1 essaie d'abord d'identifier un point de mise à l'échelle, un moment où aucune requête n'est en cours de traitement. Aurora Serverless v1 peut ne pas être en mesure de trouver un point de mise à l'échelle pour les raisons suivantes :

- Requêtes de longue durée
- Transactions en cours
- Tables temporaires ou verrous de table

Pour augmenter le taux de réussite de votre cluster de bases de données Aurora Serverless v1 lors de la recherche d'un point de mise à l'échelle, nous vous recommandons d'éviter les requêtes

et les transactions de longue durée. Pour en savoir plus sur les opérations de blocage d'échelle et comment les éviter, consultez Best practices for working with Aurora Serverless v1.

Par défaut, Aurora Serverless v1 tente de trouver un point de mise à l'échelle pendant 5 minutes (300 secondes). Vous pouvez spécifier un délai d'attente différent lorsque vous créez ou modifiez le cluster. Le délai d'attente peut être compris entre 60 secondes et 10 minutes (600 secondes). Si Aurora Serverless v1 ne peut pas trouver de point de mise à l'échelle dans la période spécifiée, l'opération de scalabilité automatique expire.

Par défaut, si la mise à l'échelle automatique ne trouve pas de point de mise à l'échelle avant l'expiration, Aurora Serverless v1 maintient le cluster à la capacité actuelle. Vous pouvez modifier ce comportement par défaut lorsque vous créez ou modifiez votre cluster de bases de données Aurora Serverless v1 en sélectionnant l'option Forcer le changement de capacité. Pour plus d'informations, consultez Action de délai d'attente pour les modifications de capacité.

Action de délai d'attente pour les modifications de capacité

Si la scalabilité automatique expire avec la recherche d'un point de mise à l'échelle, Aurora conserve par défaut la capacité actuelle. Vous pouvez faire en sorte que Aurora force le changement en sélectionnant l'option Force the capacity change (Forcer le changement de capacité). Cette option est disponible dans la section Autoscaling timeout and action (Délai de mise à l'échelle automatique et action) de la page Create database (Créer une base de données), lorsque vous créez le cluster.

Par défaut, l'option Force the capacity change (Forcer le changement de capacité) n'est pas sélectionnée. Ne la sélectionnez pas pour que la capacité de votre cluster de bases de données Aurora Serverless v1 reste inchangée si l'opération de mise à l'échelle échoue sans trouver de point de mise à l'échelle.

Si vous choisissez cette option, votre cluster de bases de données Aurora Serverless v1 appliquera le changement de capacité, même sans point de mise à l'échelle. Avant de sélectionner cette option, tenez compte des conséquences de sa sélection :

- Toutes les transactions en cours de processus sont interrompues et le message d'erreur suivant s'affiche.

  Aurora MySQL version 2 – ERREUR 1105 (HY000) : La dernière transaction a été interrompue en raison d'une mise à l'échelle transparente. Veuillez réessayer.

  Vous pouvez renvoyer les transactions dès que votre cluster de bases de données Aurora Serverless v1 est disponible.

- Les connexions aux tables temporaires et aux verrous sont abandonnées.

  Nous vous recommandons de sélectionner l'option Force the capacity change (Forcer le changement de capacité) uniquement si votre application peut récupérer des connexions interrompues ou des transactions incomplètes.

Les choix que vous faites dans AWS Management Console lorsque vous créez un cluster de bases de données Aurora Serverless v1 sont stockés dans l'objet `ScalingConfigurationInfo`, dans les propriétés `SecondsBeforeTimeout` et `TimeoutAction`. La valeur de la propriété `TimeoutAction` est définie sur l'une des valeurs suivantes lorsque vous créez votre cluster :

- `RollbackCapacityChange` – Cette valeur est définie lorsque vous sélectionnez l'option Roll back the capacity change (Restaurer le changement de capacité). Il s'agit du comportement de par défaut.
- `ForceApplyCapacityChange` – Cette valeur est définie lorsque vous sélectionnez l'option Force the capacity change (Forcer le changement de capacité).

Vous pouvez obtenir la valeur de cette propriété sur un cluster de bases de données Aurora Serverless v1 existant à l'aide de la commande AWS CLI describe-db-clusters, comme illustré ci-dessous.

Pour Linux, macOS ou Unix :

```
aws rds describe-db-clusters --region region \
  --db-cluster-identifier your-cluster-name \
  --query '*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}'
```

Pour Windows :

```
aws rds describe-db-clusters --region region ^
  --db-cluster-identifier your-cluster-name ^
  --query "*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}"
```

À titre d'exemple, ce qui suit montre la requête et la réponse pour un cluster de bases de données Aurora Serverless v1 nommé `west-coast-sles` dans la région USA Ouest (Californie du Nord).

```
$ aws rds describe-db-clusters --region us-west-1 --db-cluster-identifier west-coast-
sles
```

```
--query '*[].{ScalingConfigurationInfo:ScalingConfigurationInfo}'

[
    {
        "ScalingConfigurationInfo": {
            "MinCapacity": 1,
            "MaxCapacity": 64,
            "AutoPause": false,
            "SecondsBeforeTimeout": 300,
            "SecondsUntilAutoPause": 300,
            "TimeoutAction": "RollbackCapacityChange"
        }
    }
]
```

Comme le montre la réponse, ce cluster de bases de données Aurora Serverless v1 utilise le paramètre par défaut.

Pour plus d'informations, consultez Création d'un cluster de bases de données Aurora Serverless v1. Après avoir créé votre Aurora Serverless v1, vous pouvez modifier l'action d'expiration et d'autres paramètres de capacité à tout moment. Pour savoir comment procéder, consultez Modification d'un cluster de bases de données Aurora Serverless v1.

Mettre en pause et reprendre pour Aurora Serverless v1

Vous pouvez choisir de mettre en pause votre cluster de bases de données Aurora Serverless v1 après un certain temps sans activité. Spécifiez la durée d'inactivité du cluster de bases de données avant que celui-ci soit mis en pause. Lorsque vous sélectionnez cette option, le temps d'inactivité par défaut est de cinq minutes, mais vous pouvez modifier cette valeur. Il s'agit d'une étape facultative.

Lorsque le cluster de bases de données est en pause, aucune activité de calcul ou de mémoire ne se produit ; vous êtes facturé uniquement pour le stockage. Si des connexions de bases de données sont demandées lorsqu'un cluster de bases de données Aurora Serverless v1 est en pause, celui-ci reprend automatiquement et répond aux demandes de connexion.

Lorsque le cluster de bases de données reprend l'activité, il a la même capacité que lorsque Aurora a mis en pause le cluster. Le nombre d'ACU dépend de la quantité mise à l'échelle par Aurora pour le cluster vers le haut ou vers le bas avant de le mettre en pause.

> **Note**
>
> Si un cluster de bases de données est mis en pause pendant plus de sept jours, il peut être sauvegardé avec un instantané. Dans ce cas, Aurora restaure le cluster de bases de données à partir de l'instantané lorsqu'il y a une demande de connexion à celui-ci.

Détermination du nombre maximal de connexions à une base de données pour Aurora Serverless v1

Les exemples suivants s'appliquent à un cluster de bases de données Aurora Serverless v1 compatible avec MySQL 5.7. Vous pouvez utiliser un client MySQL ou l'éditeur de requêtes, si vous y avez configuré l'accès. Pour plus d'informations, consultez Exécution de requêtes dans l'éditeur de requête.

Pour trouver le nombre maximal de connexions à une base de données

1. Trouvez la plage de capacité de votre cluster de bases de données Aurora Serverless v1 à l'aide d'AWS CLI.

   ```
   aws rds describe-db-clusters \
       --db-cluster-identifier my-serverless-57-cluster \
       --query 'DBClusters[*].ScalingConfigurationInfo|[0]'
   ```

   Le résultat indique que sa plage de capacité est comprise entre 1 et 4 ACU.

   ```
   {
       "MinCapacity": 1,
       "AutoPause": true,
       "MaxCapacity": 4,
       "TimeoutAction": "RollbackCapacityChange",
       "SecondsUntilAutoPause": 3600
   }
   ```

2. Exécutez la requête SQL suivante pour trouver le nombre maximal de connexions.

   ```
   select @@max_connections;
   ```

   Le résultat affiché correspond à la capacité minimale du cluster, 1 ACU.

```
@@max_connections
90
```

3. Mettez le cluster à l'échelle jusqu'à 8 à 32 ACU.

   Pour plus d'informations sur le dimensionnement, consultez Modification d'un cluster de bases de données Aurora Serverless v1.

4. Confirmez la plage de capacité.

```
{
    "MinCapacity": 8,
    "AutoPause": true,
    "MaxCapacity": 32,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

5. Trouvez le nombre maximal de connexions.

```
select @@max_connections;
```

   Le résultat affiché correspond à la capacité minimale du cluster, 8 ACU.

```
@@max_connections
1000
```

6. Mettez le cluster à l'échelle jusqu'à la valeur maximale acceptée, 256 ACU.
7. Confirmez la plage de capacité.

```
{
    "MinCapacity": 256,
    "AutoPause": true,
    "MaxCapacity": 256,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

8. Trouvez le nombre maximal de connexions.

```
select @@max_connections;
```

Le résultat affiché correspond à 256 ACU.

```
@@max_connections
6000
```

> **Note**
>
> La valeur `max_connections` n'est pas mise à l'échelle de manière linéaire avec le nombre d'ACU.

9. Remettez le cluster à l'échelle sur 1 à 4 ACU.

```
{
    "MinCapacity": 1,
    "AutoPause": true,
    "MaxCapacity": 4,
    "TimeoutAction": "RollbackCapacityChange",
    "SecondsUntilAutoPause": 3600
}
```

Cette fois, la valeur `max_connections` correspond à 4 ACU.

```
@@max_connections
270
```

10. Réduisez le cluster à 2 ACU.

```
@@max_connections
180
```

Si vous avez configuré le cluster pour qu'il s'arrête après un certain temps d'inactivité, il est réduit à 0 ACU. Cependant, `max_connections` ne tombe pas en dessous de 1 ACU.

```
@@max_connections
90
```

Groupes de paramètres pour Aurora Serverless v1

Lorsque vous créez votre cluster de bases de données Aurora Serverless v1, vous choisissez un moteur de bases de données Aurora spécifique et un groupe de paramètres de cluster de bases de données associé. Contrairement aux clusters de bases de données Aurora provisionnés, un cluster de bases de données Aurora Serverless v1 possède une seule instance de base de données en lecture/écriture configurée avec un groupe de paramètres de cluster de données uniquement. — Il n'a pas de groupe de paramètres de bases de données distinct. Lors de la mise à l'échelle automatique, Aurora Serverless v1 doit être en mesure de modifier les paramètres pour que le cluster fonctionne mieux en fonction de l'augmentation ou de la diminution de la capacité. Ainsi, avec un cluster de bases de données Aurora Serverless v1, certaines des modifications que vous pourriez apporter aux paramètres d'un type de moteur de bases de données particulier peuvent ne pas s'appliquer.

Par exemple, un cluster de bases de données Aurora Serverless v1 basé sur Aurora PostgreSQL– ne peut pas utiliser `apg_plan_mgmt.capture_plan_baselines` et d'autres paramètres qui peuvent être utilisés sur des clusters de bases de données Aurora PostgreSQL provisionnés pour la gestion du plan de requête.

Vous pouvez obtenir une liste de valeurs par défaut pour les groupes de paramètres par défaut pour les différents moteurs de bases de données Aurora en utilisant la commande CLI describe-engine-default-cluster-parameters et en interrogeant la Région AWS. Pour l'option `--db-parameter-group-family`, vous pouvez utiliser les valeurs suivantes :

Aurora MySQL version 2	`aurora-mysql5.7`
Aurora PostgreSQL version 11	`aurora-postgresql11`
Aurora PostgreSQL version 13	`aurora-postgresql13`

Nous vous recommandons de configurer AWS CLI avec votre identifiant de clé d'accès AWS et votre clé d'accès secrète AWS, et de configurer votre Région AWS avant d'utiliser les commandes AWS CLI. La fourniture de la région à votre configuration CLI vous évite d'entrer dans le paramètre `--region` lors de l'exécution des commandes. Pour en savoir plus sur la configuration d'AWS CLI, consultez Principes de base de la configuration dans le Guide de l'utilisateur AWS Command Line Interface.

L'exemple suivant obtient une liste de paramètres du groupe de clusters de bases de données par défaut pour Aurora MySQL version 2.

Pour Linux, macOS ou Unix :

```
aws rds describe-engine-default-cluster-parameters \
  --db-parameter-group-family aurora-mysql5.7 --query \
  'EngineDefaults.Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} | [?
contains(SupportedEngineModes, `serverless`) == `true`] | [*].{param:ParameterName}' \
  --output text
```

Pour Windows :

```
aws rds describe-engine-default-cluster-parameters ^
   --db-parameter-group-family aurora-mysql5.7 --query ^
   "EngineDefaults.Parameters[*].
{ParameterName:ParameterName,SupportedEngineModes:SupportedEngineModes} | [?
contains(SupportedEngineModes, 'serverless') == `true`] | [*].{param:ParameterName}" ^
   --output text
```

Modification des valeurs des paramètres pour l Aurora Serverless v1

Comme expliqué dans Groupes de paramètres pour Amazon Aurora, vous ne pouvez pas modifier directement les valeurs d'un groupe de paramètres par défaut, quel que soit son type (groupe de paramètres de cluster de bases de données, groupe de paramètres de bases de données). Au lieu de cela, vous créez un groupe de paramètres personnalisé basé sur le groupe de paramètres de cluster de bases de données par défaut pour votre moteur de bases de données Aurora et modifiez les paramètres selon les besoins sur ce groupe de paramètres. Par exemple, vous pouvez modifier certains des paramètres de votre cluster de bases de données Aurora Serverless v1 afin qu'il journalise les requêtes ou charge des journaux spécifiques au moteur de base de données sur Amazon CloudWatch.

Pour créer un groupe de paramètres de cluster de bases de données personnalisé

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Groupes de paramètres.
3. Choisissez Créer un groupe de paramètres pour ouvrir le volet des détails du groupe de paramètres.

4. Choisissez le groupe de cluster de bases de données par défaut correspondant au moteur de base de données que vous souhaitez utiliser pour votre cluster de bases de données Aurora Serverless v1. Veillez à choisir les options suivantes :

    a. Pour Famille de groupes de paramètres, choisissez la famille correspondant au moteur de base de données choisi. Assurez-vous que le nom de votre choix comporte le préfixe `aurora-`.

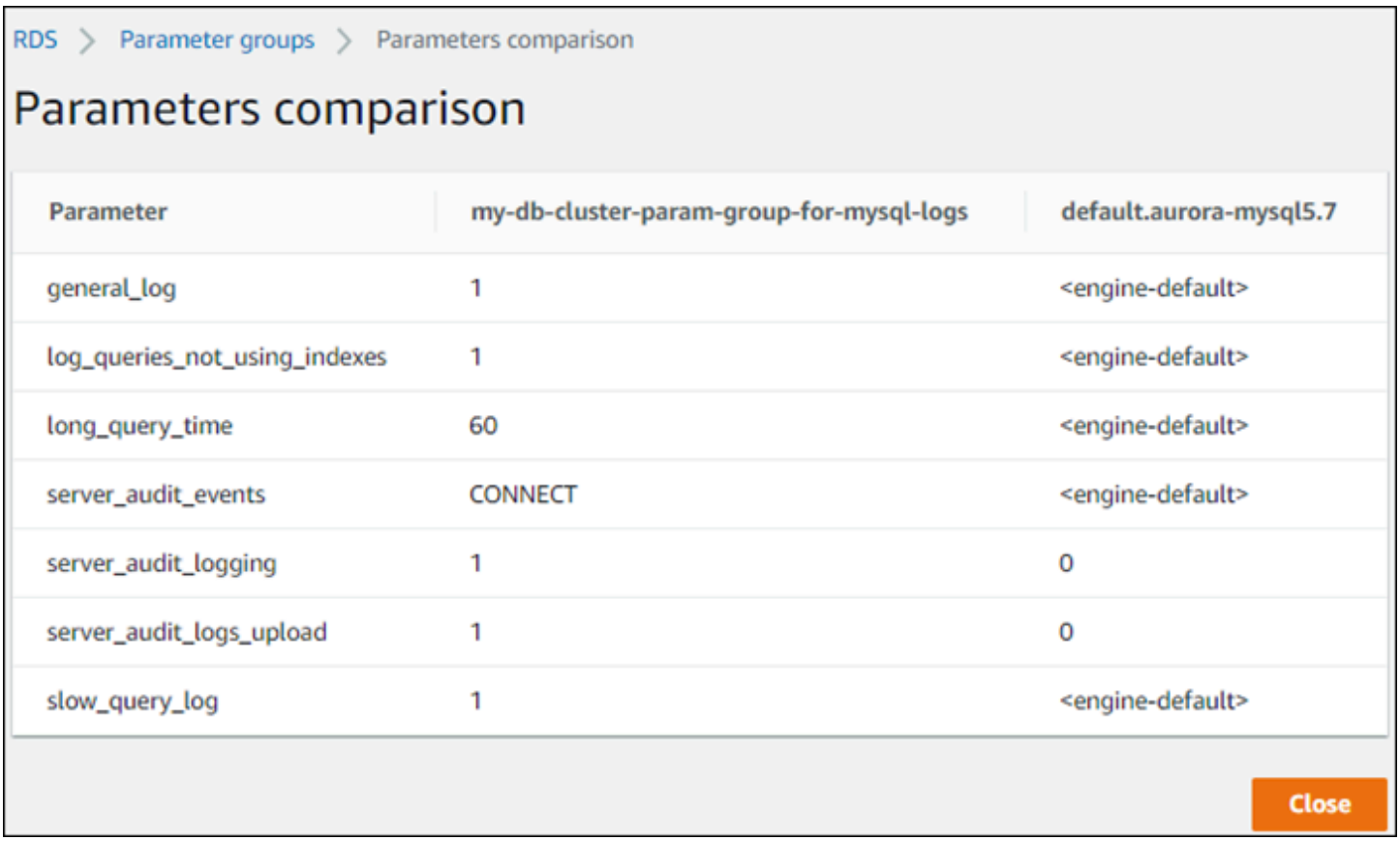
    b. Pour Type, choisissez Groupe de paramètres de cluster de bases de données.

    c. Pour Nom du groupe et Description, entrez des noms descriptifs pour vous ou les autres utilisateurs susceptibles d'utiliser votre cluster de bases de données Aurora Serverless v1 et ses paramètres.

    d. Choisissez Créer.

Votre groupe de paramètres de cluster de bases de données personnalisé est ajouté à la liste des groupes de paramètres disponibles dans votre Région AWS. Vous pouvez utiliser votre groupe de paramètres de cluster de bases de données personnalisé lorsque vous créez des clusters de bases de données Aurora Serverless v1. Vous pouvez également modifier un cluster de bases de données Aurora Serverless v1 existant pour utiliser votre groupe de paramètres de cluster de bases de données personnalisé. Une fois que votre cluster de bases de données Aurora Serverless v1 commence à utiliser votre groupe de paramètres de cluster de bases de données personnalisé, vous pouvez modifier les valeurs des paramètres dynamiques à l'aide d'AWS Management Console ou d'AWS CLI.

Vous pouvez également utiliser la console pour afficher une comparaison côte à côte des valeurs de votre groupe de paramètres de cluster de bases de données personnalisé par rapport au groupe de paramètres de cluster de bases de données par défaut, comme illustré dans la capture d'écran suivante.



Lorsque vous modifiez des valeurs de paramètre sur un cluster de bases de données actif, Aurora Serverless v1 démarre une mise à l'échelle transparente afin d'appliquer les modifications de paramètres. Si votre cluster de bases de données Aurora Serverless v1 est à l'état de pause, il reprend et commence à être mis à l'échelle afin qu'il puisse effectuer la modification. L'opération de mise à l'échelle d'une modification de groupe de paramètres force toujours un changement de capacité, sachez donc que la modification des paramètres peut entraîner l'abandon des connexions si aucun point de mise à l'échelle ne peut être trouvé pendant la période de mise à l'échelle.

Journalisation pour Aurora Serverless v1

Par défaut, les journaux d'erreurs pour Aurora Serverless v1 sont activés et automatiquement téléchargés sur Amazon CloudWatch. Vous pouvez également demander à votre cluster de bases de données Aurora Serverless v1 de charger les journaux spécifiques au moteur de base de données Aurora sur CloudWatch. Pour ce faire, activez les paramètres de configuration dans votre groupe de paramètres de cluster de bases de données personnalisé. Votre cluster de bases de données Aurora Serverless v1 charge ensuite tous les journaux disponibles sur Amazon CloudWatch. À ce stade, vous pouvez utiliser CloudWatch pour analyser les données des journaux, créer des alarmes et afficher des métriques.

Pour Aurora MySQL, le tableau suivant indique les journaux que vous pouvez activer. Lorsqu'ils sont activés, ils sont chargés automatiquement vers Amazon CloudWatch à partir de votre cluster de bases de données Aurora Serverless v1.

Journal Aurora MySQL	Description
`general_log`	Crée le journal général. Paramétrez sur 1 pour activer. La valeur par défaut est désactivée (0).
`log_queries_not_using_indexes`	Journalise les requêtes dans le journal des requêtes lentes qui n'utilisent pas d'index. La valeur par défaut est désactivée (0). Paramétrez sur 1 pour activer ce journal.
`long_query_time`	Empêche l'enregistrement des requêtes rapides dans le journal des requêtes lentes. Peut être défini sur une valeur flottante comprise entre 0 et 31 536 000. La valeur par défaut est 0 (non active).
`server_audit_events`	Liste des événements à capturer dans les journaux. Les valeurs prises en charge sont `CONNECT`, `QUERY`, `QUERY_DCL` , `QUERY_DDL` , `QUERY_DML` , et `TABLE`.
`server_audit_logging`	Paramétrez sur 1 pour activer la journalisation d'audit de serveur. Si vous activez cette option, vous pouvez spécifier les événements d'audit à envoyer à CloudWatch en les répertoriant dans le paramètre `server_audit_events` .
`slow_query_log`	Crée un journal des requêtes lentes. Paramétrez sur 1 pour activer le journal des requêtes lentes. La valeur par défaut est désactivée (0).

Pour plus d'informations, consultez Utilisation de l'Audit avancé avec un cluster de bases de données Amazon Aurora MySQL.

Pour Aurora PostgreSQL, le tableau suivant indique les journaux que vous pouvez activer. Lorsqu'ils sont activés, ils sont chargés automatiquement vers Amazon CloudWatch à partir de votre cluster de bases de données Aurora Serverless v1, aux côtés des journaux d'erreurs habituels.

Journal Aurora PostgreSQL	Description
`log_connections`	Activé par défaut et ne peut pas être modifié. Il journalise les détails pour toutes les nouvelles connexions client.
`log_disconnections`	Activé par défaut et ne peut pas être modifié. Journalise toutes les déconnexions du client.
`log_hostname`	Désactivé par défaut et ne peut pas être modifié. Les noms d'hôte ne sont pas journalis és.
`log_lock_waits`	La valeur par défaut est 0 (désactivée). Paramétrez sur 1 pour journaliser les attentes de verrouillage.
`log_min_duration_statement`	Durée minimale (en millisecondes) d'exécution d'une instruction avant qu'elle ne soit journalis ée.
`log_min_messages`	Définit les niveaux des messages qui sont enregistrés. Les valeurs prises en charge sont `debug5`, `debug4`, `debug3`, `debug2`, `debug1`, `info`, `notice`, `warning`, `error`, `log`, `fatal`, `panic`. Pour consigner les données de performances dans le journal `postgres` , définissez la valeur sur `debug1`.
`log_temp_files`	Journalise l'utilisation de fichiers temporair es qui sont au-dessus des kilo-octets (Ko) spécifiés.

Journal Aurora PostgreSQL	Description
`log_statement`	Contrôle les instructions SQL spécifiques qui sont journalisées. Les valeurs prises en charge sont `none`, `ddl`, `mod` et `all`. La valeur par défaut est `none`.

Une fois que vous avez activé les journaux pour Aurora MySQL ou Aurora PostgreSQL pour votre cluster de bases de données Aurora Serverless v1, vous pouvez afficher les journaux dans CloudWatch.

Affichage de journaux Aurora Serverless v1 avec Amazon CloudWatch

Aurora Serverless v1 charge automatiquement (« publie ») sur Amazon CloudWatch tous les journaux activés dans votre groupe de paramètres de cluster de bases de données personnalisé. Vous n'avez pas besoin de choisir ou de spécifier les types de journaux. Le chargement des journaux démarre dès que vous activez le paramètre de configuration du journal. Si vous désactivez ultérieurement le paramètre de journal, les chargements supplémentaires s'arrêtent. Cependant, tous les journaux qui ont déjà été publiés dans CloudWatch restent jusqu'à ce que vous les supprimiez.

Pour plus d'informations sur l'utilisation de CloudWatch avec les journaux Aurora MySQL, consultez Surveillance des événements de journaux dans Amazon CloudWatch.

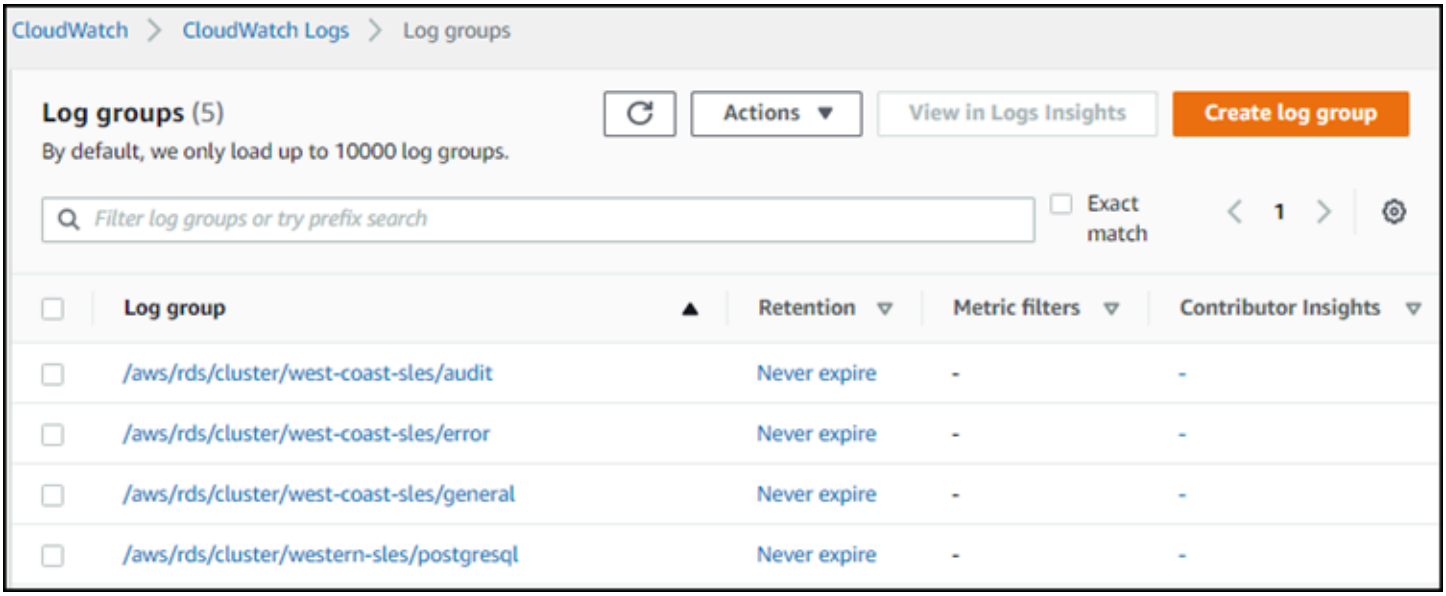
Pour plus d'informations sur CloudWatch et Aurora PostgreSQL, consultez Publication de journaux Aurora PostgreSQL sur Amazon CloudWatch Logs.

Pour afficher les journaux de votre cluster de bases de données Aurora Serverless v1

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Choisissez votre Région AWS.
3. Choisissez Groupes de journaux.
4. Choisissez votre journal de cluster de bases de données Aurora Serverless v1 dans la liste. Pour les journaux d'erreurs, le modèle d'affectation de noms est le suivant.

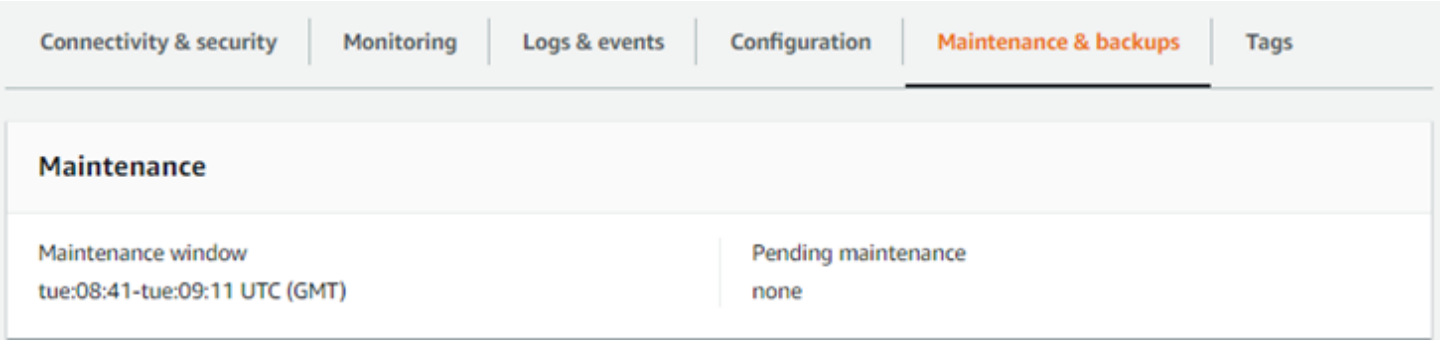
```
/aws/rds/cluster/cluster-name/error
```

Par exemple, dans la capture d'écran suivante, vous pouvez trouver des listes pour les journaux publiés pour un cluster de bases de données Aurora PostgreSQL Aurora Serverless v1 nommé `western-sles`. Vous y trouverez également plusieurs listes pour le cluster de bases de données Aurora MySQL Aurora Serverless v1, `west-coast-sles`. Choisissez le journal qui vous intéresse pour commencer à explorer son contenu.



Aurora Serverless v1 et maintenance

La maintenance du cluster de bases de données Aurora Serverless v1, comme l'application des dernières fonctionnalités, des correctifs et des mises à jour de sécurité, est effectuée automatiquement pour vous. Aurora Serverless v1 dispose d'une fenêtre de maintenance que vous pouvez visualiser dans la AWS Management Console, dans Maintenance et sauvegardes, pour votre cluster de bases de données Aurora Serverless v1. Vous pouvez trouver la date et l'heure auxquelles la maintenance peut être effectuée et voir si une maintenance est en attente pour votre cluster de bases de données Aurora Serverless v1, comme illustré dans la figure ci-dessous.



Vous pouvez définir la fenêtre de maintenance lorsque vous créez le cluster de bases de données Aurora Serverless v1, et vous pouvez modifier cette fenêtre ultérieurement. Pour plus d'informations, consultez Ajustement du créneau de maintenance préféré pour un cluster de bases de données.

Les fenêtres de maintenance sont utilisées pour les mises à niveau des versions majeures planifiées. Les mises à niveau et les correctifs des versions mineures sont appliqués immédiatement lors de la mise à l'échelle. La mise à l'échelle s'effectue en fonction de vos paramètres pour `TimeoutAction` :

- `ForceApplyCapacityChange` : la modification s'applique immédiatement.
- `RollbackCapacityChange` : Aurora met à jour de force le cluster 3 jours après la première tentative de correctif.

Comme pour toute modification forcée sans point de mise à l'échelle approprié, votre charge de travail peut être interrompue.

Dans la mesure du possible, Aurora Serverless v1 effectue la maintenance sans aucune perturbation. Lorsqu'une maintenance est requise, votre cluster de bases de données Aurora Serverless v1 met à l'échelle sa capacité pour gérer les opérations nécessaires. Avant la mise à l'échelle, Aurora Serverless v1 recherche un point de mise à l'échelle. Il le fait pendant trois jours si nécessaire.

Si, au terme de chaque journée, Aurora Serverless v1 ne trouve pas de point de mise à l'échelle, il crée un événement de cluster. Cet événement vous informe de la maintenance en attente et de la nécessité de procéder à une mise à l'échelle pour l'effectuer. Cette notification inclut la date à laquelle Aurora Serverless v1 peut forcer le cluster de bases de données à procéder à une mise à l'échelle.

Pour plus d'informations, consultez Action de délai d'attente pour les modifications de capacité.

Aurora Serverless v1 et basculement

Si l'instance de base de données pour un cluster de bases de données Aurora Serverless v1 devient indisponible ou si la zone de disponibilité dans laquelle elle se trouve échoue, Aurora recrée l'instance de base de données dans une autre zone de disponibilité. Cependant, le cluster Aurora Serverless v1 n'est pas un cluster multi-AZ. En effet, il comporte une instance de base de données unique dans une seule zone de disponibilité. Le mécanisme de basculement prend donc plus de temps pour un cluster Aurora comportant des instances Aurora Serverless v2 ou provisionnées. Le délai de basculement d'Aurora Serverless v1 est indéfini, car il dépend de la demande et de la capacité disponible dans les autres zones de disponibilité de la Région AWS.

Étant donné qu'Aurora sépare la capacité de calcul et le stockage, le volume de stockage pour le cluster est réparti entre plusieurs zones de disponibilité. Vos données restent disponibles même si des pannes affectent l'instance de base de données ou la zone de disponibilité associée.

Aurora Serverless v1 et instantanés

Le volume de cluster d'un cluster Aurora Serverless v1 est toujours chiffré. Vous pouvez choisir la clé de chiffrement, mais vous ne pouvez pas désactiver le chiffrement. Pour copier ou partager un instantané d'un cluster Aurora Serverless v1, chiffrez cet instantané à l'aide de votre propre AWS KMS key. Pour plus d'informations, consultez Copie d'un instantané de cluster de bases de données. Pour en savoir plus sur le chiffrement et sur Amazon Aurora, consultez Chiffrement d'un cluster de bases de données Amazon Aurora

Création d'un cluster de bases de données Aurora Serverless v1

> **Important**
>
> AWS a annoncé la date de fin de vie d'Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

La procédure suivante crée un cluster Aurora Serverless v1 sans objet ni données de votre schéma. Pour créer un cluster Aurora Serverless v1 qui est un doublon d'un cluster approvisionné ou Aurora Serverless v1 existant, vous pouvez effectuer une opération de restauration d'instantané ou de clonage à la place. Pour plus d'informations, consultez Restauration à partir d'un instantané de cluster de bases de données et Clonage d'un volume pour un cluster de bases de données Amazon Aurora. Vous ne pouvez pas convertir un cluster approvisionné en Aurora Serverless v1. Vous ne pouvez également pas reconvertir un cluster Aurora Serverless v1 existant en cluster approvisionné.

Lorsque vous créez un cluster de bases de données Aurora Serverless v1, vous pouvez définir la capacité minimale et maximale du cluster. Une unité de capacité correspond à une configuration de calcul et de mémoire spécifique. Aurora Serverless v1 crée des règles de mise à l'échelle pour les

seuils d'utilisation du processeur, de connexions et de mémoire disponible, puis est mis à l'échelle de manière transparente pour atteindre la plage d'unités de capacité nécessaire à vos applications. Pour plus d'informations, consultez Architecture d'Aurora Serverless v1.

Vous pouvez définir les valeurs spécifiques suivantes pour votre cluster de bases de données Aurora Serverless v1 :

- Unité de capacité Aurora minimale – Aurora Serverless v1 peut réduire la capacité jusqu'à cette unité de capacité.
- Unité de capacité Aurora maximale – Aurora Serverless v1 peut augmenter la capacité jusqu'à cette unité de capacité.

Vous pouvez également choisir les options facultatives de configuration de mise à l'échelle suivantes :

- Forcer la mise à l'échelle de la capacité aux valeurs spécifiées lorsque le délai d'expiration est atteint – Vous pouvez choisir ce paramètre si vous souhaitez qu'Aurora Serverless v1 force la mise à l'échelle d'Aurora Serverless v1 même s'il ne trouve pas de point de mise à l'échelle avant d'expirer. Pour permettre à Aurora Serverless v1 d'annuler les modifications de capacité en l'absence de point de mise à l'échelle, ne choisissez pas ce paramètre. Pour plus d'informations, consultez Action de délai d'attente pour les modifications de capacité.
- Suspendre la capacité de calcul après des minutes consécutives d'inactivité – Vous pouvez choisir ce paramètre si vous souhaitez qu'Aurora Serverless v1 effectue une réduction verticale à zéro quand il n'y a pas d'activité sur votre cluster de bases de données pendant une période spécifiée. Lorsque ce paramètre est activé, votre cluster de bases de données Aurora Serverless v1 reprend automatiquement le traitement et procède à une mise à l'échelle vers la capacité nécessaire pour gérer la charge de travail lorsque le trafic de base de données reprend. Pour en savoir plus, consultez Mettre en pause et reprendre pour Aurora Serverless v1.

Pour créer un cluster de bases de données Aurora Serverless v1, vous devez disposer d'un compte AWS. Vous devez également avoir terminé les tâches de configuration pour travailler avec Amazon Aurora. Pour plus d'informations, consultez Configuration de votre environnement pour Amazon Aurora. Vous devez également effectuer d'autres étapes préliminaires pour créer un cluster de bases de données Aurora. Pour en savoir plus, consultez Création d'un cluster de bases de données Amazon Aurora.

Aurora Serverless v1 est disponible dans certaines Régions AWS et uniquement pour des versions spécifiques d'Aurora MySQL et d'Aurora PostgreSQL. Pour plus d'informations, consultez Aurora Serverless v1.

> **Note**
>
> Le volume de cluster d'un cluster Aurora Serverless v1 est toujours chiffré. Lorsque vous créez votre cluster de bases de données Aurora Serverless v1, il ne vous est pas possible de désactiver le chiffrement, mais vous pouvez choisir d'utiliser votre propre clé de chiffrement. Avec Aurora Serverless v2, vous pouvez choisir de chiffrer le volume du cluster.

Vous pouvez créer un cluster de bases de données Aurora Serverless v1 à l'aide de l'AWS CLI ou de l'API RDS.

> **Note**
>
> Si vous recevez le message d'erreur suivant lorsque vous essayez de créer votre cluster, votre compte a besoin d'autorisations supplémentaires.
> `Unable to create the resource. Verify that you have permission to create service linked role. Otherwise wait and try again later.`
> Pour plus d'informations, consultez Utilisation des rôles liés à un service pour Amazon Aurora.

Vous ne pouvez pas vous connecter directement à l'instance de base de données sur votre cluster de bases de données Aurora Serverless v1. Pour vous connecter à votre cluster de bases de données Aurora Serverless v1, vous utilisez le point de terminaison de base de données. Le point de terminaison de votre cluster de bases de données Aurora Serverless v1 est disponible sous l'onglet Connectivité et sécurité de votre cluster dans AWS Management Console. Pour plus d'informations, consultez Connexion à un cluster de bases de données Amazon Aurora.

AWS CLI

Pour créer un cluster de bases de données Aurora Serverless v1 avec l'AWS CLI, exécutez la commande create-db-cluster et spécifiez `serverless` pour l'option `--engine-mode`.

Vous pouvez, si vous le souhaitez, spécifier l'option `--scaling-configuration` pour configurer la capacité minimale, la capacité maximale et la mise en pause automatique s'il n'y a aucune connexion.

La commande suivante crée un cluster de bases de données sans serveur en définissant l'option `--engine-mode` sur `serverless`. L'exemple spécifie également des valeurs pour l'option `--scaling-configuration`.

Exemple pour Aurora MySQL

La commande suivante crée un nouveau cluster de bases de données sans serveur compatible avec Aurora MySQL. Les valeurs de capacité valides pour Aurora MySQL sont 1, 2, 4, 8, 16, 32, 64, 128 et 256.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster \
    --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.4 \
    --engine-mode serverless \
    --scaling-configuration
 MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true \
    --master-username username --master-user-password password
```

Pour Windows :

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
    --engine aurora-mysql --engine-version 5.7.mysql_aurora.2.11.4 ^
    --engine-mode serverless ^
    --scaling-configuration
 MinCapacity=4,MaxCapacity=32,SecondsUntilAutoPause=1000,AutoPause=true ^
    --master-username username --master-user-password password
```

Exemple pour Aurora PostgreSQL

La commande suivante crée un nouveau cluster de bases de données sans serveur compatible avec PostgreSQL 13.9. Les valeurs de capacité valides pour Aurora PostgreSQL sont 2, 4, 8, 16, 32, 64, 192 et 384.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster \
    --engine aurora-postgresql --engine-version 13.9 \
```

```
    --engine-mode serverless \
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=1000,AutoPause=true \
    --master-username username --master-user-password password
```

Pour Windows :

```
aws rds create-db-cluster --db-cluster-identifier sample-cluster ^
    --engine aurora-postgresql --engine-version 13.9 ^
    --engine-mode serverless ^
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=1000,AutoPause=true ^
    --master-username username --master-user-password password
```

API RDS

Pour créer un cluster de bases de données Aurora Serverless v1 à partir de l'API RDS, exécutez l'opération CreateDBCluster et spécifiez `serverless` pour le paramètre `EngineMode`.

Vous pouvez, si vous le souhaitez, spécifier le paramètre `ScalingConfiguration` pour configurer la capacité minimale, la capacité maximale et la mise en pause automatique s'il n'y a aucune connexion. Les valeurs de capacité valides sont notamment les suivantes :

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 et 256.
- Aurora PostgreSQL : 2, 4, 8, 16, 32, 64, 192 et 384.

Restauration d'un cluster de bases de données Aurora Serverless v1

> **Important**
>
> AWS a annoncé la date de fin de vie d'Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

Vous pouvez configurer un cluster de bases de données Aurora Serverless v1 lorsque vous restaurez un instantané de cluster de bases de données provisionné avec l'AWS CLI ou l'API RDS.

Lorsque vous restaurez un instantané dans un cluster de bases de données Aurora Serverless v1, vous pouvez définir les valeurs spécifiques suivantes :

- Unité de capacité Aurora minimale – Aurora Serverless v1 peut réduire la capacité jusqu'à cette unité de capacité.
- Unité de capacité Aurora maximale – Aurora Serverless v1 peut augmenter la capacité jusqu'à cette unité de capacité.
- Action d'expiration – Action à effectuer quand une modification de capacité expire parce qu'elle ne trouve pas de point de mise à l'échelle.Aurora Serverless v1 Le cluster de bases de données peut forcer l'application des nouveaux paramètres de capacité à votre cluster de bases de données si vous définissez l'option Forcer la mise à l'échelle de la capacité aux valeurs spécifiées.... Si vous n'activez pas cette option, il peut également annuler la modification de capacité. Pour plus d'informations, consultez Action de délai d'attente pour les modifications de capacité.
- Pause after inactivity (Mise en pause après inactivité) – Durée sans trafic de base de données avant mise à l'échelle à une capacité de traitement égale à zéro. Lors de la reprise du trafic de base de données, Aurora reprend automatiquement la capacité de traitement et effectue une mise à l'échelle pour gérer le trafic.

Pour obtenir des informations générales sur la restauration d'un cluster de bases de données à partir d'un instantané, consultez Restauration à partir d'un instantané de cluster de bases de données.

AWS CLI

Vous pouvez configurer un cluster de bases de données Aurora Serverless lorsque vous restaurez un instantané de cluster de bases de données provisionné avec l'AWS Management Console, l'AWS CLI ou l'API RDS.

Lorsque vous restaurez un instantané dans un cluster de bases de données Aurora Serverless, vous pouvez définir les valeurs spécifiques suivantes :

- Unité de capacité Aurora minimale – Aurora Serverless peut réduire la capacité jusqu'à cette unité de capacité.
- Unité de capacité Aurora maximale – Aurora Serverless peut augmenter la capacité jusqu'à cette unité de capacité.

- Action d'expiration – Action à effectuer quand une modification de capacité expire parce qu'elle ne trouve pas de point de mise à l'échelle.Aurora Serverless v1 Le cluster de bases de données peut forcer l'application des nouveaux paramètres de capacité à votre cluster de bases de données si vous définissez l'option Forcer la mise à l'échelle de la capacité aux valeurs spécifiées.... Si vous n'activez pas cette option, il peut également annuler la modification de capacité. Pour plus d'informations, consultez Action de délai d'attente pour les modifications de capacité.
- Pause after inactivity (Mise en pause après inactivité) – Durée sans trafic de base de données avant mise à l'échelle à une capacité de traitement égale à zéro. Lors de la reprise du trafic de base de données, Aurora reprend automatiquement la capacité de traitement et effectue une mise à l'échelle pour gérer le trafic.

> **Note**
>
> La version de l'instantané du cluster de bases de données doit être compatible avec Aurora Serverless v1. Pour obtenir la liste des versions prises en charge, consultez Aurora Serverless v1.

Pour restaurer un instantané dans un cluster Aurora Serverless v1 avec compatibilité MySQL 5.7, incluez les paramètres supplémentaires suivants :

- `--engine` **`aurora-mysql`**
- `--engine-version` **`5.7`**

Les paramètres `--engine` et `--engine-version` vous permettent de créer un cluster Aurora Serverless v1 compatible MySQL 5.7 à partir d'un instantané Aurora compatible MySQL 5.6 ou Aurora Serverless v1. L'exemple suivant restaure un instantané de cluster compatible MySQL 5.6 nommé *`mydbclustersnapshot`* vers un cluster Aurora Serverless v1 compatible MySQL 5.7 nommé *`mynewdbcluster`*.

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine-mode serverless \
    --engine aurora-mysql \
```

```
    --engine-version 5.7
```

Pour Windows :

```
aws rds restore-db-cluster-from-snapshot ^
    --db-instance-identifier mynewdbcluster ^
    --db-snapshot-identifier mydbclustersnapshot ^
    --engine aurora-mysql ^
    --engine-version 5.7
```

Vous pouvez, si vous le souhaitez, spécifier l'option `--scaling-configuration` pour configurer la capacité minimale, la capacité maximale et la mise en pause automatique s'il n'y a aucune connexion. Les valeurs de capacité valides sont notamment les suivantes :

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 et 256.
- Aurora PostgreSQL : 2, 4, 8, 16, 32, 64, 192 et 384.

Dans l'exemple suivant, vous restaurez à partir d'un instantané de cluster de bases de données créé précédemment nommé *mydbclustersnapshot* vers un nouveau cluster de bases de données nommé *mynewdbcluster*. Vous définissez le `--scaling-configuration` afin que le nouveau cluster de bases de données Aurora Serverless v1 puisse évoluer de 8 ACU à 64 ACU (unités de capacité Aurora) selon les besoins pour traiter la charge de travail. Une fois le traitement terminé et après 1 000 secondes sans connexion à prendre en charge, le cluster s'arrête jusqu'à ce que les demandes de connexion l'invitent à redémarrer.

Pour Linux, macOS ou Unix :

```
aws rds restore-db-cluster-from-snapshot \
    --db-cluster-identifier mynewdbcluster \
    --snapshot-identifier mydbclustersnapshot \
    --engine-mode serverless --scaling-configuration
 MinCapacity=8,MaxCapacity=64,TimeoutAction='ForceApplyCapacityChange',SecondsUntilAutoPause=10
```

Pour Windows :

```
aws rds restore-db-cluster-from-snapshot ^
    --db-instance-identifier mynewdbcluster ^
    --db-snapshot-identifier mydbclustersnapshot ^
```

```
    --engine-mode serverless --scaling-configuration
MinCapacity=8,MaxCapacity=64,TimeoutAction='ForceApplyCapacityChange',SecondsUntilAutoPause=10
```

API RDS

Pour configurer un cluster de bases de données Aurora Serverless v1 lorsque vous procédez à une restauration à partir d'un cluster de bases de données avec l'API RDS, exécutez l'opération RestoreDBClusterFromSnapshot et spécifiez `serverless` pour le paramètre `EngineMode`.

Vous pouvez, si vous le souhaitez, spécifier le paramètre `ScalingConfiguration` pour configurer la capacité minimale, la capacité maximale et la mise en pause automatique s'il n'y a aucune connexion. Les valeurs de capacité valides sont notamment les suivantes :

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 et 256.
- Aurora PostgreSQL : 2, 4, 8, 16, 32, 64, 192 et 384.

Modification d'un cluster de bases de données Aurora Serverless v1

> **Important**
>
> AWS a annoncé la date de fin de vie d'Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

Après avoir configuré un cluster de bases de données Aurora Serverless v1, vous pouvez modifier certaines propriétés avec AWS Management Console, AWS CLI ou l'API RDS. La plupart des propriétés que vous pouvez modifier sont identiques à celles des autres types de clusters Aurora.

Voici les changements les plus pertinents pour Aurora Serverless v1 :

- Modification de la configuration de mise à l'échelle

- Mise à niveau de la version majeure
- Conversion d'Aurora Serverless v1 en mode approvisionné

Modification de la configuration de mise à l'échelle d'un cluster de bases de données Aurora Serverless v1

Vous pouvez définir la capacité minimale et maximale pour le cluster de bases de données. Chaque unité de capacité correspond à une configuration de calcul et de mémoire spécifique. Aurora Serverless crée automatiquement des règles de mise à l'échelle pour les seuils d'utilisation de l'UC, de connexions et de mémoire disponible. Vous pouvez également définir si Aurora Serverless met en pause la base de données en l'absence d'activité, puis la réactive lorsque l'activité reprend.

Vous pouvez définir les valeurs spécifiques suivantes pour la configuration de la mise à l'échelle :

- Unité de capacité Aurora minimale – Aurora Serverless peut réduire la capacité jusqu'à cette unité de capacité.
- Unité de capacité Aurora maximale – Aurora Serverless peut augmenter la capacité jusqu'à cette unité de capacité.
- Délai et action de scalabilité automatique – Cette section spécifie combien de temps Aurora Serverless attend de trouver un point de mise à l'échelle avant d'expirer. Il spécifie également l'action à effectuer lorsqu'une modification de capacité expire, car elle ne trouve pas de point de mise à l'échelle. Aurora peut forcer la modification de capacité à définir la capacité sur la valeur spécifiée dès que possible. Il peut également annuler la modification de capacité. Pour plus d'informations, consultez Action de délai d'attente pour les modifications de capacité.
- Pause après inactivité : utilisez le paramètre facultatif Faire passer la capacité à 0 ACU lorsque le cluster est inactif pour mettre à l'échelle la base de données avec une capacité de traitement nulle lorsqu'elle est inactive. Lors de la reprise du trafic de base de données, Aurora reprend automatiquement la capacité de traitement et effectue une mise à l'échelle pour gérer le trafic.

> **Note**
>
> Lorsque vous modifiez la plage de capacité d'un cluster de bases de données Aurora Serverless, la modification intervient immédiatement, que vous choisissiez de l'appliquer immédiatement ou lors du prochain créneau de maintenance planifié.

Console

Vous pouvez modifier la configuration de mise à l'échelle d'un cluster de bases de données Aurora avec AWS Management Console.

Pour modifier un cluster de bases de données Aurora Serverless v1

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster de bases de données Aurora Serverless v1 à modifier.
4. Pour Actions, choisissez Modifier le cluster.
5. Dans la section Capacity settings (Paramètres de capacité), modifiez la configuration de mise à l'échelle.
6. Choisissez Continuer.
7. Sur la page Modifier le cluster de bases de données, vérifiez vos modifications, puis choisissez quand les appliquer.
8. Choisissez Modifier le cluster.

AWS CLI

Pour modifier la configuration de mise à l'échelle d'un cluster de bases de données Aurora Serverless v1 à partir de l'AWS CLI, exécutez la commande modify-db-cluster de l'AWS CLI. Spécifiez l'option `--scaling-configuration` pour configurer la capacité minimale, la capacité maximale et la mise en pause automatique s'il n'y a aucune connexion. Les valeurs de capacité valides sont notamment les suivantes :

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 et 256.
- Aurora PostgreSQL : 2, 4, 8, 16, 32, 64, 192 et 384.

Dans cet exemple, vous modifiez la configuration de mise à l'échelle d'un cluster de bases de données Aurora Serverless v1 nommé *`sample-cluster`*.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
```

```
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=500,TimeoutAction='ForceApplyCapacityChange
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --scaling-configuration
 MinCapacity=8,MaxCapacity=64,SecondsUntilAutoPause=500,TimeoutAction='ForceApplyCapacityChange
```

API RDS

Vous pouvez modifier la configuration de mise à l'échelle d'un cluster de bases de données Aurora avec l'opération d'API ModifyDBCluster. Spécifiez le paramètre `ScalingConfiguration` pour configurer la capacité minimale, la capacité maximale et la mise en pause automatique s'il n'y a aucune connexion. Les valeurs de capacité valides sont notamment les suivantes :

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 et 256.
- Aurora PostgreSQL : 2, 4, 8, 16, 32, 64, 192 et 384.

Mise à niveau de la version majeure d'un cluster de bases de données Aurora Serverless v1

> **Important**
>
> AWS a annoncé la date de fin de vie d'Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

Vous pouvez mettre à niveau la version majeure d'un cluster de bases de données Aurora Serverless v1 compatible avec PostgreSQL 11 vers une version compatible avec PostgreSQL 13 correspondante.

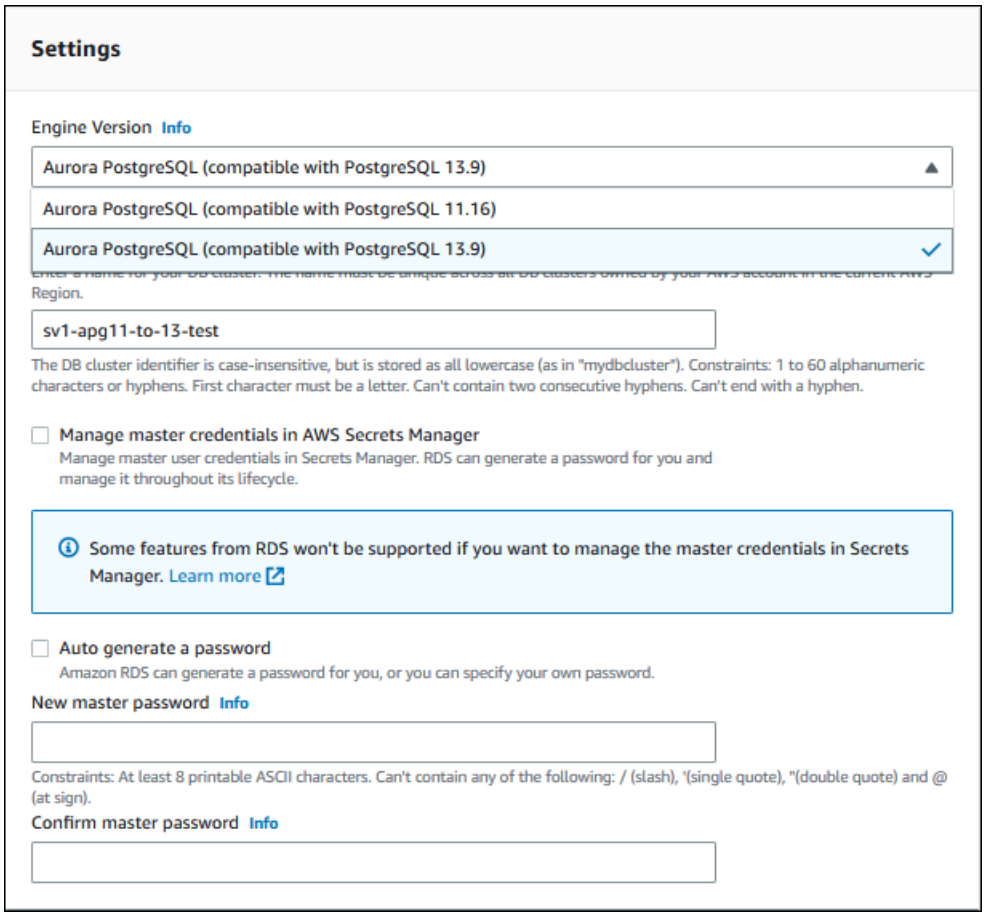
Console

Vous pouvez effectuer une mise à niveau sur place d'un cluster de bases de données Aurora Serverless v1 avec la AWS Management Console.

Pour mettre à niveau un cluster de bases de données Aurora Serverless v1

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Choisissez le cluster de bases de données Aurora Serverless v1 que vous voulez mettre à niveau.
4. Pour Actions, choisissez Modifier le cluster.
5. Pour Version, choisissez un numéro de version d'Aurora PostgreSQL version 13.

   L'exemple suivant montre une mise à niveau sur place d'Aurora PostgreSQL 11.16 vers la version 13.9.



Si vous effectuez une mise à niveau de version majeure, ne modifiez aucune autre propriété. Pour modifier d'autres propriétés, effectuez une autre opération Modifier une fois la mise à niveau terminée.

6. Choisissez Continuer.
7. Sur la page Modifier le cluster de bases de données, vérifiez vos modifications, puis choisissez quand les appliquer.
8. Choisissez Modifier le cluster.

AWS CLI

Pour effectuer une mise à niveau sur place d'un cluster de bases de données Aurora Serverless v1 compatible avec PostgreSQL 11 vers un cluster compatible avec PostgreSQL 13, spécifiez le paramètre `--engine-version` avec un numéro de version 13 d'Aurora PostgreSQL compatible avec Aurora Serverless v1. Incluez également le paramètre `--allow-major-version-upgrade`.

Dans cet exemple, vous modifiez la version majeure d'un cluster de bases de données Aurora Serverless v1 compatible avec PostgreSQL 11 nommé `sample-cluster`. Ce faisant, vous effectuez une mise à niveau sur place vers un cluster de bases de données Aurora Serverless v1 compatible avec PostgreSQL 13.

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --engine-version 13.serverless_12 \
    --allow-major-version-upgrade
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --engine-version 13.serverless_12 ^
    --allow-major-version-upgrade
```

API RDS

Pour effectuer une mise à niveau sur place d'un cluster de bases de données Aurora Serverless v1 compatible avec PostgreSQL 11 vers un cluster compatible avec PostgreSQL 13, spécifiez le paramètre `EngineVersion` avec un numéro de version 13 d'Aurora PostgreSQL compatible avec Aurora Serverless v1. Incluez également le paramètre `AllowMajorVersionUpgrade`.

Conversion d'un cluster de bases de données Aurora Serverless v1 provisionné

Vous pouvez convertir un cluster de bases de données Aurora Serverless v1 en un cluster de bases de données provisionné. Pour effectuer la conversion, utilisez l'interface AWS CLI ou l'API Amazon RDS afin de remplacer la classe d'instance de base de données par Provisionné. Suivez les étapes ci-dessous pour modifier votre classe d'instance de base de données.

L'exemple suivant montre comment utiliser l'interface AWS CLI pour convertir un cluster de bases de données Aurora Serverless v1 en cluster provisionné.

AWS CLI

Pour convertir un cluster de bases de données Aurora Serverless v1 en cluster provisionné, exécutez la commande AWS CLI modify-db-cluster.

Les paramètres suivants sont obligatoires :

- `--db-cluster-identifier` : le cluster de bases de données Aurora Serverless v1 que vous convertissez en cluster provisionné.
- `--engine-mode` : utilisez la valeur `provisioned`.
- `--allow-engine-mode-change`
- `--db-cluster-instance-class` : choisissez la classe d'instance de base de données pour le cluster de bases de données provisionné en fonction de la capacité du cluster de bases de données Aurora Serverless v1.

Dans cet exemple, vous convertissez un cluster de bases de données Aurora Serverless v1 nommé `sample-cluster` et utilisez la classe d'instance de base de données `db.r5.xlarge`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
--db-cluster-identifier sample-cluster \
--engine-mode provisioned \
--allow-engine-mode-change \
--db-cluster-instance-class db.r5.xlarge
```

Pour Windows :

```
aws rds modify-db-cluster ^
--db-cluster-identifier sample-cluster ^
--engine-mode provisioned ^
--allow-engine-mode-change ^
--db-cluster-instance-class db.r5.xlarge
```

L'exemple suivant montre comment utiliser l'API Amazon RDS pour convertir un cluster de bases de données Aurora Serverless v1 en cluster provisionné.

API RDS

Pour convertir un cluster de bases de données Aurora Serverless v1 en cluster provisionné, exécutez l'opération d'API ModifyDBCluster.

Les paramètres suivants sont obligatoires :

- `DBClusterIdentifier` : le cluster de bases de données Aurora Serverless v1 que vous convertissez en cluster provisionné.
- `EngineMode` : utilisez la valeur `provisioned`.
- `AllowEngineModeChange`
- `DBClusterInstanceClass` : choisissez la classe d'instance de base de données pour le cluster de bases de données provisionné en fonction de la capacité du cluster de bases de données Aurora Serverless v1.

Considérations relatives à la conversion d'un cluster de bases de données Aurora Serverless v1 en cluster provisionné

Les considérations suivantes s'appliquent lorsqu'un cluster de bases de données Aurora Serverless v1 est converti en cluster provisionné :

- Vous pouvez utiliser cette conversion dans le cadre de la mise à niveau de votre cluster de bases de données d'Aurora Serverless v1 à Aurora Serverless v2. Pour plus d'informations, consultez Mise à niveau d'un cluster Aurora Serverless v1 vers Aurora Serverless v2.
- Le processus de conversion crée une instance de base de données de lecteur dans le cluster de bases de données, transforme l'instance de lecteur en instance d'enregistreur, puis supprime l'instance Aurora Serverless v1 d'origine. Lorsque vous convertissez le cluster de bases de données, vous ne pouvez effectuer aucune autre modification en même temps, telle que la

modification de la version du moteur de base de données ou du groupe de paramètres du cluster de bases de données. L'opération de conversion est appliquée immédiatement et ne peut être annulée.

- Au cours de la conversion, un instantané de sauvegarde du cluster de bases de données est pris au cas où une erreur se produirait. L'identifiant de l'instantané de cluster de bases de données a le format `pre-modify-engine-mode-`*`DB_cluster_identifier`*`-`*`timestamp`*.
- Aurora utilise la version mineure actuelle du moteur de base de données par défaut pour le cluster de bases de données provisionné.
- Si vous ne fournissez pas de classe d'instance de base de données pour votre cluster de bases de données converti, Aurora en recommande une en fonction de la capacité maximale du cluster de bases de données Aurora Serverless v1. Les mappages de capacité et de classes d'instance recommandés sont indiqués dans la table suivante.

Capacité maximale de Serverless (ACU)	Classe d'instance de base de données provisionnée
1	db.t3.small
2	db.t3.medium
4	db.t3.large
8	db.r5.large
16	db.r5.xlarge
32	db.r5.2xlarge
64	db.r5.4xlarge
128	db.r5.8xlarge
192	db.r5.12xlarge
256	db.r5.16xlarge
384	db.r5.24xlarge

> **Note**
>
> En fonction de la classe d'instance de base de données que vous choisissez et de l'utilisation de votre base de données, vous pouvez constater des coûts différents pour un cluster de bases de données provisionné par rapport à Aurora Serverless v1.
> Si vous convertissez votre cluster de bases de données Aurora Serverless v1 en une classe d'instance de base de données évolutive (db.t*), vous risquez d'encourir des coûts supplémentaires liés à l'utilisation du cluster de bases de données. Pour plus d'informations, consultez Types de classes d'instance de base de données.

Mise à l'échelle manuelle de la capacité d'un cluster de bases de données Aurora Serverless v1

> **Important**
>
> AWS a annoncé la date de fin de vie d'Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

En règle générale, les clusters de bases de données Aurora Serverless v1 se mettent à l'échelle de façon transparente en fonction de la charge de travail. Toutefois, la capacité peut ne pas se mettre à l'échelle suffisamment vite pour répondre à des situations extrêmes, telles qu'une augmentation exponentielle des transactions. Dans ce cas, vous pouvez lancer l'opération de mise à l'échelle manuellement en définissant une nouvelle valeur de capacité. Une fois que vous avez défini la capacité de manière explicite, Aurora Serverless v1 procède automatiquement à la mise à l'échelle du cluster de bases de données. Il effectue cette opération en fonction de la période de stabilisation avant une diminution de capacité.

Vous pouvez définir explicitement une valeur spécifique pour la capacité d'un cluster de bases de données Aurora Serverless v1 avec l'AWS Management Console, l'AWS CLI ou l'API RDS.

Console

Vous pouvez définir la capacité d'un cluster de bases de données Aurora avec l'AWS Management Console.

Pour modifier un cluster de bases de données Aurora Serverless v1

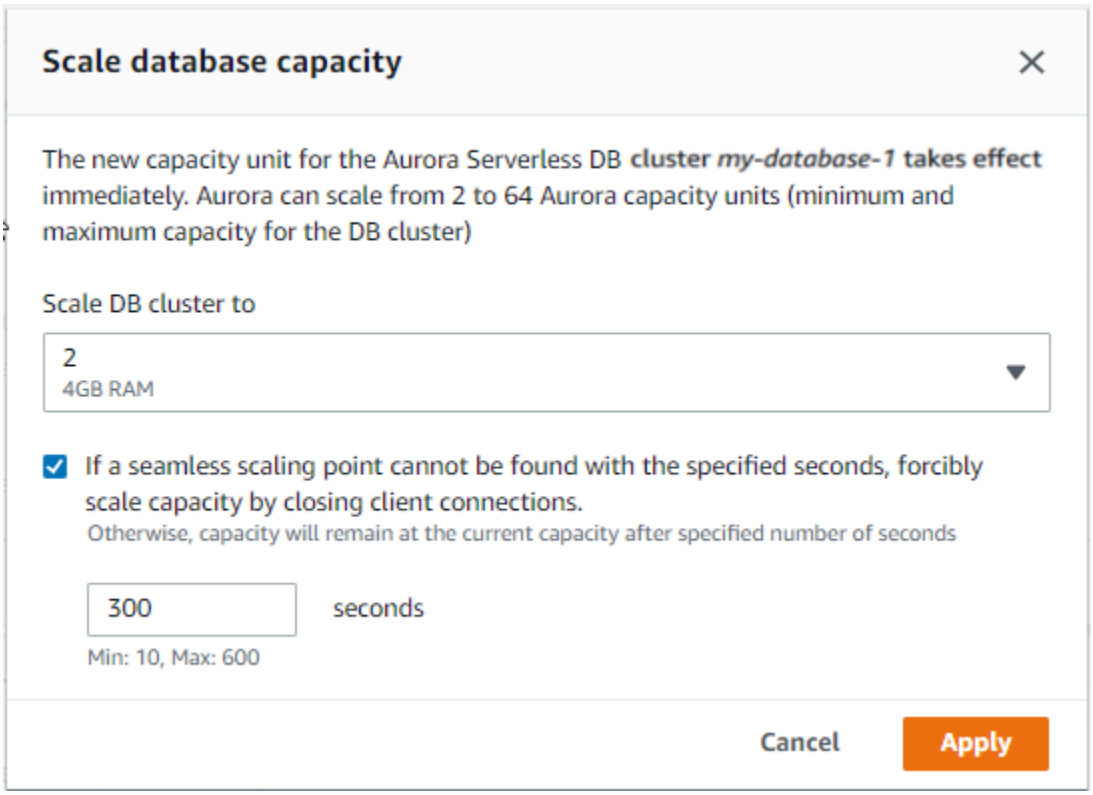
1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le cluster de bases de données Aurora Serverless v1 à modifier.
4. Pour Actions, sélectionnez Set capacity (Définir la capacité).
5. Dans la fenêtre Définir la capacité de la base de données, choisissez ce qui suit :

    a. Pour le sélecteur déroulant Mettre à l'échelle le cluster de bases de données sur, choisissez la nouvelle capacité souhaitée pour votre cluster de bases de données.

    b. Pour la case à cocher Si un point de mise à l'échelle transparente est introuvable, choisissez le comportement souhaité pour le paramètre `TimeoutAction` de votre cluster de bases de données Aurora Serverless v1, comme suit :

    - Cochez cette option pour ne pas modifier la capacité si Aurora Serverless v1 ne trouve pas de point de mise à l'échelle avant l'expiration.
    - Sélectionnez cette option pour forcer votre cluster de bases de données Aurora Serverless v1 à modifier sa capacité même s'il ne trouve pas de point de mise à l'échelle avant l'expiration. Cette option peut entraîner l'interruption des connexions Aurora Serverless v1 qui l'empêchent de trouver un point de mise à l'échelle.

    c. Dans le champ secondes, entrez la durée pendant laquelle votre cluster de bases de données Aurora Serverless v1 doit rechercher un point de mise à l'échelle avant l'expiration. Vous pouvez spécifier une valeur comprise entre 10 et 600 secondes (10 minutes). La valeur par défaut est de cinq minutes (300 secondes). L'exemple suivant force le cluster de bases de données Aurora Serverless v1 à réduire la capacité à 2 unités de capacité, même s'il ne trouve pas de point de mise à l'échelle dans les cinq minutes.



6. Choisissez Apply.

Pour plus d'informations sur les points de mise à l'échelle, `TimeoutAction` et les temps de stabilisation, consultez Mise à l'échelle automatique pour Aurora Serverless v1.

AWS CLI

Pour définir la capacité d'un cluster de bases de données Aurora Serverless v1 à partir de l'AWS CLI, exécutez la commande modify-current-db-cluster-capacity de l'AWS CLI et spécifiez l'option `--capacity`. Les valeurs de capacité valides sont notamment les suivantes :

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 et 256.
- Aurora PostgreSQL : 2, 4, 8, 16, 32, 64, 192 et 384.

Dans cet exemple, vous définissez la capacité d'un cluster de bases de données Aurora Serverless v1 nommé *`sample-cluster`* sur *`64`*.

```
aws rds modify-current-db-cluster-capacity --db-cluster-identifier sample-cluster --
capacity 64
```

API RDS

Vous pouvez définir la capacité d'un cluster de bases de données Aurora avec l'opération ModifyCurrentDBClusterCapacity de l'API. Spécifiez le paramètre `Capacity`. Les valeurs de capacité valides sont notamment les suivantes :

- Aurora MySQL: 1, 2, 4, 8, 16, 32, 64, 128 et 256.
- Aurora PostgreSQL : 2, 4, 8, 16, 32, 64, 192 et 384.

Affichage de clusters de bases de données Aurora Serverless v1

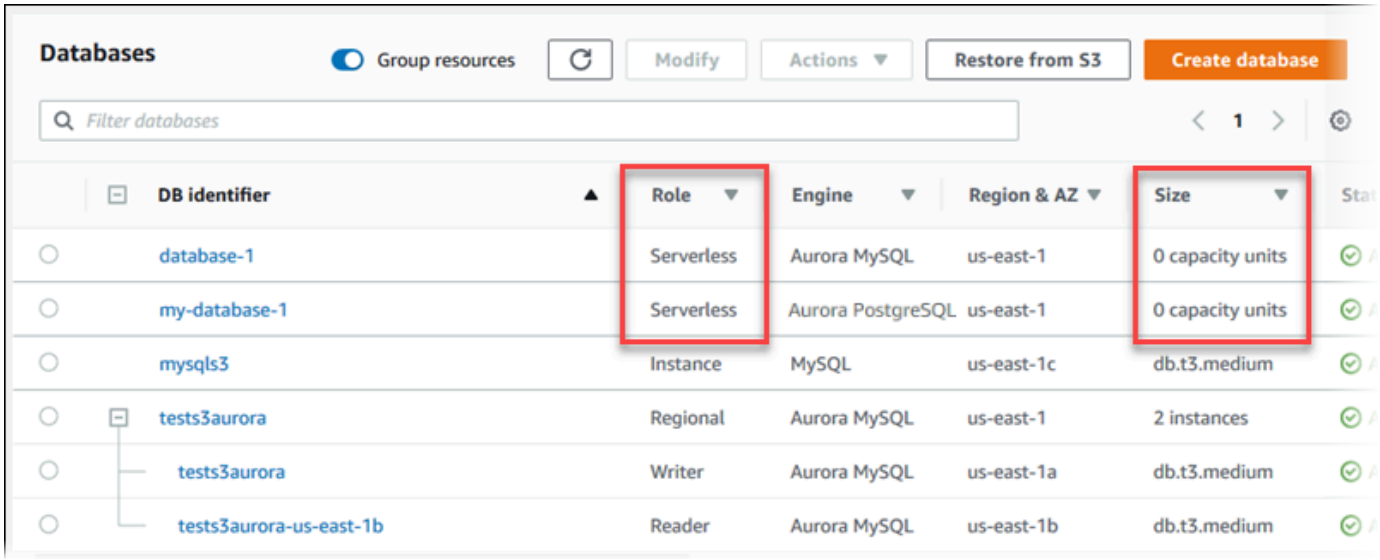
> **Important**
>
> AWS a annoncé la date de fin de vie d'Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

Après avoir créé un ou plusieurs clusters de bases de données Aurora Serverless v1, vous pouvez voir quels clusters de bases de données sont de type Sans serveur et de type Instance. Vous pouvez également consulter le nombre actuel d'unités de capacité Aurora (ACU) qu'utilise chaque cluster de bases de données Aurora Serverless v1. Chaque unité de capacité est une combinaison de traitement (UC) et de capacité mémoire (RAM).

Pour afficher vos clusters de bases de données Aurora Serverless v1

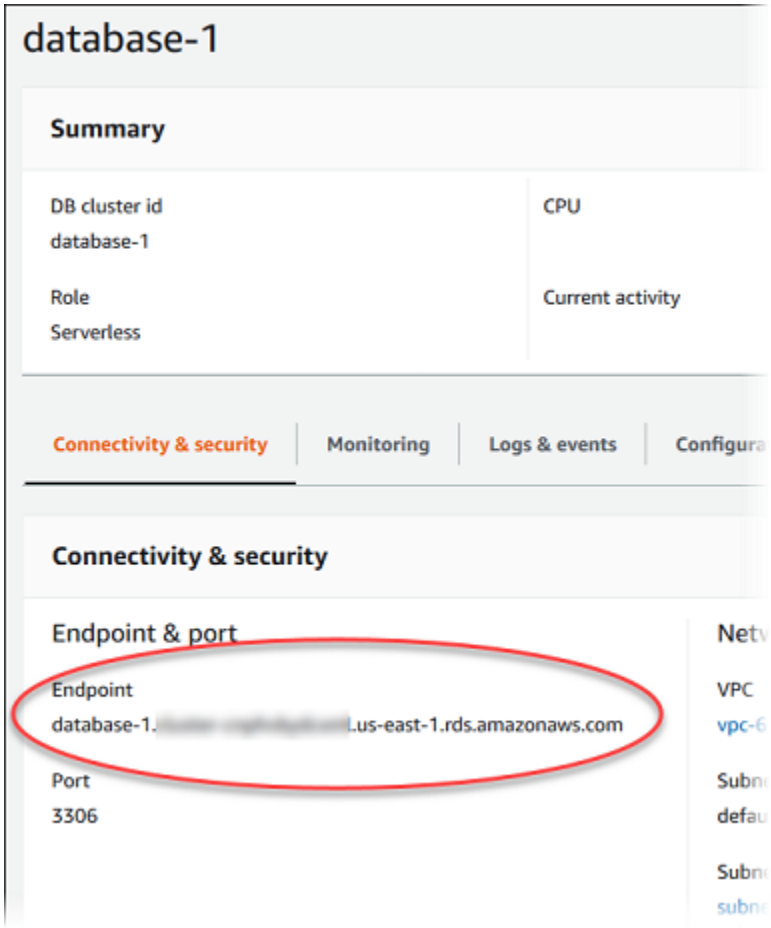
1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le coin supérieur droit d'AWS Management Console, sélectionnez la Région AWS dans laquelle vous avez créé les clusters de bases de données Aurora Serverless v1.
3. Dans la panneau de navigation, choisissez Bases de données.

Pour chaque cluster de bases de données, le type est indiqué sous Rôle. Les clusters de bases de données Aurora Serverless v1 affichent le type Sans serveur. Vous pouvez voir la capacité actuelle d'un cluster de bases de données Aurora Serverless v1 sous Taille.

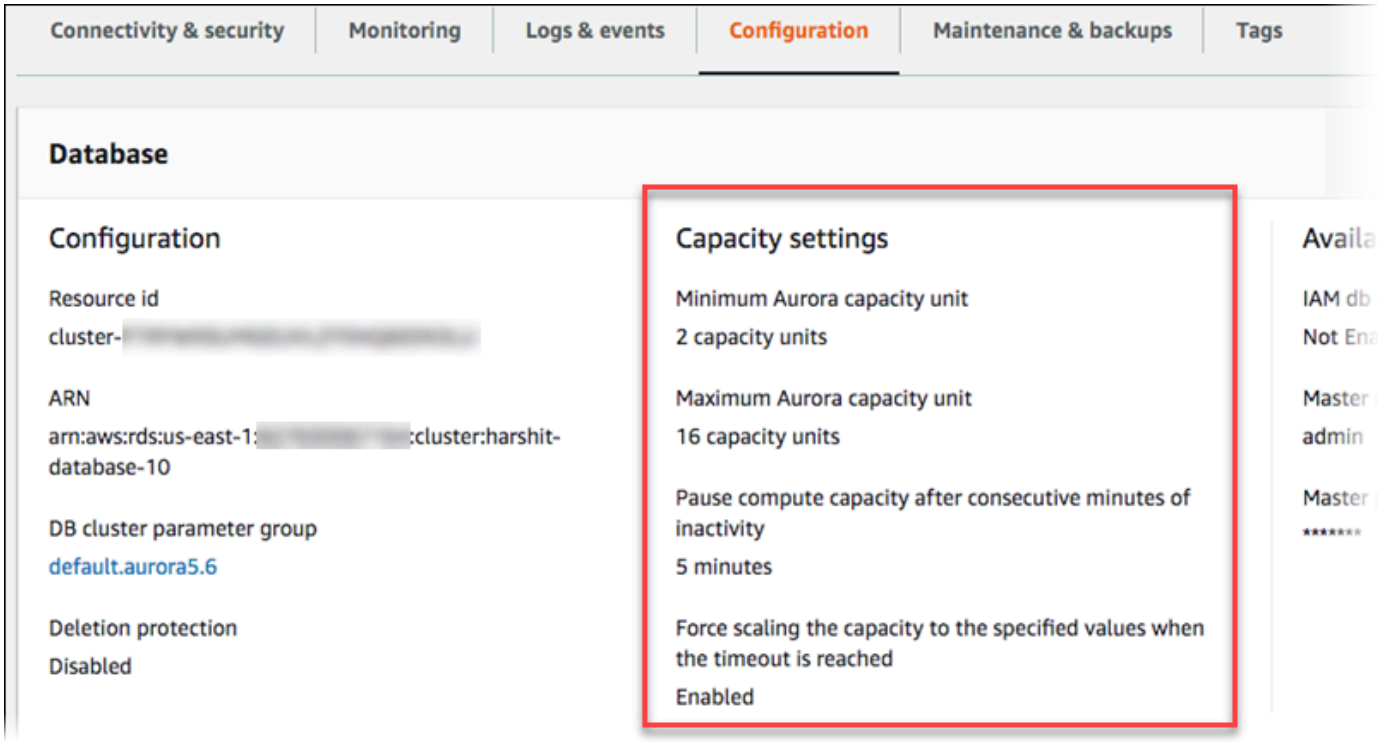


4. Choisissez le nom d'un cluster de bases de données Aurora Serverless v1 pour en afficher les détails.

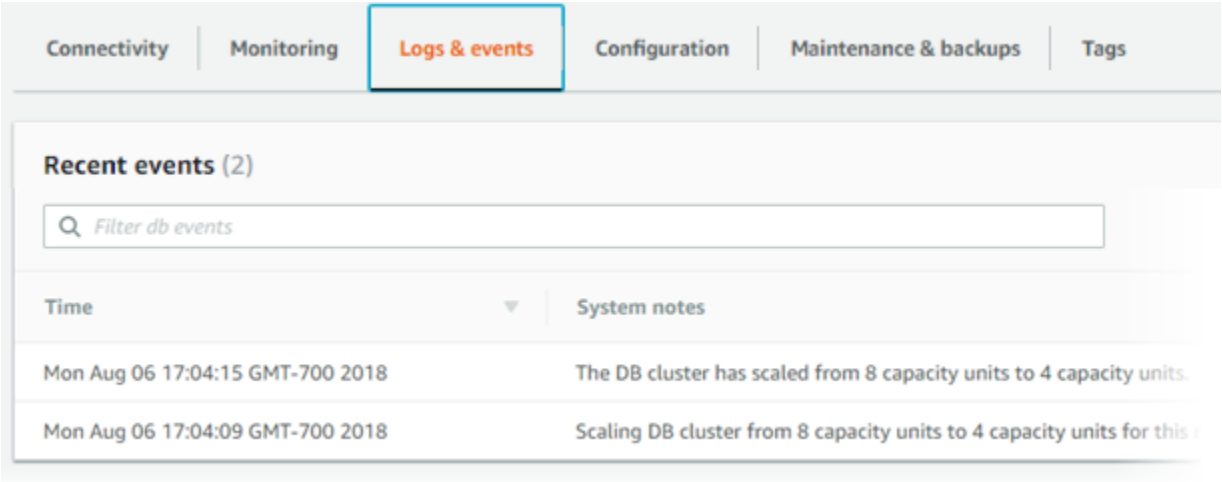
   Sous l'onglet Connectivité et sécurité , notez le point de terminaison de base de données. Utilisez ce point de terminaison pour vous connecter à votre cluster de bases de données Aurora Serverless v1.



Choisissez l'onglet Configuration pour afficher les paramètres de capacité.



Un événement de mise à l'échelle est généré lorsque le cluster de bases de donnée ajuste sa capacité (à la hausse ou à la baisse), se met en pause ou reprend son activité. Choisissez l'onglet Logs & events (Journaux et événements) pour afficher les événements récents. L'image suivante présente des exemples de ces événements.



Surveillance de la capacité et des événements de mise à l'échelle de votre cluster de bases de données Aurora Serverless v1

Vous pouvez afficher votre cluster de bases de données Aurora Serverless v1 dans CloudWatch pour surveiller la capacité qui lui est allouée avec la métrique `ServerlessDatabaseCapacity`. Vous pouvez également surveiller toutes les métriques Aurora CloudWatch standard, comme `CPUUtilization`, `DatabaseConnections`, `Queries`, etc.

Vous pouvez faire en sorte qu'Aurora publie certaines journaux de base de données ou leur totalité sur CloudWatch. Vous sélectionnez les journaux à publier en activant les paramètres de configuration tels que `general_log` et `slow_query_log` dans le groupe de paramètres de cluster de bases de données associé au cluster Aurora Serverless v1. Contrairement aux clusters approvisionnés, les clusters Aurora Serverless v1 ne requièrent pas que vous spécifiiez dans les paramètres de cluster de bases de données les types de journalisations à charger sur CloudWatch. Les clusters Aurora Serverless v1 chargent automatiquement tous les journaux disponibles. Lorsque vous désactivez un paramètre de configuration de journal, la publication du journal dans CloudWatch cesse. Vous pouvez également supprimer les journaux dans CloudWatch s'ils sont devenus inutiles.

Pour commencer à utiliser Amazon CloudWatch pour votre cluster de bases de données Aurora Serverless v1, consultez Affichage de journaux Aurora Serverless v1 avec Amazon CloudWatch. Pour en savoir plus sur la surveillance des clusters de bases de données Aurora via CloudWatch, consultez Surveillance des événements de journaux dans Amazon CloudWatch.

Pour vous connecter à un cluster de bases de données Aurora Serverless v1, utilisez le point de terminaison de base de données. Pour plus d'informations, consultez Connexion à un cluster de bases de données Amazon Aurora.

> **Note**
>
> Vous ne pouvez pas vous connecter directement à des instances de base de données spécifiques dans vos clusters de bases de données Aurora Serverless v1.

Suppression d'un cluster de bases de données Aurora Serverless v1

> **Important**
>
> AWS a annoncé la date de fin de vie d'Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

En fonction de la façon dont vous créez un cluster de bases de données Aurora Serverless v1, la protection contre la suppression peut être activée par défaut. Vous ne pouvez pas supprimer immédiatement un cluster de bases de données Aurora Serverless v1 pour lequel l'option Protection contre la suppression est activée. Pour supprimer les clusters de bases de données Aurora Serverless v1 protégés contre la suppression à l'aide de la AWS Management Console, modifiez d'abord le cluster afin de supprimer cette protection. Pour plus d'informations sur l'utilisation de la AWS CLI pour cette tâche, consultez AWS CLI.

Pour désactiver la protection contre la suppression à l'aide de la AWS Management Console

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Clusters de bases de données.
3. Choisissez votre cluster de bases de données Aurora Serverless v1 dans la liste.
4. Choisissez Modifier pour ouvrir la configuration de votre cluster de bases de données. La page Modifier le cluster de bases de données ouvre les paramètres, les paramètres de capacité et autres détails de configuration relatifs à votre cluster de bases de données Aurora Serverless v1. La protection contre la suppression se trouve dans la section Configuration supplémentaire.
5. Décochez la case Enable deletion protection (Activer la protection contre la suppression) dans la carte des propriétés Additional configuration (Configuration supplémentaire).
6. Choisissez Continuer. Le récapitulatif des modifications s'affiche.

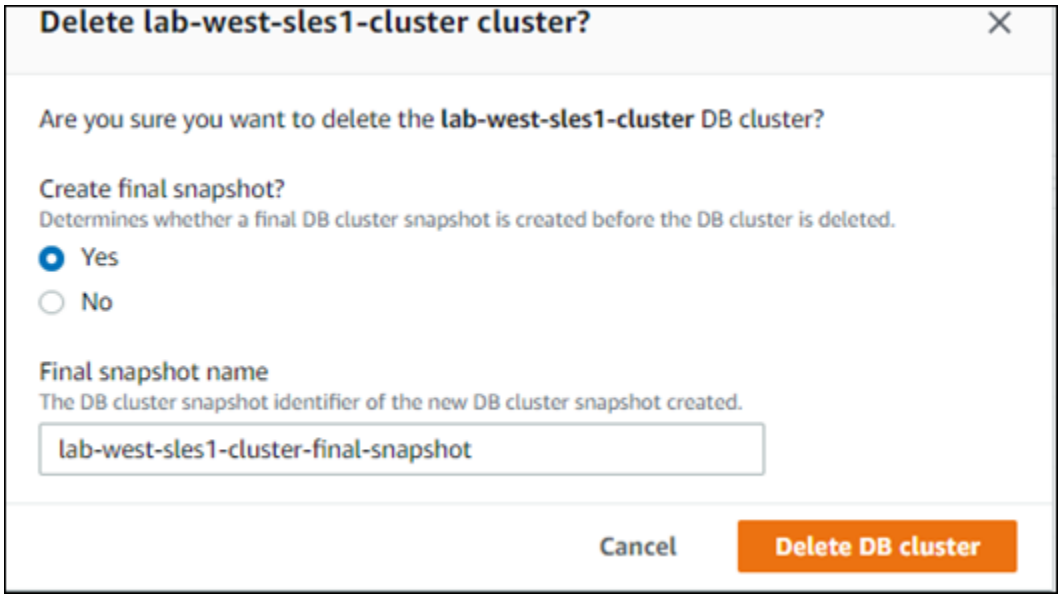
7. Choisissez Modifier le cluster pour accepter le récapitulatif des modifications. Vous pouvez également choisir Précédent pour modifier vos modifications ou Annuler pour les ignorer.

Lorsque la protection contre la suppression n'est plus active, vous pouvez supprimer votre cluster de bases de données Aurora Serverless v1 à l'aide de la AWS Management Console.

Console

Pour supprimer un cluster de bases de données Aurora Serverless v1

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la section Ressources, choisissez Clusters de bases de données.
3. Choisissez le cluster de bases de données Aurora Serverless v1 à supprimer.
4. Pour Actions, choisissez Supprimer. Vous êtes invité à confirmer que vous souhaitez supprimer votre cluster de bases de données Aurora Serverless v1.
5. Nous vous recommandons de conserver les options présélectionnées :
   - Oui pour Créer un instantané final ?
   - Le nom de votre cluster de bases de données Aurora Serverless v1, ainsi que `-final-snapshot` pour Nom de l'instantané final. Vous pouvez cependant modifier le nom de votre instantané final dans ce champ.



Si vous choisissez Non pour Créer un instantané final ?, vous ne pouvez pas restaurer votre cluster de bases de données à l'aide d'instantanés ni d'une reprise ponctuelle.

6. Choisissez Supprimer le cluster de bases de données.

Aurora Serverless v1 supprime votre cluster de bases de données. Si vous avez opté pour un instantané final, le statut de votre cluster de bases de données Aurora Serverless v1 indique « Sauvegarde » jusqu'à ce qu'il soit supprimé et n'apparaisse plus dans la liste.

AWS CLI

Avant de commencer, configurez AWS CLI avec votre identifiant de clé d'accès AWS, votre clé d'accès secrète AWS et la Région AWS dans laquelle se situe votre cluster de bases de données Aurora Serverless v1. Pour en savoir plus, consultez Principes de base de la configuration dans le guide de l'utilisateur AWS Command Line Interface.

Vous ne pouvez pas supprimer un cluster de bases de données Aurora Serverless v1 avant d'avoir désactivé la protection contre la suppression des clusters configurés avec cette option. Si vous essayez de supprimer un cluster protégé contre la suppression, le message d'erreur suivant s'affiche.

```
An error occurred (InvalidParameterCombination) when calling the DeleteDBCluster
  operation: Cannot delete protected Cluster, please disable deletion protection and
 try again.
```

Vous pouvez modifier le paramètre de protection contre la suppression de votre cluster de bases de données Aurora Serverless v1 à l'aide de la commande AWS CLI modify-db-cluster, comme illustré ci-dessous :

```
aws rds modify-db-cluster --db-cluster-identifier your-cluster-name --no-deletion-
protection
```

Cette commande renvoie les propriétés révisées pour le cluster de bases de données spécifié. Vous pouvez maintenant supprimer votre cluster de bases de données Aurora Serverless v1.

Nous vous recommandons de toujours créer un instantané final chaque fois que vous supprimez un cluster de bases de données Aurora Serverless v1. L'exemple d'utilisation du cluster AWS CLI delete-db-cluster suivant vous montre comment procéder. Vous indiquez le nom de votre cluster de bases de données et un nom pour l'instantané.

Pour Linux, macOS ou Unix :

```
aws rds delete-db-cluster --db-cluster-identifier \
  your-cluster-name --no-skip-final-snapshot \
  --final-db-snapshot-identifier name-your-snapshot
```

Pour Windows :

```
aws rds delete-db-cluster --db-cluster-identifier ^
  your-cluster-name --no-skip-final-snapshot ^
  --final-db-snapshot-identifier name-your-snapshot
```

Versions de moteur de base de données Aurora Serverless v1 et Aurora

> **Important**
>
> AWS a annoncé la date de fin de vie d'Aurora Serverless v1, qui sera le 31 mars 2025. Tous les clusters Aurora Serverless v1 qui ne seront pas migrés avant le 31 mars 2025 seront migrés vers Aurora Serverless v2 pendant la période de maintenance. Si la mise à niveau échoue, Amazon Aurora convertira le cluster sans serveur v1 en cluster provisionné avec la version de moteur équivalente pendant la période de maintenance. Le cas échéant, Amazon Aurora inscrira le cluster provisionné converti dans Amazon RDS Extended Support. Pour plus d'informations, consultez Support étendu RDS.

Aurora Serverless v1 est disponible dans certaines Régions AWS et uniquement pour des versions spécifiques d'Aurora MySQL et d'Aurora PostgreSQL. Pour obtenir la liste actuelle des Régions AWS prenant en charge Aurora Serverless v1, ainsi que les versions spécifiques d'Aurora MySQL et d'Aurora PostgreSQL disponibles dans chaque région, consultez Aurora Serverless v1.

Aurora Serverless v1 utilise le moteur de base de données Aurora qui lui est associé pour identifier les versions spécifiques prises en charge pour chaque moteur de base de données pris en charge, comme suit :

- Aurora MySQL sans serveur
- Aurora PostgreSQL sans serveur

Lorsque des versions mineures des moteurs de base de données sont disponibles pour Aurora Serverless v1, elles sont automatiquement appliquées dans les différentes Régions AWS où Aurora Serverless v1 est disponible. En d'autres termes, vous n'avez pas besoin de mettre à niveau votre cluster de bases de données Aurora Serverless v1 pour obtenir une nouvelle version mineure pour le moteur de base de données de votre cluster lorsqu'il est disponible pour Aurora Serverless v1.

Aurora MySQL sans serveur

Aurora Serverless v1 est disponible dans certaines Régions AWS et uniquement pour des versions spécifiques d'Aurora MySQL. Pour obtenir la liste actuelle des Régions AWS prenant en charge Aurora Serverless v1, ainsi que les versions spécifiques d'Aurora MySQL disponibles dans chaque région, consultez Aurora Serverless v1 avec Aurora MySQL.

Pour découvrir les améliorations et les correctifs de bogues pour Aurora MySQL version 2, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL version 2 dans Notes de mise à jour d'Aurora MySQL.

Pour utiliser une version plus récente d'Aurora MySQL, vous pouvez utiliser Aurora Serverless v2. Pour en savoir plus sur les Régions AWS et les versions Aurora MySQL que vous pouvez utiliser avec Aurora Serverless v2, consultez Aurora Serverless v2 avec Aurora MySQL. Pour en savoir plus sur l'utilisation d'Aurora Serverless v2, consultez Utiliser Aurora Serverless v2.

Aurora PostgreSQL sans serveur

Aurora Serverless v1 est disponible dans certaines Régions AWS et uniquement pour des versions spécifiques d'Aurora PostgreSQL. Pour obtenir la liste actuelle des Régions AWS prenant en charge Aurora Serverless v1, ainsi que les versions spécifiques d'Aurora PostgreSQL disponibles dans chaque région, consultez Aurora Serverless v1 avec Aurora PostgreSQL.

Si vous voulez utiliser Aurora PostgreSQL pour votre cluster de bases de données Aurora Serverless v1, vous pouvez utiliser les versions compatibles avec Aurora PostgreSQL 13. Les versions mineures pour Édition compatible avec Aurora PostgreSQL incluent uniquement des modifications rétrocompatibles. Votre cluster de bases de données Aurora Serverless v1 est mis à niveau de manière transparente quand une version mineure d'Aurora PostgreSQL devient disponible pour Aurora Serverless v1 dans votre Région AWS.

Pour utiliser une version plus récente d'Aurora PostgreSQL, vous pouvez utiliser Aurora Serverless v2. Pour les Régions AWS et les versions Aurora PostgreSQL que vous pouvez utiliser avec Aurora

Serverless v2, consultez Aurora Serverless v2 avec Aurora PostgreSQL. Pour en savoir plus sur l'utilisation d'Aurora Serverless v2, consultez Utiliser Aurora Serverless v2.

Mises à niveau automatiques des versions mineures pour Aurora Serverless v1

Lorsque des versions mineures des moteurs de base de données sont disponibles pour Aurora Serverless v1, elles sont automatiquement appliquées dans les différentes Régions AWS où Aurora Serverless v1 est disponible. En d'autres termes, vous n'avez pas besoin de mettre à niveau votre cluster de bases de données Aurora Serverless v1 pour obtenir une nouvelle version mineure pour le moteur de base de données de votre cluster lorsqu'il est disponible pour Aurora Serverless v1.

Utilisation de l’API de données Amazon RDS

L’API de données RDS (API de données), vous permet d’accéder à votre cluster de bases de données Aurora via une interface de services Web. L’API de données ne nécessite pas de connexion persistante au cluster de bases de données. Au lieu de cela, il fournit un point de terminaison HTTP sécurisé et une intégration avec AWS SDKs. Vous pouvez utiliser le point de terminaison pour exécuter des instructions SQL sans avoir à gérer de connexions.

Les utilisateurs n'ont pas besoin de transmettre des informations d'identification lors des appels à l'API de données, car l'API de données utilise les informations d'identification de base de données stockées dans AWS Secrets Manager. Pour stocker des informations d’identification dans Secrets Manager, les utilisateurs doivent disposer des autorisations appropriées pour utiliser Secrets Manager et l’API de données. Pour plus d’informations sur les autorisations accordées aux utilisateurs, consultez Autorisation de l’accès à l’API de données Amazon RDS.

Vous pouvez également utiliser l'API de données pour intégrer Amazon Aurora à d'autres AWS applications telles que AWS Lambda AWS AppSync, et AWS Cloud9. L’API de données permet d’utiliser AWS Lambda de façon plus sécurisée. Il vous permet d’avoir accès au cluster de bases de données sans avoir à configurer une fonction Lambda pour accéder aux ressources au sein d’un cloud privé virtuel (VPC). Pour plus d’informations, consultez AWS Lambda, AWS AppSync et AWS Cloud9.

Vous pouvez activer l’API de données lorsque vous créez le cluster de bases de données Aurora. Vous pouvez également modifier la configuration ultérieurement. Pour plus d’informations, consultez Activation de l’API de données Amazon RDS.

Après avoir activé l’API de données, vous pouvez également utiliser l’éditeur de requête pour exécuter des requête ad hoc sans configurer d’outil de requête pour accéder à Aurora dans un VPC. Pour plus d’informations, consultez Utilisation de l’éditeur de requêtes Aurora.

Rubriques

- Activation de l’API de données Amazon RDS
- Création d’un point de terminaison de VPC Amazon pour l’API de données Amazon RDS (AWS PrivateLink)
- Appel de l’API de données Amazon RDS
- Utilisation de la bibliothèque cliente Java pour l’API de données RDS
- Traitement des requêtes d’API Amazon RDS Data au format JSON
- Dépannage de l’API de données Amazon RDS
- Journalisation des appels d’API de données Amazon RDS avec AWS CloudTrail
- Surveillance des requêtes de l’API de données RDS avec Performance Insights

Disponibilité des régions et des versions de pour l’API de données Amazon RDS

Pour connaître les régions et les versions de moteurs prises en charge par l’API de données, consultez les sections suivantes.

Type de cluster	Disponibilité des régions et des versions
Aurora PostgreSQL provisionné et sans serveur v2	API de données avec Aurora PostgreSQL sans serveur v2 et provisionné
Aurora MySQL provisionné et sans serveur v2	API de données avec Aurora MySQL sans serveur v2 et provisionné
Aurora PostgreSQL sans serveur v1	API de données avec Aurora PostgreSQL sans serveur v1
Aurora MySQL sans serveur v1	API de données avec Aurora MySQL sans serveur v1

Si vous avez besoin de modules cryptographiques validés FIPS 140-2 lorsque vous accédez à l’API de données via une CLI ou une API, utilisez un point de terminaison FIPS. Pour plus d’informations sur les points de terminaison FIPS (Federal Information Processing Standard) disponibles, consultez

Federal Information Processing Standard (FIPS) 140-2 (Normes de traitement de l'information fédérale).

Utilisation IPv6 avec l'API de données Amazon RDS

L'API Amazon RDS Data prend en charge la IPv6 connectivité via des points de terminaison à double pile. Cela vous permet de vous connecter à l'API de données à l'aide d' IPv6 adresses tout en maintenant la rétrocompatibilité avec IPv4.

IPv6 assistance aux terminaux

L'API de données fournit des points de terminaison à double pile qui prennent en charge à la fois les connexions IPv4 et les connexions IPv6 . Ces points de terminaison utilisent le domaine `.aws` au lieu du domaine `.amazonaws.com` traditionnel.

Types de point de terminaison disponibles

Points de terminaison Dual-Stack publics

- Format : `rds-data.region.api.aws`
- Exemple : `rds-data.us-east-1.api.aws`

Points de terminaison FIPS Dual-Stack

- Format : `rds-data-fips.region.api.aws`
- Exemple : `rds-data-fips.us-east-1.api.aws`

PrivateLink IPv6 points de terminaison

- Disponible via les points de terminaison VPC avec support IPv6
- Permet une IPv6 connectivité privée au sein de votre VPC

IPv4Terminaux traditionnels uniquement

Les `.amazonaws.com` points de terminaison existants continuent de prendre en charge les IPv4 connexions uniquement :

- `rds-data.region.amazonaws.com`
- `rds-data-fips.region.amazonaws.com`

> **Note**
>
> Les points de terminaison existants restent inchangés afin de garantir la rétrocompatibilité avec les applications existantes.

Utilisation des points de IPv6 terminaison

Pour l'utiliser IPv6 avec l'API Data, mettez à jour votre application pour utiliser les nouveaux points de terminaison à double pile. Votre application sera automatiquement utilisée IPv6 si elle est disponible, ou reviendra à IPv4.

Pour obtenir des conseils généraux sur la configuration IPv6 dans votre VPC, consultez la section Migration vers IPv6 le guide de l'utilisateur Amazon VPC.

Vous pouvez configurer les IPv6 points de terminaison de deux manières :

- Utilisation d'une variable d'environnement : définissez `AWS_USE_DUALSTACK_ENDPOINT=true` dans votre IPv6 environnement. Le AWS CLI et AWS SDKs construira automatiquement les `api.aws` points de terminaison appropriés sans que vous ayez à spécifier le point de terminaison URLs manuellement.
- Utilisation d'un point de terminaison explicite URLs : spécifiez l'URL du point de terminaison à double pile directement dans vos AWS CLI commandes ou dans la configuration du SDK, comme indiqué dans les exemples ci-dessous.

AWS CLI configuration

Configurez les IPv6 points AWS CLI de terminaison pour les utiliser en spécifiant l'URL du point de terminaison :

Pour Linux, macOS ou Unix :

```
aws rds-data execute-statement \
    --endpoint-url https://rds-data.us-east-1.api.aws \
    --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster" \
    --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret" \
    --database "mydb" \
    --sql "SELECT * FROM users LIMIT 10"
```

Pour Windows :

```
aws rds-data execute-statement ^
    --endpoint-url https://rds-data.us-east-1.api.aws ^
    --resource-arn "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster" ^
    --secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret" ^
    --database "mydb" ^
    --sql "SELECT * FROM users LIMIT 10"
```

AWS Configuration du SDK

Configurez AWS SDKs pour utiliser des points de terminaison à double pile :

Python

```
import boto3

# Create RDS Data API client with IPv6 dual-stack endpoint
client = boto3.client(
    'rds-data',
    endpoint_url='https://rds-data.us-east-1.api.aws'
)

# Execute a SQL statement
response = client.execute_statement(
    resourceArn='arn:aws:rds:us-east-1:123456789012:cluster:my-cluster',
    secretArn='arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret',
    database='mydb',
    sql='SELECT * FROM users LIMIT 10'
)

print(response['records'])
```

Java

```
import software.amazon.awssdk.services.rdsdata.RdsDataClient;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.rdsdata.model.ExecuteStatementResponse;
import java.net.URI;

// Create RDS Data API client with IPv6 dual-stack endpoint
RdsDataClient client = RdsDataClient.builder()
```

```
    .endpointOverride(URI.create("https://rds-data.us-east-1.api.aws"))
    .build();

// Execute a SQL statement
ExecuteStatementRequest request = ExecuteStatementRequest.builder()
    .resourceArn("arn:aws:rds:us-east-1:123456789012:cluster:my-cluster")
    .secretArn("arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret")
    .database("mydb")
    .sql("SELECT * FROM users LIMIT 10")
    .build();

ExecuteStatementResponse response = client.executeStatement(request);
System.out.println(response.records());
```

JavaScript

```
const { RDSDataClient, ExecuteStatementCommand } = require("@aws-sdk/client-rds-
data");

// Create RDS Data API client with IPv6 dual-stack endpoint
const client = new RDSDataClient({
    endpoint: "https://rds-data.us-east-1.api.aws"
});

// Execute a SQL statement
const command = new ExecuteStatementCommand({
    resourceArn: "arn:aws:rds:us-east-1:123456789012:cluster:my-cluster",
    secretArn: "arn:aws:secretsmanager:us-east-1:123456789012:secret:my-secret",
    database: "mydb",
    sql: "SELECT * FROM users LIMIT 10"
});

const response = await client.send(command);
console.log(response.records);
```

Utilisation AWS PrivateLink avec IPv6

Vous pouvez créer des points de terminaison VPC pour l'API de données qui prennent en charge la IPv6 connectivité au sein de votre VPC. Pour obtenir des instructions détaillées sur la création de points de terminaison de VPC pour l’API de données, consultez Création d’un point de terminaison de VPC Amazon pour l’API de données Amazon RDS (AWS PrivateLink).

Lorsque vous créez un point de terminaison VPC pour le IPv6 support, assurez-vous que :

- Votre VPC et vos sous-réseaux sont configurés pour prendre en charge IPv6
- Les groupes de sécurité autorisent le IPv6 trafic sur les ports requis (généralement 443 pour HTTPS)
- Le réseau ACLs est configuré pour autoriser IPv6 le trafic

Considérations concernant la migration

Lors de la migration vers des IPv6 terminaux, tenez compte des points suivants :

- Migration progressive : vous pouvez migrer les applications progressivement en mettant à jour le terminal URLs une application à la fois.
- Compatibilité réseau : assurez-vous que votre infrastructure réseau est compatible IPv6 avant de procéder à la migration.
- Politiques de sécurité : mettez à jour les règles du groupe de sécurité et ACLs le réseau pour autoriser IPv6 le trafic si nécessaire.
- Surveillance : mettez à jour les configurations de surveillance et de journalisation pour gérer IPv6 les adresses.

> **Note**
>
> Adresses de connexion à la base de données : lorsque vous utilisez des IPv6 points de terminaison pour l'API de données, les connexions à la base de données sous-jacentes et les journaux de base de données affichent toujours IPv4 les adresses. Ce comportement est attendu et n'affecte pas les fonctionnalités de vos applications IPv6 activées.

Résolution des problèmes IPv6 de connectivité

Si vous rencontrez des problèmes de IPv6 connectivité, vérifiez les points suivants :

Configuration réseau

Vérifiez que votre réseau est compatible IPv6 et que IPv6 le routage est correctement configuré.

Résolution DNS

Assurez-vous que votre résolveur DNS peut résoudre les enregistrements AAAA pour les points de terminaison Dual-Stack.

Groupes de sécurité

Mettez à jour les règles du groupe de sécurité pour autoriser le IPv6 trafic sur les ports requis (généralement 443 pour HTTPS).

Bibliothèques clientes

Vérifiez que vos bibliothèques clientes HTTP prennent en charge IPv6 la connectivité à double pile.

Limites de l’API de données Amazon RDS

L’API de données RDS présente les limitations suivantes :

- Vous ne pouvez exécuter des requêtes d’API de données que sur des instances en écriture dans un cluster de bases de données. Toutefois, les instances en écriture peuvent accepter à la fois des requêtes d’écriture et de lecture.
- Les bases de données globales Aurora permettent d’activer l’API de données aussi bien sur le cluster de bases de données principal que sur le cluster secondaire. Toutefois, un cluster secondaire ne possède aucune instance en écriture tant qu’il n’est pas devenu le cluster principal. L’API de données requiert un accès à l’instance en écriture pour traiter les requêtes, y compris les requêtes en lecture. Ainsi, les requêtes de lecture et d’écriture envoyées à un cluster secondaire échouent tant qu’il ne dispose pas d’une instance en écriture. Dès qu’un cluster secondaire est promu et qu’une instance en écriture devient disponible, les requêtes de l’API de données sur cette base de données aboutissent.
- L’API de données n’est pas prise en charge sur les classes d’instance de base de données T.
- L’API de données RDS ne prend pas en charge certains types de données lorsqu’elle est utilisée avec Aurora Serverless v2 ou des clusters de bases de données provisionnés. Pour obtenir la liste des types pris en charge, consultez the section called “Comparaison de l’API de données avec Aurora Serverless v2 et Aurora Serverless v1”.
- Pour les bases de données Aurora PostgreSQL version 14 et ultérieures, l’API de données prend uniquement en charge `scram-sha-256` pour le chiffrement des mots de passe.

- La taille de réponse est limitée à 1 Mio. Si l’appel renvoie plus de 1 Mio de données de réponse, l’appel est arrêté.
- Le nombre maximal de demandes par seconde est 1 000 pour Aurora Serverless v1. Aucune limite n’est imposée pour les autres bases de données prises en charge.
- La taille de l’API de données ne doit pas dépasser 64 Ko par ligne dans le jeu de résultat renvoyé par la base de données. Assurez-vous que la taille de chaque ligne d’un jeu de résultat est inférieure ou égale à 64 Ko.

Comparaison des comportements de l’API de données Amazon RDS entre Aurora Serverless v2 et les clusters provisionnés avec les clusters Aurora Serverless v1

Les dernières améliorations apportées aux API de données Amazon RDS les rendent compatibles avec les clusters fonctionnant sur les versions récentes des moteurs PostgreSQL ou MySQL. Ces clusters peuvent être configurés pour utiliser Aurora Serverless v2 ou provisionner des classes d’instance telles que `db.r6g` ou `db.r6i`.

Les sections suivantes décrivent les différences de l’API de données Amazon RDS entre Aurora Serverless v2 et les clusters de bases de données provisionnés, ainsi que les clusters de bases de données Aurora Serverless v1. Les clusters de bases de données Aurora Serverless v1 utilisent le mode moteur `serverless`. Les clusters de bases de données provisionnés utilisent le mode moteur `provisioned`. Un cluster de bases de données Aurora Serverless v2 utilise le mode moteur `provisioned` et contient une ou plusieurs instances de base de données Aurora Serverless v2 avec la classe d’instance `db.serverless`.

Nombre maximal de demandes par seconde

Aurora Serverless v1

Les API de données peuvent traiter jusqu’à 1 000 demandes par seconde.

Aurora Serverless v2

Les API de données peuvent traiter un nombre illimité de demandes par seconde.

Activation ou désactivation de l’API de données Amazon RDS sur une base de données existante

Aurora Serverless v1

- Avec l’API Amazon RDS : utilisez l’opération `ModifyCluster` et indiquez `True` ou `False`, selon le cas, pour le paramètre `EnableHttpEndpoint`.
- Avec AWS CLI : utilisez l’opération `modify-db-cluster` avec l’option `--enable-http-endpoint` ou `--no-enable-http-endpoint`, selon le cas.

Aurora Serverless v2

- Avec l’API Amazon RDS : utilisez les opérations `EnableHttpEndpoint` et `DisableHttpEndpoint`.
- Avec AWS CLI : utilisez les opérations `enable-http-endpoint` et `disable-http-endpoint`.

Événements CloudTrail

Aurora Serverless v1

Les événements provenant des appels de l’API de données sont des événements de gestion. Par défaut, ces événements sont consignés automatiquement dans un journal d’activité. Pour plus d’informations, consultez the section called “Exclusion d’événements d’API de données d’un journal d’activité CloudTrail (Aurora Serverless v1 uniquement)”.

Aurora Serverless v2

Les événements provenant des appels de l’API de données sont des événements de données. Par défaut, ces événements sont automatiquement exclus du journal d’activité. Pour plus d’informations, consultez the section called “Inclusion d’événements d’API de données dans un journal d’activité CloudTrail”.

Prise en charge d’instructions multiples

Aurora Serverless v1

- Les instructions multiples ne sont pas prises en charge avec Aurora MySQL.

- Les instructions multiples renvoient uniquement la première réponse à la requête dans Aurora PostgreSQL.

Aurora Serverless v2

Les instructions multiples ne sont pas prises en charge. Toute tentative d'exécution d'instructions multiples dans un seul appel d'API renvoie "`An error occurred (ValidationException) when calling the ExecuteStatement operation: Multistatements aren't supported.`". Pour exécuter des instructions multiples, effectuez des appels d'API `ExecuteStatement` distincts ou utilisez `BatchExecuteStatement` pour le traitement par lots.

L'exemple suivant illustre le message d'erreur renvoyé par un appel d'API qui tente d'exécuter une instruction multiple.

```
 aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM your_table; Select * FROM next_table;

                                "An error occurred (ValidationException) when calling
 the ExecuteStatement operation: Multistatements aren't supported.
```

L'exemple suivant exécute des instructions multiples avec des appels d'API `ExecuteStatement` distincts.

```
aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM your_table;"

aws rds-data execute-statement \
    --resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
    --secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
    --database "your_database" \
    --sql "SELECT * FROM next_table;"
```

Demandes simultanées pour le même ID de transaction

Aurora Serverless v1

Les demandes suivantes sont mises en attente jusqu'à ce que la demande en cours soit achevée. Votre application doit traiter les erreurs de délai d'expiration en cas d'attente prolongée.

Aurora Serverless v2

Lorsque l'API de données reçoit plusieurs demandes avec le même ID de transaction, elle renvoie immédiatement cette erreur :

`DatabaseErrorException: Transaction is still running a query`

Cette erreur se produit dans deux situations :

- Votre application effectue des demandes asynchrones (comme des promesses JavaScript) en utilisant le même ID de transaction.
- Une demande précédente avec cet ID de transaction est toujours en cours de traitement.

L'exemple suivant illustre toutes les demandes exécutées en parallèle avec `promise.all()`.

```
const api_calls = [];
for (let i = 0; i < 10; i++) {
api_calls.push(
    client.send(
    new ExecuteStatementCommand({
        ...params,
        sql: `insert into table_name values (i);`,
        transactionId
    })
    )
);
}
await Promise.all(api_calls);
```

Pour résoudre cette erreur, attendez que la demande en cours se termine avant d'envoyer une autre demande avec le même ID de transaction, ou supprimez l'ID de transaction afin d'autoriser les demandes parallèles.

L'exemple suivant illustre un appel d'API qui utilise une exécution séquentielle avec le même ID de transaction.

```
 for (let i = 0; i < 10; i++) {
```

```
    await client.send(
    new ExecuteStatementCommand({
        ...params,
        sql: `insert into table_name values (i);`,
        transactionId
    })
    ).promise()
);
}
```

Comportement de l’instruction BatchExecuteStatement

Pour plus d’informations sur l’instruction `BatchExecuteStatement`, consultez BatchExecuteStatement.

Aurora Serverless v1

L’objet des champs générés dans le résultat de la mise à jour inclut les valeurs insérées.

Aurora Serverless v2

- L’objet des champs générés dans le résultat de la mise à jour inclut les valeurs insérées dans Aurora MySQL.
- L’objet des champs générés est vide dans Aurora PostgreSQL.

Comportement de l’instruction ExecuteSQL

Pour plus d’informations sur l’instruction `ExecuteSQL`, consultez ExecuteSQL.

Aurora Serverless v1

L’opération `ExecuteSQL` est obsolète.

Aurora Serverless v2

L’opération `ExecuteSQL` n’est pas prise en charge.

Comportement de l’instruction ExecuteStatement

Pour plus d’informations sur l’instruction `ExecuteStatement`, consultez ExecuteStatement.

Aurora Serverless v1

Le paramètre `ExecuteStatement` prend en charge la récupération de colonnes de tableaux multidimensionnels ainsi que de tous les types de données avancés.

Aurora Serverless v2

Le paramètre `ExecuteStatement` ne prend pas en charge les colonnes de tableaux multidimensionnels. Certains types de données PostgreSQL ne sont également pas pris en charge, en particulier les types géométriques et les types monétaires. Lorsqu'une API de données rencontre un type de données non pris en charge, elle renvoie cette erreur : `UnsupportedResultException: The result contains the unsupported data type data_type`.

Pour contourner ce problème, convertissez le type de données non pris en charge en `TEXT`. L'exemple suivant convertit un type de données non pris en charge en `TEXT`.

```
SELECT custom_type::TEXT FROM my_table;--
ORSELECT CAST(custom_type AS TEXT) FROM my_table;
```

Pour obtenir la liste des types de données pris en charge pour chaque moteur de base de données Aurora, consultez Référence des opérations de l'API de données.

Comportement des paramètres de schéma

Aurora Serverless v1

Le paramètre `Schema` n'est pas pris en charge. Si le paramètre `Schema` est inclus dans un appel d'API, il est ignoré par l'API de données.

Aurora Serverless v2

Le paramètre `Schema` est obsolète. Si le paramètre `Schema` est inclus dans un appel d'API, l'API de données renvoie l'erreur suivante : `ValidationException: The schema parameter isn't supported`. L'exemple suivant illustre un appel de l'API de données qui renvoie l'erreur `ValidationException`.

```
aws rds-data execute-statement \
--resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
--secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
```

```
--database "your_database" \
--schema "your_schema" \
--sql "SELECT * FROM your_table LIMIT 10"
```

Pour résoudre ce problème, supprimez le paramètre `Schema` de votre appel d'API.

L'exemple suivant illustre un appel de l'API de données où le paramètre `Schema` a été supprimé.

```
aws rds-data execute-statement \
--resource-arn "arn:aws:rds:region:account:cluster:cluster-name" \
--secret-arn "arn:aws:secretsmanager:region:account:secret:secret-name" \
--database "your_database" \
--sql "SELECT * FROM your_table LIMIT 10"
```

Autorisation de l'accès à l'API de données Amazon RDS

Les utilisateurs peuvent invoquer les opérations de l'API de données Amazon RDS (API de données) seulement s'ils sont autorisés à le faire. Vous pouvez autoriser un utilisateur à utiliser l'API de données en joignant une politique Gestion des identités et des accès AWS (IAM) qui définit ses privilèges. Vous pouvez également attacher la politique à un rôle si vous utilisez les rôles IAM. Une politique AWS gérée inclut `AmazonRDSDataFullAccess` des autorisations pour l'API de données.

La `AmazonRDSDataFullAccess` politique inclut également des autorisations permettant à l'utilisateur d'obtenir la valeur d'un secretAWS Secrets Manager. Les utilisateurs doivent utiliser Secrets Manager pour stocker les secrets qu'ils peuvent utiliser dans leurs appels à l'API de données. L'utilisation de secrets signifie que les utilisateurs n'ont pas besoin d'inclure des informations d'identification de base de données pour les ressources qu'ils ciblent dans leurs appels à l'API de données. L'API de données appelle Secrets Manager de manière transparente, qui autorise (ou refuse) la demande de secret de l'utilisateur. Pour plus d'informations sur la configuration des secrets à utiliser avec l'API de données, consultez Stockage des identifiants de base de données dans AWS Secrets Manager.

La politique `AmazonRDSDataFullAccess` fournit un accès complet (via l'API de données) aux ressources. Vous pouvez limiter la portée en définissant vos propres politiques qui spécifient l'ARN (Amazon Resource Name) d'une ressource.

Par exemple, la politique suivante présente un exemple des autorisations minimales requises pour qu'un utilisateur puisse accéder à l'API de données pour le cluster de bases de données identifié par

son ARN. La politique comprend les autorisations nécessaires pour accéder à Secrets Manager et obtenir l'autorisation sur l'instance de base de données pour l'utilisateur.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerDbCredentialsAccess",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*"
        },
        {
            "Sid": "RDSDataServiceAccess",
            "Effect": "Allow",
            "Action": [
                "rds-data:BatchExecuteStatement",
                "rds-data:BeginTransaction",
                "rds-data:CommitTransaction",
                "rds-data:ExecuteStatement",
                "rds-data:RollbackTransaction"
            ],
            "Resource": "arn:aws:rds:us-east-2:111122223333:cluster:prod"
        }
    ]
}
```

Nous vous recommandons d'utiliser un ARN spécifique pour l'élément « Resources » dans les instructions de votre politique (comme indiqué dans l'exemple) plutôt qu'un caractère générique (*).

Utilisation de l'autorisation basée sur les balises

L'API de données RDS (API de données) et Secrets Manager prennent en charge l'autorisation basée sur les balises. Les balises sont des paires clé-valeur qui étiquettent une ressource, telle qu'un cluster RDS, avec une valeur de chaîne supplémentaire, par exemple :

- `environment:production`

- `environment:development`

Vous pouvez appliquer des balises à vos ressources pour la répartition des coûts, la prise en charge des opérations, le contrôle d'accès et bien d'autres raisons. (Si vous n'avez pas appliqué de balises à vos ressources et que vous souhaitez le faire, vous pouvez en savoir plus en consultant Balisage de ressources Amazon RDS.) Vous pouvez utiliser les balises de vos instructions de politique pour limiter l'accès aux clusters RDS étiquetés avec ces balises. Par exemple, un cluster de bases de données Aurora peut avoir des balises qui identifient son environnement en tant que production ou développement.

L'exemple suivant montre comment utiliser les balises dans vos instructions de politique. Cette instruction exige que le cluster et le secret transmis dans la demande d'API de données aient une balise `environment:production`.

Voici comment la politique est appliquée : lorsqu'un utilisateur effectue un appel via l'API de données, la demande est envoyée au service. L'API de données vérifie d'abord que l'ARN du cluster transmis dans la demande contient la balise `environment:production`. Elle appelle ensuite Secrets Manager pour récupérer la valeur du secret de l'utilisateur dans la requête. Secrets Manager vérifie également que le secret de l'utilisateur est étiqueté avec `environment:production`. Si tel est le cas, l'API de données utilise ensuite la valeur extraite en tant que mot de passe de base de données de l'utilisateur. Enfin, si cette valeur est également correcte, la demande d'API de données est appelée avec succès pour l'utilisateur.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "SecretsManagerDbCredentialsAccess",
            "Effect": "Allow",
            "Action": [
                 "secretsmanager:GetSecretValue"
               ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*",
            "Condition": {
                    "StringEquals": {
                        "aws:ResourceTag/environment": [
                                        "production"
```

```
                                        ]
                    }
                }
            },
            {
                "Sid": "RDSDataServiceAccess",
                "Effect": "Allow",
                "Action": [
                      "rds-data:*"
                   ],
                "Resource": "arn:aws:rds:us-east-2:111122223333:cluster:*",
                "Condition": {
                        "StringEquals": {
                            "aws:ResourceTag/environment": [
                                        "production"
                                        ]
                        }
                }
            }
        ]
}
```

L'exemple montre des actions distinctes pour `rds-data` et `secretsmanager` pour l'API de données et Secrets Manager. Cependant, vous pouvez combiner des actions et définir des conditions de balise de différentes manières afin de prendre en charge vos cas d'utilisation spécifiques. Pour plus d'informations, consultez Utilisation des stratégies basées sur l'identité (stratégies IAM) pour Secrets Manager.

Dans l'élément « Condition » de la politique, vous pouvez choisir les clés de balise parmi les suivantes :

- `aws:TagKeys`
- `aws:ResourceTag/${TagKey}`

Pour en savoir plus sur les balises de ressources et sur leur utilisation`aws:TagKeys`, consultez la section Contrôle de l'accès aux AWS ressources à l'aide de balises de ressources.

> **Note**
>
> À la fois l'API de données et AWS Secrets Manager les utilisateurs autorisés. Si vous ne disposez pas des autorisations requises pour toutes les actions définies dans une stratégie, une erreur `AccessDeniedException` s'affiche.

Stockage des identifiants de base de données dans AWS Secrets Manager

Lorsque vous appelez l'API de données Amazon RDS (API de données), vous pouvez transmettre des informations d'identification pour le cluster de bases de données en utilisant un secret dans Secrets Manager. Pour ce faire, vous devez spécifier le nom du secret ou son ARN (Amazon Resource Name).

Pour stocker les informations d'identification de cluster de bases de données dans un secret

1. Utilisez Secrets Manager pour créer un secret qui contient les informations d'identification du cluster de bases de données Aurora.

   Pour obtenir des instructions, consultez Création d'un secret de base de données dans le Guide de l'utilisateur AWS Secrets Manager.
2. Utilisez la console Secrets Manager pour afficher les détails du secret que vous avez créé ou exécutez la `aws secretsmanager describe-secret` AWS CLI commande.

   Notez le nom et l'ARN du secret. Vous pouvez les utiliser dans les appels à l'API de données.

Pour plus d'informations sur l'utilisation de Secrets Manager, consultez le Guide de l'utilisateur AWS Secrets Manager.

Pour comprendre comment Amazon Aurora assure la gestion des identités et des accès, consultez Comment Amazon Aurora fonctionne avec IAM.

Pour en savoir plus sur la création d'une stratégie IAM, consultez Création de stratégies IAM dans le Guide de l'utilisateur IAM. Pour en savoir plus sur l'ajout d'une stratégie IAM, consultez Ajout et suppression de stratégies IAM dans le Guide de l'utilisateur IAM.

Activation de l’API de données Amazon RDS

L’API de données Amazon RDS (API de données) doit être activée sur le cluster de bases de données Aurora avant de pouvoir être utilisée. Vous pouvez activer l’API de données lorsque vous créez ou modifiez le cluster de bases de données.

> **Note**
>
> La disponibilité de l’API de données pour votre cluster dépend de votre version d’Aurora, de votre moteur de base de données et de votre région AWS. Sur les anciennes versions d’Aurora, l’API de données ne peut être utilisée qu’avec des clusters Aurora Serverless v1. Sur les nouvelles versions d’Aurora, l’API de données fonctionne avec des clusters qui utilisent à la fois des instances provisionnées et des instances Aurora Serverless v2. Pour vérifier si votre cluster peut utiliser l’API de données, consultez Régions et moteurs de base de données Aurora pris en charge pour l’API de données RDS.

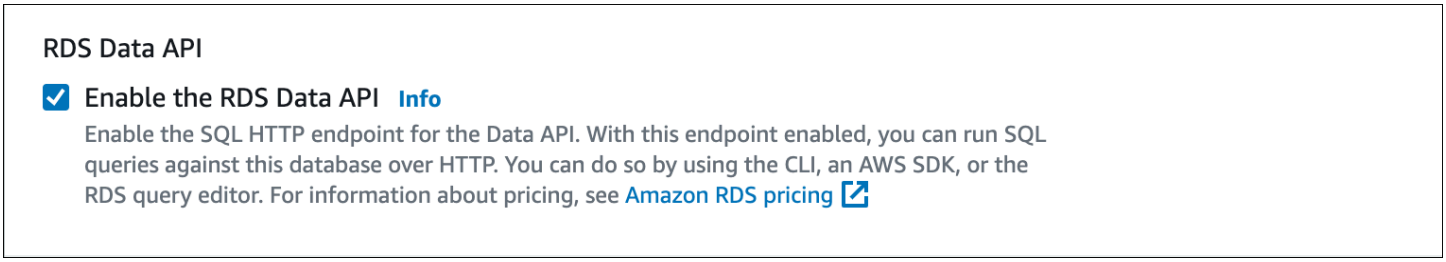
Rubriques

Activation de l’API de données RDS lors de la création d’une base de données

Lorsque vous créez une base de données compatible avec l’API de données RDS (API de données), vous pouvez activer cette fonctionnalité. Les procédures suivantes décrivent comment procéder lorsque vous utilisez la AWS Management Console, l’AWS CLI ou l’API RDS.

Console

Pour activer l’API de données lors de la création d’un cluster de bases de données, cochez la case Activer l’API de données RDS dans la section Connectivité de la page Créer une base de données, comme illustré dans la capture d’écran suivante.



Pour obtenir des instructions sur la création d'un cluster de bases de données Aurora compatible avec l'API de données RDS, consultez les informations suivantes :

- Pour Aurora Serverless v2 et les clusters provisionnés : Création d'un cluster de bases de données Amazon Aurora
- Pour Aurora Serverless v1 : Création d'un cluster de bases de données Aurora Serverless v1

AWS CLI

Pour activer l'API de données lors de la création d'un cluster de bases de données Aurora, exécutez la commande AWS CLI create-db-cluster avec l'option `--enable-http-endpoint`.

Voici un exemple de création d'un cluster de bases de données Aurora PostgreSQL où l'API de données est activée.

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
    --db-cluster-identifier my_pg_cluster \
    --engine aurora-postgresql \
    --enable-http-endpoint
```

Pour Windows :

```
aws rds create-db-cluster ^
    --db-cluster-identifier my_pg_cluster ^
    --engine aurora-postgresql ^
    --enable-http-endpoint
```

API RDS

Pour activer l'API de données lors de la création d'un cluster de bases de données Aurora, utilisez l'opération CreateDBCluster en définissant le paramètre `EnableHttpEndpoint` sur `true`.

Activation de l’API de données RDS sur une base de données existante

Vous pouvez modifier un cluster de bases de données compatible avec l’API de données RDS (API de données) pour activer ou désactiver cette fonctionnalité.

Rubriques

- Activation ou désactivation de l’API de données (Aurora Serverless v2 et base de données provisionnée)
- Activation ou désactivation de l’API de données (Aurora Serverless v1 uniquement)

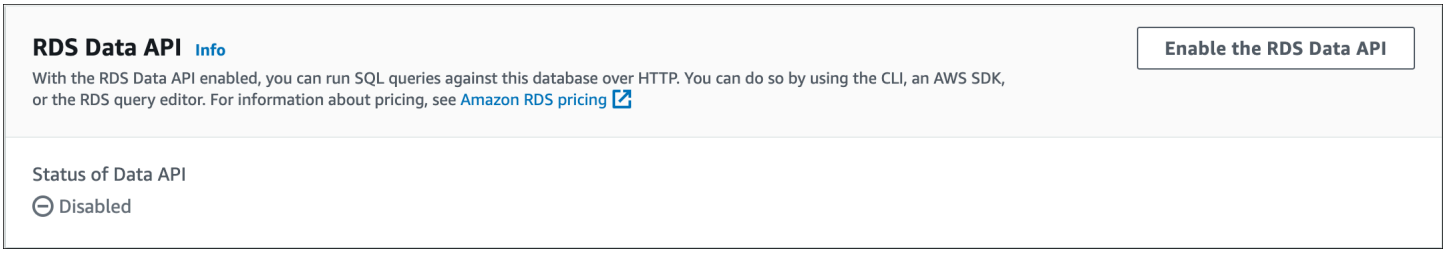
Activation ou désactivation de l’API de données (Aurora Serverless v2 et base de données provisionnée)

Utilisez les procédures suivantes pour activer ou désactiver l’API de données sur Aurora Serverless v2 et les bases de données provisionnées. Pour activer ou désactiver l’API de données sur les bases de données Aurora Serverless v1, utilisez les procédures indiquées dans the section called “Activation ou désactivation de l’API de données (Aurora Serverless v1 uniquement)”.

Console

Si votre cluster de bases de données prend en charge cette fonctionnalité, vous pouvez activer ou désactiver l’API de données directement depuis la console RDS. Pour ce faire, ouvrez la page des détails du cluster de la base de données sur laquelle vous souhaitez activer ou désactiver l’API de données, puis dans l’onglet Connectivité et sécurité, accédez à la section API de données RDS. Cette section affiche l’état de l’API de données et vous permet de l’activer ou de la désactiver.

Dans la capture d’écran suivante, l’API de données RDS est désactivée.



AWS CLI

Pour activer ou désactiver l’API de données sur une base de données existante, exécutez la commande AWS CLI enable-http-endpoint ou disable-http-endpoint et indiquez l’ARN de votre cluster de bases de données.

Dans l'exemple suivant, l'API de données est activée.

Pour Linux, macOS ou Unix :

```
aws rds enable-http-endpoint \
    --resource-arn cluster_arn
```

Pour Windows :

```
aws rds enable-http-endpoint ^
    --resource-arn cluster_arn
```

API RDS

Pour activer ou désactiver l'API de données sur une base de données existante, utilisez les opérations EnableHttpEndpoint et DisableHttpEndpoint.

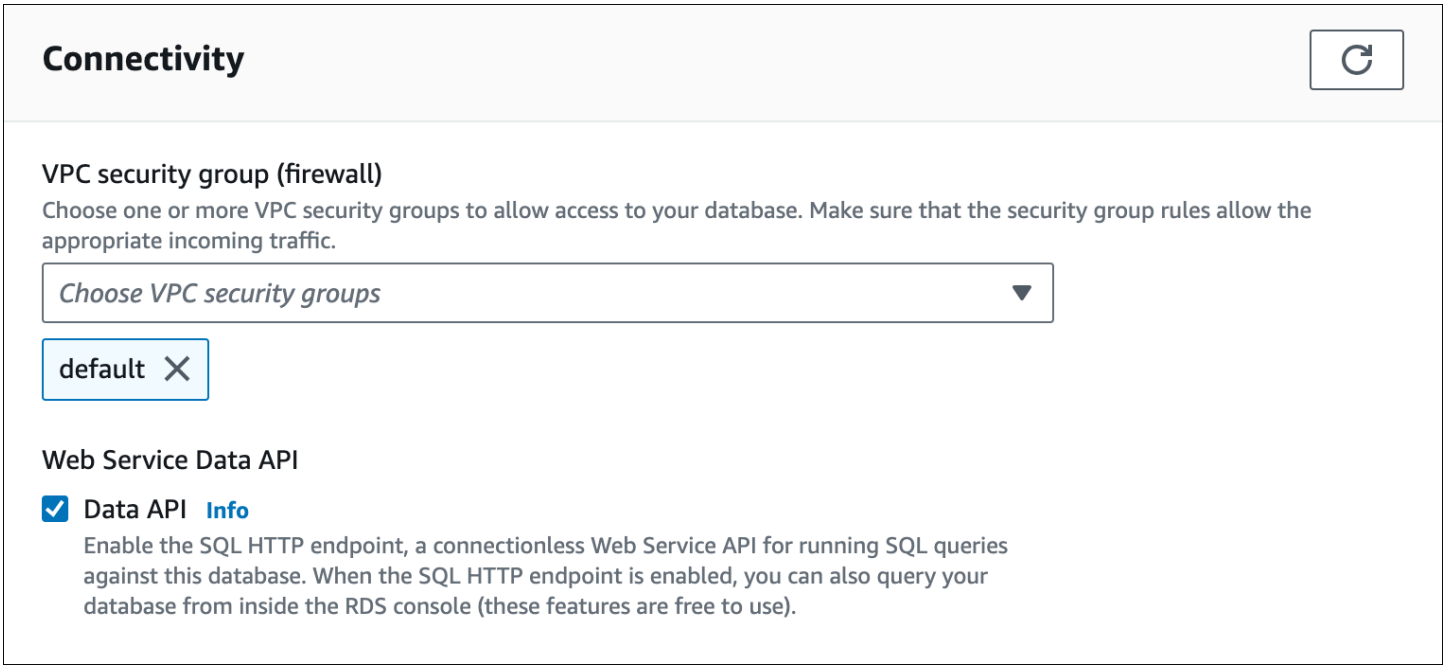
Activation ou désactivation de l'API de données (Aurora Serverless v1 uniquement)

Utilisez les procédures suivantes pour activer ou désactiver l'API de données sur les bases de données Aurora Serverless v1 existantes. Utilisez les procédures indiquées dans the section called "Activation ou désactivation de l'API de données" pour activer ou désactiver l'API de données sur Aurora Serverless v2 et les bases de données provisionnées.

Console

Lorsque vous créez ou modifiez un cluster de bases de données Aurora Serverless v1, vous pouvez activez l'API de données dans la section Connectivité de la console RDS.

Dans la capture d'écran ci-dessous, l'API de données est activée lors de la modification d'un cluster de bases de données Aurora.



Pour obtenir des instructions sur la modification d'un cluster de bases de données Aurora Serverless v1, consultez Modification d'un cluster de bases de données Aurora Serverless v1.

AWS CLI

Pour activer ou désactiver l'API de données, exécutez la commande AWS CLI modify-db-cluster, avec `--enable-http-endpoint` ou `--no-enable-http-endpoint`, selon le cas.

Dans l'exemple suivant, l'API de données est activée sur `sample-cluster`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier sample-cluster \
    --enable-http-endpoint
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier sample-cluster ^
    --enable-http-endpoint
```

API RDS

Pour activer l'API de données, utilisez l'opération ModifyDBCluster et définissez la valeur de `EnableHttpEndpoint` sur `true` ou `false`, selon le cas.

Création d'un point de terminaison de VPC Amazon pour l'API de données Amazon RDS (AWS PrivateLink)

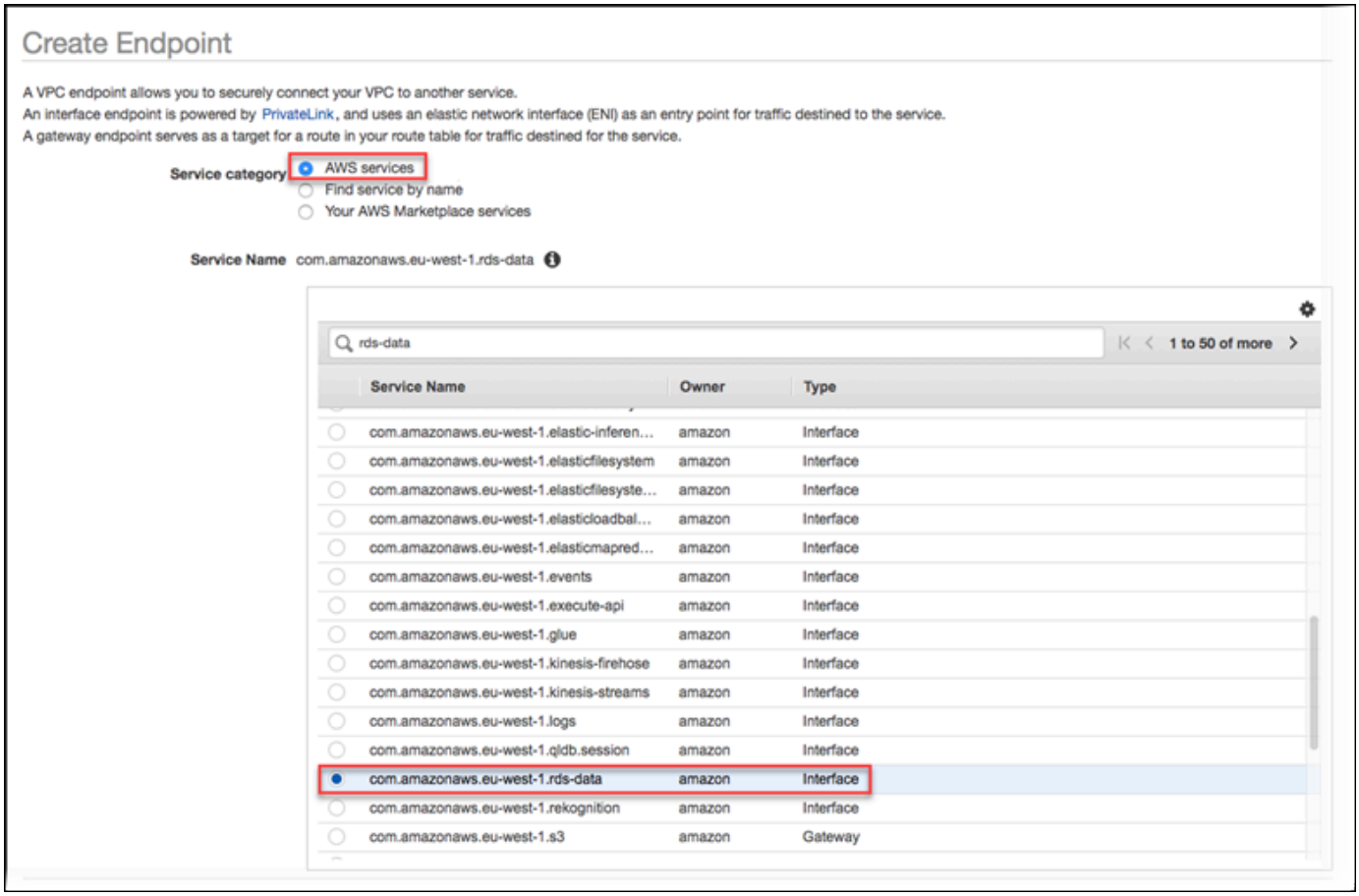
Amazon VPC vous permet de lancer des ressources AWS, telles que des clusters de base de données et des applications Aurora, dans un cloud privé virtuel (VPC). AWS PrivateLink fournit une connectivité privée entre les VPC et les services AWS en toute sécurité sur le réseau Amazon. AWS PrivateLink vous permet de créer des points de terminaison Amazon VPC grâce auxquels vous pouvez vous connecter à des services sur différents comptes et VPC basés sur Amazon VPC. Pour plus d'informations sur AWS PrivateLink, consultez Services de points de terminaison de VPC (AWS PrivateLink) dans le Guide de l'utilisateur Amazon Virtual Private Cloud.

Vous pouvez appeler l'API de données RDS (API de données) avec des points de terminaison Amazon VPC. L'utilisation d'un point de terminaison Amazon VPC permet de maintenir le trafic entre les applications de votre Amazon VPC et l'API de données dans le réseau AWS, sans utiliser d'adresses IP publiques. Les points de terminaison Amazon VPC peuvent vous aider à respecter les exigences réglementaires et de conformité liées à la limitation de la connectivité Internet publique. Par exemple, si vous utilisez un point de terminaison Amazon VPC, vous pouvez maintenir le trafic entre une application exécutée sur une instance Amazon EC2 et l'API de données dans les VPC qui les hébergent.

Une fois que vous avez créé le point de terminaison Amazon VPC, vous pouvez commencer à l'utiliser sans modifier le code ou la configuration de votre application.

Pour créer un point de terminaison Amazon VPC pour l'API de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Choisissez Points de terminaison, puis Créer un point de terminaison.
3. Sur la page Créer un point de terminaison, pour Catégorie de services, choisissez Services AWS. Pour Nom du service, choisissez rds-data.



4. Pour VPC, choisissez le VPC dans lequel créer le point de terminaison.

   Choisissez le VPC contenant l'application qui effectue des appels de l'API de données.

5. Pour Subnets (Sous-réseaux), choisissez le sous-réseau de chaque zone de disponibilité (AZ) utilisée par le service AWS qui exécute votre application.

   Pour créer un point de terminaison Amazon VPC, spécifiez la plage d'adresses IP privées dans laquelle le point de terminaison sera accessible. Pour ce faire, choisissez le sous-réseau de chaque zone de disponibilité. Cela limite le point de terminaison de VPC à la plage d'adresses

IP privées spécifique à chaque zone de disponibilité et crée également un point de terminaison Amazon VPC dans chaque zone de disponibilité.

6. Pour Enable DNS Name (Activer le nom DNS), sélectionnez Activer pour ce point de terminaison.

   Private DNS résout le nom d'hôte DNS standard de l'API de données (`https://rds-data.`*`region`*`.amazonaws.com`) par les adresses IP privées associées au nom d'hôte DNS spécifique à votre point de terminaison Amazon VPC. Par conséquent, vous pouvez accéder au point de terminaison de VPC de l'API de données à l'aide de l'AWS CLI ou des kits AWS SDK sans modifier le code ou la configuration pour mettre à jour l'URL du point de terminaison de l'API de données.

7. Pour Groupe de sécurité, choisissez un groupe de sécurité à associer au point de terminaison Amazon VPC.

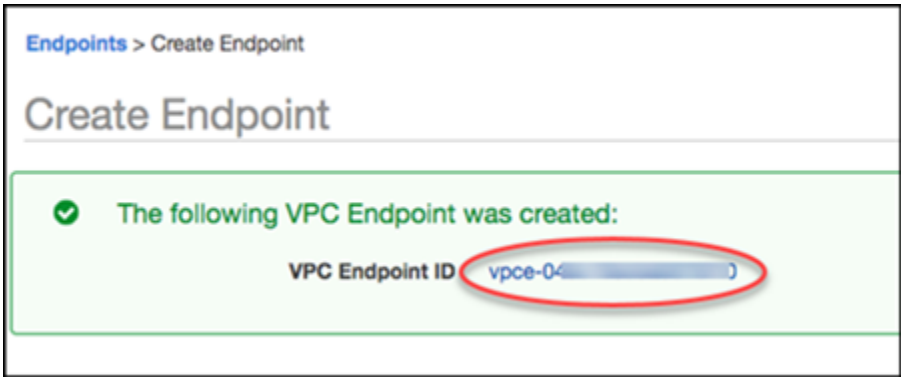
   Choisissez le groupe de sécurité qui autorise l'accès au service AWS qui exécute votre application. Par exemple, si une instance Amazon EC2 exécute votre application, choisissez le groupe de sécurité qui autorise l'accès à cette instance Amazon EC2. Le groupe de sécurité vous permet de contrôler le trafic vers le point de terminaison Amazon VPC à partir des ressources de votre VPC.

8. Pour Stratégie, choisissez Accès complet pour permettre à toute personne à l'intérieur de l'Amazon VPC d'accéder à l'API de données via ce point de terminaison. Ou choisissez Personnalisé pour spécifier une stratégie qui limite l'accès.

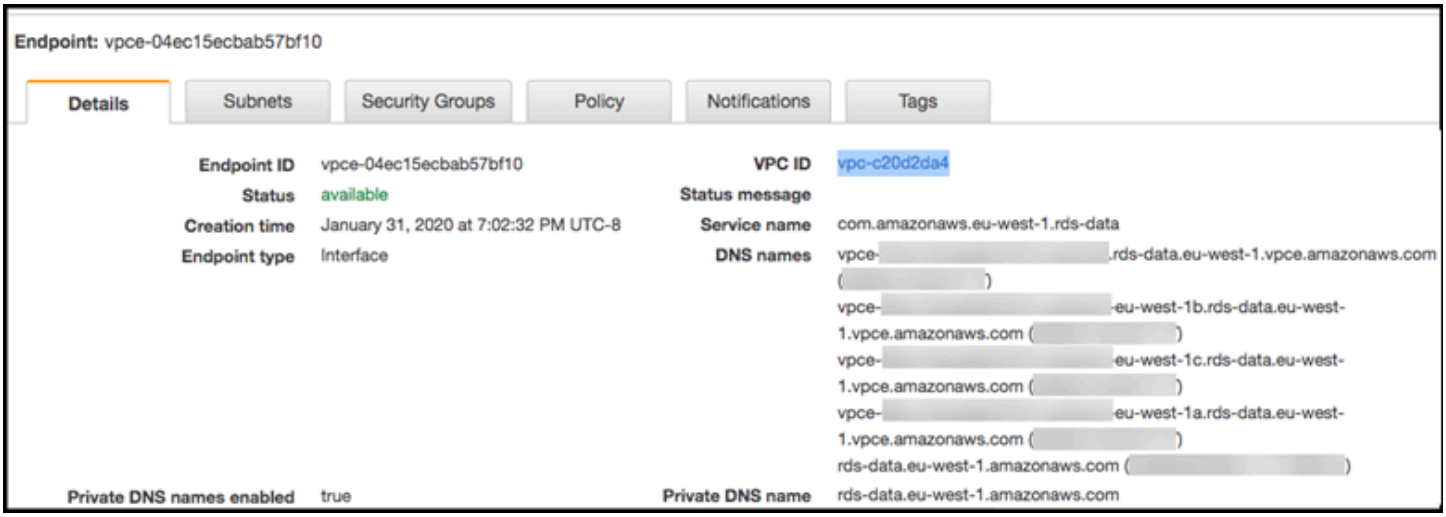
   Si vous choisissez Personnalisé, entrez la stratégie dans l'outil de création de stratégie.

9. Choisissez Créer un point de terminaison.

Une fois le point de terminaison créé, choisissez le lien dans l'AWS Management Console pour afficher les détails du point de terminaison.



L'onglet Détails du point de terminaison affiche les noms d'hôte DNS générés lors de la création du point de terminaison Amazon VPC.



Vous pouvez utiliser le point de terminaison standard (`rds-data.`*`region`*`.amazonaws.com`) ou l'un des points de terminaison spécifiques au VPC pour appeler l'API de données dans l'Amazon VPC. Le point de terminaison standard de l'API de données effectue automatiquement un routage vers le point de terminaison Amazon VPC. Ce routage se produit car le nom d'hôte DNS privé a été activé lors de la création du point de terminaison Amazon VPC.

Lorsque vous utilisez un point de terminaison Amazon VPC dans un appel de l'API de données, tout le trafic entre votre application et l'API de données reste dans les Amazon VPC qui les contiennent. Vous pouvez utiliser un point de terminaison Amazon VPC pour n'importe quel type d'appel à l'API de données. Pour plus d'informations sur l'appel de l'API de données, consultez Appel de l'API de données Amazon RDS.

Appel de l’API de données Amazon RDS

Quand l’API de données Amazon RDS (API de données) est activée sur votre cluster de bases de données Aurora, vous pouvez exécuter des instructions SQL sur le cluster de bases de données Aurora à l’aide de l’API de données ou de l’AWS CLI. L’API de données est compatible avec les langages de programmation pris en charge par les kits AWS SDK. Pour plus d’informations, consultez Outils pour créer sur AWS.

Rubriques

Référence des opérations de l’API de données Amazon RDS

L’API de données Amazon RDS fournit les opérations suivantes pour exécuter les instructions SQL.

Opération d’API de données	AWS CLI commande	Description
ExecuteStatement	aws rds-data execute-statement	Exécute une instruction SQL sur une base de données.
BatchExecuteStatement	aws rds-data batch-execute-statement	Exécute une instruction SQL par lots sur un tableau de données pour les opérations d’insertion et de mise à jour en bloc. Vous pouvez exécuter une instruction en langage de manipulation de données (DML) avec un tableau de jeux de paramètres. Une instruction SQL par lots peut nettement améliorer les performances sur des instructions d’insertion et de mise à jour.

Vous pouvez utiliser l'une ou l'autre des opérations pour exécuter des instructions SQL individuelles ou des transactions. Pour les transactions, l'API de données fournit les opérations suivantes.

Opération d'API de données	AWS CLI commande	Description
BeginTransaction	aws rds-data begin-transaction	Démarre une transaction SQL.
CommitTransaction	aws rds-data commit-transaction	Termine une transaction SQL et valide les modifications.
RollbackTransaction	aws rds-data rollback-transaction	Restaure une transaction.

Les opérations pour effectuer les instructions SQL et pour la prise en charge des transactions comportent les mêmes options AWS CLI et paramètres d'API de données indiqués ci-après. Certaines opérations prennent en charge d'autres paramètres et options.

Paramètre d'opération d'API de données	AWS CLIOption de la commande	Obligatoire	Description
`resourceArn`	`--resource-arn`	Oui	Amazon Resource Name (ARN) du cluster de bases de données Aurora. Le cluster doit être dans le même Compte AWS que le rôle IAM ou l'utilisateur IAM qui invoque l'API de données. Pour accéder à un cluster dans un autre compte, vous devez endosser un rôle dans ce compte.
`secretArn`	`--secret-arn`	Oui	Nom ou ARN du secret qui active l'accès au cluster de bases de données.

L’API de données RDS prend en charge les types de données suivants pour Aurora MySQL :

- `TINYINT(1)`, `BOOLEAN`, `BOOL`
- `TINYINT`
- `SMALLINT` [`SIGNED` | `UNSIGNED`]
- `MEDIUMINT` [`SIGNED` | `UNSIGNED`]
- `INT` [`SIGNED` | `UNSIGNED`]
- `BIGINT` [`SIGNED` | `UNSIGNED`]
- `FLOAT`
- `DOUBLE`
- `VARCHAR`, `CHAR`, `TEXT`, `ENUM`
- `VARBINARY`, `BINARY`, `BLOB`
- `DATE`, `TIME`, `DATETIME`, `TIMESTAMP`
- `DECIMAL`
- `JSON`
- `BIT`, `BIT(N)`

L’API de données RDS prend en charge les types scalaires Aurora PostgreSQL suivants :

- `BOOL`
- `BYTEA`
- `DATE`
- `CIDR`
- `DECIMAL`, `NUMERIC`
- `ENUM`
- `FLOAT8`, `DOUBLE PRECISION`
- `INET`
- `INT`, `INT4`, `SERIAL`
- `INT2`, `SMALLINT`, `SMALLSERIAL`
- `INT8`, `BIGINT`, `BIGSERIAL`
- `JSONB`, `JSON`
- `REAL`, `FLOAT`

- TEXT, CHAR(N), VARCHAR, NAME
- TIME
- TIMESTAMP
- UUID
- VECTOR

L'API de données RDS prend en charge les types de tableaux Aurora PostgreSQL suivants :

- BOOL[], BIT[]
- DATE[]
- DECIMAL[], NUMERIC[]
- FLOAT8[], DOUBLE PRECISION[]
- INT[], INT4[]
- INT2[]
- INT8[], BIGINT[]
- JSON[]
- REAL[], FLOAT[]
- TEXT[], CHAR(N)[], VARCHAR[], NAME[]
- TIME[]
- TIMESTAMP[]
- UUID[]

Vous pouvez utiliser les paramètres dans les appels d'API de données pour `ExecuteStatement` et `BatchExecuteStatement`, et lorsque vous exécutez les commandes AWS CLI `execute-statement` et `batch-execute-statement`. Pour utiliser un paramètre, spécifiez une paire nom-valeur dans le type de données `SqlParameter`. Vous devez spécifier la valeur avec le type de données `Field`. Le tableau suivant associe les types de données Java Database Connectivity (JDBC) aux types de données que vous spécifiez dans les appels d'API de données.

Type de données JDBC	Type de données API de données
INTEGER, TINYINT, SMALLINT, BIGINT	LONG (ou STRING)

Type de données JDBC	Type de données API de données
`FLOAT, REAL, DOUBLE`	`DOUBLE`
`DECIMAL`	`STRING`
`BOOLEAN, BIT`	`BOOLEAN`
`BLOB, BINARY, LONGVARBINARY, VARBINARY`	`BLOB`
`CLOB`	`STRING`
Autres types (y compris les types liés à la date et à l'heure)	`STRING`

> **Note**
>
> Vous pouvez spécifier le type de données `LONG` ou `STRING` dans votre appel d'API de données pour les valeurs `LONG` renvoyées par la base de données. Nous vous recommandons de le faire pour éviter de perdre en précision pour les très grands nombres, ce qui peut se produire lorsque vous travaillez avec JavaScript.

Certains types, tels que `DECIMAL` et `TIME`, nécessitent un indice pour que l'API de données transmette les valeurs `String` à la base de données en tant que type correct. Pour utiliser un indice, incluez des valeurs pour `typeHint` dans le type de données `SqlParameter`. Les valeurs possibles pour `typeHint` sont les suivantes :

- `DATE` – La valeur de paramètre `String` correspondante est envoyée en tant qu'objet de type `DATE` à la base de données. Le format accepté est `YYYY-MM-DD`.
- `DECIMAL` – La valeur de paramètre `String` correspondante est envoyée en tant qu'objet de type `DECIMAL` à la base de données.
- `JSON` – La valeur de paramètre `String` correspondante est envoyée en tant qu'objet de type `JSON` à la base de données.
- `TIME` – La valeur de paramètre `String` correspondante est envoyée en tant qu'objet de type `TIME` à la base de données. Le format accepté est `HH:MM:SS[.FFF]`.

- `TIMESTAMP` – La valeur de paramètre `String` correspondante est envoyée en tant qu'objet de type `TIMESTAMP` à la base de données. Le format accepté est `YYYY-MM-DD HH:MM:SS[.FFF]`.
- `UUID` – La valeur de paramètre `String` correspondante est envoyée en tant qu'objet de type `UUID` à la base de données.

> **Note**
>
> Actuellement, l'API de données ne prend pas en charge les tableaux d'identifiants uniques universels (UUID).

> **Note**
>
> Pour Amazon Aurora PostgreSQL, l'API de données renvoie toujours le type de données Aurora PostgreSQL `TIMESTAMPTZ` dans le fuseau horaire UTC.

Appel de l'API de données Amazon RDS à l'aide de l'AWS CLI

Vous pouvez appeler l'API de données RDS (API de donnée) à l'aide d'AWS CLI.

Les exemples suivants utilisent AWS CLI pour l'API de données. Pour plus d'informations, consultez le document de référence AWS CLI pour l'API de données.

Dans chaque exemple, remplacez l'Amazon Resource Name (ARN) du cluster de bases de données par l'ARN de votre cluster de bases de données Aurora. De même, remplacez l'ARN du secret par l'ARN du secret dans Secrets Manager qui autorise l'accès au cluster de bases de données.

> **Note**
>
> AWS CLI peut formater les réponses en JSON.

Rubriques

- Démarrage d'une transaction SQL
- Exécution d'une instruction SQL
- Exécution d'une instruction SQL par lots sur un tableau de données
- Validation d'une transaction SQL

- Restauration d'une transaction

Démarrage d'une transaction SQL

Vous pouvez démarrer une transaction SQL à l'aide de la commande CLI `aws rds-data begin-transaction`. L'appel renvoie un identifiant de transaction.

> **Important**
>
> Dans l'API de données, une transaction expire si aucun appel n'utilise son identifiant de transaction dans un délai de trois minutes. Si une transaction expire avant d'être validée, l'API de données la restaure automatiquement.
> Les instructions DDL (Data Definition Language) MySQL d'une transaction provoquent une validation implicite. Nous vous recommandons d'exécuter chaque instruction DDL MySQL dans une commande `execute-statement` séparée avec l'option `--continue-after-timeout`.

En plus des options communes, spécifiez l'option `--database` qui indique le nom de la base de données.

Par exemple, la commande CLI suivante démarre une transaction SQL.

Pour Linux, macOS ou Unix :

```
aws rds-data begin-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret"
```

Pour Windows :

```
aws rds-data begin-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret"
```

Voici un exemple de réponse.

```
{
```

```
    "transactionId": "ABC1234567890xyz"
}
```

Exécution d'une instruction SQL

Vous pouvez exécuter une instruction SQL à l'aide de la commande CLI `aws rds-data execute-statement`.

Vous pouvez exécuter une instruction SQL à l'intérieur d'une transaction en spécifiant l'identifiant de la transaction avec l'option `--transaction-id`. Vous pouvez démarrer une transaction à l'aide de la commande CLI `aws rds-data begin-transaction`. Vous pouvez terminer et valider une transaction à l'aide de la commande CLI `aws rds-data commit-transaction`.

> **Important**
>
> Si vous ne spécifiez pas l'option `--transaction-id`, les modifications renvoyées par l'appel sont automatiquement validées.

En plus des options courantes, spécifiez les options suivantes :

- `--sql` (obligatoire) – Instruction SQL à exécuter sur le cluster de bases de données.
- `--transaction-id` (facultatif) – Identifiant d'une transaction démarrée à l'aide de la commande CLI `begin-transaction`. Spécifiez l'identifiant de la transaction dans laquelle vous souhaitez intégrer l'instruction SQL.
- `--parameters` (facultatif) – Paramètres pour l'instruction SQL.
- `--include-result-metadata | --no-include-result-metadata` (facultatif) – Valeur indiquant si les métadonnées doivent apparaître dans les résultats. La valeur par défaut est `--no-include-result-metadata`.
- `--database` (facultatif) – Nom de la base de données.

  L'option `--database` peut ne pas fonctionner lorsque vous exécutez une instruction SQL après avoir exécuté `--sql "use database_name;"` dans la demande précédente. Nous vous recommandons d'utiliser l'option `--database` plutôt que d'exécuter des instructions `--sql "use database_name;"`.
- `--continue-after-timeout | --no-continue-after-timeout` (facultatif) – Valeur indiquant si l'exécution de l'instruction doit se poursuivre lorsque l'appel dépasse le délai

d'expiration de 45 secondes de l'API de données. La valeur par défaut est `--no-continue-after-timeout`.

Pour les instructions en langage de définition de données (DDL), nous vous recommandons de continuer à exécuter l'instruction après l'expiration de l'appel afin d'éviter les erreurs et la corruption de structures de données.

- `--format-records-as "JSON"|"NONE"` : une valeur facultative qui spécifie si l'ensemble de résultats doit être formaté en tant que chaîne JSON. La valeur par défaut est `"NONE"`. Pour obtenir des informations sur l'utilisation du traitement des ensembles de résultats JSON, consultez Traitement des requêtes d'API Amazon RDS Data au format JSON.

Le cluster de bases de données renvoie une réponse pour l'appel.

> **Note**
>
> La taille de réponse est limitée à 1 Mio. Si l'appel renvoie plus de 1 Mio de données de réponse, l'appel est arrêté.
> Le nombre maximal de demandes par seconde est 1 000 pour Aurora Serverless v1. Aucune limite n'est imposée pour les autres bases de données prises en charge.

Par exemple, la commande CLI suivante exécute une instruction SQL unique et omet les métadonnées dans les résultats (par défaut).

Pour Linux, macOS ou Unix :

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "select * from mytable"
```

Pour Windows :

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
```

```
--sql "select * from mytable"
```

Voici un exemple de réponse.

```
{
    "numberOfRecordsUpdated": 0,
    "records": [
        [
            {
                "longValue": 1
            },
            {
                "stringValue": "ValueOne"
            }
        ],
        [
            {
                "longValue": 2
            },
            {
                "stringValue": "ValueTwo"
            }
        ],
        [
            {
                "longValue": 3
            },
            {
                "stringValue": "ValueThree"
            }
        ]
    ]
}
```

La commande CLI suivante exécute une instruction SQL unique dans une transaction en spécifiant l'option `--transaction-id`.

Pour Linux, macOS ou Unix :

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
```

```
--sql "update mytable set quantity=5 where id=201" --transaction-id "ABC1234567890xyz"
```

Pour Windows :

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "update mytable set quantity=5 where id=201" --transaction-id "ABC1234567890xyz"
```

Voici un exemple de réponse.

```
{
    "numberOfRecordsUpdated": 1
}
```

La commande CLI suivante exécute une seule instruction SQL avec des paramètres.

Pour Linux, macOS ou Unix :

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "insert into mytable values (:id, :val)" --parameters "[{\"name\": \"id\",
 \"value\": {\"longValue\": 1}},{\"name\": \"val\", \"value\": {\"stringValue\":
 \"value1\"}}]"
```

Pour Windows :

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "insert into mytable values (:id, :val)" --parameters "[{\"name\": \"id\",
 \"value\": {\"longValue\": 1}},{\"name\": \"val\", \"value\": {\"stringValue\":
 \"value1\"}}]"
```

Voici un exemple de réponse.

```
{
```

```
    "numberOfRecordsUpdated": 1
}
```

La commande CLI suivante exécute une instruction SQL en langage de définition de données (DDL). L'instruction DDL renomme la colonne `job` en colonne `role`.

> **Important**
>
> Pour les instructions DDL, nous vous recommandons de continuer à exécuter l'instruction une fois l'appel expiré. Lorsqu'une instruction DDL se termine avant la fin de son exécution, cela peut entraîner des erreurs et corrompre les structures de données. Pour maintenir l'exécution d'une instruction lorsqu'un appel dépasse le délai d'expiration de 45 secondes de l'API de données RDS, sélectionnez l'option `--continue-after-timeout`.

Pour Linux, macOS ou Unix :

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "alter table mytable change column job role varchar(100)" --continue-after-
timeout
```

Pour Windows :

```
aws rds-data execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "alter table mytable change column job role varchar(100)" --continue-after-
timeout
```

Voici un exemple de réponse.

```
{
    "generatedFields": [],
    "numberOfRecordsUpdated": 0
}
```

> **Note**
>
> Les données `generatedFields` ne sont pas prises en charge par Aurora PostgreSQL. Pour obtenir les valeurs des champs générés, utilisez la clause `RETURNING`. Pour plus d'informations, consultez Renvoi de données de lignes modifiées dans la documentation PostgreSQL.

Exécution d'une instruction SQL par lots sur un tableau de données

Vous pouvez exécuter une instruction SQL par lots sur un tableau de données à l'aide de la commande CLI `aws rds-data batch-execute-statement`. Vous pouvez utiliser cette commande pour réaliser une opération d'importation ou de mise à jour en bloc.

Vous pouvez exécuter une instruction SQL à l'intérieur d'une transaction en spécifiant l'identifiant de la transaction avec l'option `--transaction-id`. Vous pouvez démarrer une transaction à l'aide de la commande CLI `aws rds-data begin-transaction`. Vous pouvez terminer et valider une transaction à l'aide de la commande CLI `aws rds-data commit-transaction`.

> **Important**
>
> Si vous ne spécifiez pas l'option `--transaction-id`, les modifications renvoyées par l'appel sont automatiquement validées.

En plus des options courantes, spécifiez les options suivantes :

- `--sql` (obligatoire) – Instruction SQL à exécuter sur le cluster de bases de données.

  > **Tip**
  >
  > Pour les instructions compatibles avec MySQL, n'incluez pas de point-virgule à la fin du paramètre `--sql`. Un point-virgule final peut entraîner une erreur de syntaxe.

- `--transaction-id` (facultatif) – Identifiant d'une transaction démarrée à l'aide de la commande CLI `begin-transaction`. Spécifiez l'identifiant de la transaction dans laquelle vous souhaitez intégrer l'instruction SQL.
- `--parameter-set` (facultatif) – Ensembles de paramètres pour l'opération par lots.

- `--database` (facultatif) – Nom de la base de données.

Le cluster de bases de données renvoie une réponse à l'appel.

> **Note**
>
> Il n'existe pas de limite supérieure fixe pour le nombre d'ensembles de paramètres. Toutefois, la taille maximale de la demande HTTP envoyée via l'API de données est de 4 MiB. Si la demande dépasse cette limite, l'API de données renvoie une erreur et ne traite pas la demande. Cette limite de 4 MiB inclut la taille des en-têtes HTTP et la notation JSON dans la demande. Ainsi, le nombre d'ensembles de paramètres que vous pouvez inclure dépend d'une combinaison de facteurs, tels que la taille de l'instruction SQL et la taille de chaque ensemble de paramètres.
>
> La taille de réponse est limitée à 1 Mio. Si l'appel renvoie plus de 1 Mio de données de réponse, l'appel est arrêté.
>
> Le nombre maximal de demandes par seconde est 1 000 pour Aurora Serverless v1. Aucune limite n'est imposée pour les autres bases de données prises en charge.

Par exemple, la commande CLI suivante exécute une instruction SQL par lots sur un tableau de données avec un ensemble de paramètres.

Pour Linux, macOS ou Unix :

```
aws rds-data batch-execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" \
--sql "insert into mytable values (:id, :val)" \
--parameter-sets "[[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"ValueOne\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 2}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueTwo\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 3}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueThree\"}}]]"
```

Pour Windows :

```
aws rds-data batch-execute-statement --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
```

```
--database "mydb" --secret-arn "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret" ^
--sql "insert into mytable values (:id, :val)" ^
--parameter-sets "[[{\"name\": \"id\", \"value\": {\"longValue\": 1}},{\"name\":
 \"val\", \"value\": {\"stringValue\": \"ValueOne\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 2}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueTwo\"}}],
[{\"name\": \"id\", \"value\": {\"longValue\": 3}},{\"name\": \"val\", \"value\":
 {\"stringValue\": \"ValueThree\"}}]]"
```

> **Note**
>
> N'incluez pas les sauts de ligne présents dans l'option `--parameter-sets`.

Validation d'une transaction SQL

À l'aide de la commande CLI `aws rds-data commit-transaction`, vous pouvez terminer une transaction SQL que vous avez démarrée avec `aws rds-data begin-transaction` et valider les modifications.

En plus des options courantes, spécifiez l'option suivante :

- `--transaction-id` (obligatoire) – Identifiant d'une transaction démarrée à l'aide de la commande CLI `begin-transaction`. Spécifiez l'identifiant de la transaction que vous souhaitez terminer et valider.

Par exemple, la commande CLI suivante termine une transaction SQL et valide les changements.

Pour Linux, macOS ou Unix :

```
aws rds-data commit-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--transaction-id "ABC1234567890xyz"
```

Pour Windows :

```
aws rds-data commit-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" ^
```

```
--transaction-id "ABC1234567890xyz"
```

Voici un exemple de réponse.

```
{
    "transactionStatus": "Transaction Committed"
}
```

Restauration d'une transaction

À l'aide de la commande CLI `aws rds-data rollback-transaction`, vous pouvez restaurer une transaction SQL que vous avez démarrée avec `aws rds-data begin-transaction`. La restauration d'une transaction annule les changements apportés.

> **Important**
>
> L'expiration de l'identifiant de la transaction entraîne automatiquement sa restauration. Dans ce cas, une commande `aws rds-data rollback-transaction` qui spécifie l'identifiant de transaction expiré renvoie une erreur.

En plus des options courantes, spécifiez l'option suivante :

- `--transaction-id` (obligatoire) – Identifiant d'une transaction démarrée à l'aide de la commande CLI `begin-transaction`. Spécifiez l'identifiant de la transaction que vous souhaitez restaurer.

Par exemple, la commande AWS CLI suivante restaure une transaction SQL.

Pour Linux, macOS ou Unix :

```
aws rds-data rollback-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" \
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--transaction-id "ABC1234567890xyz"
```

Pour Windows :

```
aws rds-data rollback-transaction --resource-arn "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster" ^
```

```
--secret-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" ^
--transaction-id "ABC1234567890xyz"
```

Voici un exemple de réponse.

```
{
    "transactionStatus": "Rollback Complete"
    }
```

Appel à l'API de données Amazon RDS depuis une application Python

Vous pouvez appeler l'API de données Amazon RDS (API de données) depuis une application Python.

Les exemples suivants utilisent le kit AWS SDK pour Python (Boto). Pour plus d'informations sur Boto, consultez la documentation AWS SDK pour Python (Boto 3).

Dans chaque exemple, remplacez l'Amazon Resource Name (ARN) du cluster de bases de données par l'ARN de votre cluster de bases de données Aurora. De même, remplacez l'ARN du secret par l'ARN du secret dans Secrets Manager qui autorise l'accès au cluster de bases de données.

Rubriques

- Exécution d'une requête SQL
- Exécution d'une instruction SQL DML
- Exécution d'une transaction SQL

Exécution d'une requête SQL

Vous pouvez exécuter une instruction `SELECT` puis extraire les résultats avec une application Python.

L'exemple suivant exécute une requête SQL.

```
import boto3

rdsData = boto3.client('rds-data')

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

response1 = rdsData.execute_statement(
```

```
            resourceArn = cluster_arn,
            secretArn = secret_arn,
            database = 'mydb',
            sql = 'select * from employees limit 3')

print (response1['records'])
[
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'ROSALEZ'
        },
        {
            'stringValue': 'ALEJANDRO'
        },
        {
            'stringValue': '2016-02-15 04:34:33.0'
        }
    ],
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'DOE'
        },
        {
            'stringValue': 'JANE'
        },
        {
            'stringValue': '2014-05-09 04:34:33.0'
        }
    ],
    [
        {
            'longValue': 1
        },
        {
            'stringValue': 'STILES'
        },
        {
            'stringValue': 'JOHN'
```

```
        },
        {
            'stringValue': '2017-09-20 04:34:33.0'
        }
    ]
]
```

Exécution d'une instruction SQL DML

Vous pouvez exécuter une instruction en langage de manipulation de données (DML) pour intégrer, mettre à jour ou supprimer des données dans votre base de données. Vous pouvez également utiliser des paramètres dans les instructions DML.

> **Important**
>
> Si un appel ne fait pas partie d'une transaction, car il ne comprend pas le paramètre `transactionID`, les modifications résultant de l'appel sont validées automatiquement.

L'exemple suivant exécute une instruction SQL INSERT et utilise des paramètres.

```
import boto3

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

rdsData = boto3.client('rds-data')


param1 = {'name':'firstname', 'value':{'stringValue': 'JACKSON'}}
param2 = {'name':'lastname', 'value':{'stringValue': 'MATEO'}}
paramSet = [param1, param2]

response2 = rdsData.execute_statement(resourceArn=cluster_arn,
                                      secretArn=secret_arn,
                                      database='mydb',
                                      sql='insert into employees(first_name, last_name)
 VALUES(:firstname, :lastname)',
                                      parameters = paramSet)

print (response2["numberOfRecordsUpdated"])
```

Exécution d’une transaction SQL

Vous pouvez démarrer une transaction SQL, exécuter une ou plusieurs instructions SQL, puis valider les modifications avec une application Python.

> **Important**
>
> Une transaction expire si aucun appel n’utilise son identifiant de transaction dans un délai de trois minutes. Si une transaction expire avant d’être validée, elle est automatiquement restaurée.
>
> Si vous ne spécifiez pas d’identifiant de transaction, les modifications résultant de l’appel sont validées automatiquement.

L’exemple suivant exécute une transaction SQL qui insère une ligne dans un tableau.

```
import boto3

rdsData = boto3.client('rds-data')

cluster_arn = 'arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster'
secret_arn = 'arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret'

tr = rdsData.begin_transaction(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     database = 'mydb')

response3 = rdsData.execute_statement(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     database = 'mydb',
     sql = 'insert into employees(first_name, last_name) values('XIULAN', 'WANG')',
     transactionId = tr['transactionId'])

cr = rdsData.commit_transaction(
     resourceArn = cluster_arn,
     secretArn = secret_arn,
     transactionId = tr['transactionId'])

cr['transactionStatus']
'Transaction Committed'
```

```
response3['numberOfRecordsUpdated']
1
```

> **Note**
>
> Si vous exécutez une instruction en langage de définition de données (DDL), nous vous recommandons de continuer à exécuter l'instruction une fois l'appel expiré. Lorsqu'une instruction DDL se termine avant la fin de son exécution, cela peut entraîner des erreurs et corrompre les structures de données. Pour maintenir l'exécution d'une instruction lorsqu'un appel dépasse le délai d'expiration de 45 secondes de l'API de données RDS, définissez le paramètre `continueAfterTimeout` sur `true`.

Appel à l'API de données Amazon RDS depuis une application Java

Vous pouvez appeler l'API de données Amazon RDS (API de données) depuis une application Java.

Les exemples suivants utilisent le kit AWS SDK pour Java. Pour plus d'informations, consultez le Manuel du développeur AWS SDK pour Java.

Dans chaque exemple, remplacez l'Amazon Resource Name (ARN) du cluster de bases de données par l'ARN de votre cluster de bases de données Aurora. De même, remplacez l'ARN du secret par l'ARN du secret dans Secrets Manager qui autorise l'accès au cluster de bases de données.

Rubriques

- Exécution d'une requête SQL
- Exécution d'une transaction SQL
- Exécution d'une opération SQL par lots

Exécution d'une requête SQL

Vous pouvez exécuter une instruction `SELECT`? puis extraire les résultats avec une application Java.

L'exemple suivant exécute une requête SQL.

```
package com.amazonaws.rdsdata.examples;
```

```
import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.ExecuteStatementRequest;
import com.amazonaws.services.rdsdata.model.ExecuteStatementResult;
import com.amazonaws.services.rdsdata.model.Field;

import java.util.List;

public class FetchResultsExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
    AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    ExecuteStatementRequest request = new ExecuteStatementRequest()
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withDatabase("mydb")
            .withSql("select * from mytable");

    ExecuteStatementResult result = rdsData.executeStatement(request);

    for (List<Field> fields: result.getRecords()) {
      String stringValue = fields.get(0).getStringValue();
      long numberValue = fields.get(1).getLongValue();

      System.out.println(String.format("Fetched row: string = %s, number = %d",
 stringValue, numberValue));
    }
  }
}
```

Exécution d’une transaction SQL

Vous pouvez démarrer une transaction SQL, exécuter une ou plusieurs instructions SQL, puis valider les modifications avec une application Java.

> **Important**
>
> Une transaction expire si aucun appel n'utilise son identifiant de transaction dans un délai de trois minutes. Si une transaction expire avant d'être validée, elle est automatiquement restaurée.
>
> Si vous ne spécifiez pas d'identifiant de transaction, les modifications résultant de l'appel sont validées automatiquement.

L'exemple suivant exécute une transaction SQL.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.BeginTransactionRequest;
import com.amazonaws.services.rdsdata.model.BeginTransactionResult;
import com.amazonaws.services.rdsdata.model.CommitTransactionRequest;
import com.amazonaws.services.rdsdata.model.ExecuteStatementRequest;

public class TransactionExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
    AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    BeginTransactionRequest beginTransactionRequest = new BeginTransactionRequest()
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withDatabase("mydb");
    BeginTransactionResult beginTransactionResult =
 rdsData.beginTransaction(beginTransactionRequest);
    String transactionId = beginTransactionResult.getTransactionId();

    ExecuteStatementRequest executeStatementRequest = new ExecuteStatementRequest()
            .withTransactionId(transactionId)
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withSql("INSERT INTO test_table VALUES ('hello world!')");
```

```
    rdsData.executeStatement(executeStatementRequest);

    CommitTransactionRequest commitTransactionRequest = new CommitTransactionRequest()
            .withTransactionId(transactionId)
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN);
    rdsData.commitTransaction(commitTransactionRequest);
  }
}
```

> **Note**
>
> Si vous exécutez une instruction en langage de définition de données (DDL), nous vous recommandons de continuer à exécuter l'instruction une fois l'appel expiré. Lorsqu'une instruction DDL se termine avant la fin de son exécution, cela peut entraîner des erreurs et corrompre les structures de données. Pour maintenir l'exécution d'une instruction lorsqu'un appel dépasse le délai d'expiration de 45 secondes de l'API de données RDS, définissez le paramètre `continueAfterTimeout` sur `true`.

Exécution d'une opération SQL par lots

Vous pouvez exécuter des opérations d'insertion et de mise à jour en bloc sur un tableau de données avec une application Java. Vous pouvez exécuter une instruction DML avec des groupes de valeurs de paramètres.

> **Important**
>
> Si vous ne spécifiez pas d'identifiant de transaction, les modifications résultant de l'appel sont validées automatiquement.

L'exemple suivant exécute une opération d'insertion par lots.

```
package com.amazonaws.rdsdata.examples;

import com.amazonaws.services.rdsdata.AWSRDSData;
import com.amazonaws.services.rdsdata.AWSRDSDataClient;
import com.amazonaws.services.rdsdata.model.BatchExecuteStatementRequest;
```

```
import com.amazonaws.services.rdsdata.model.Field;
import com.amazonaws.services.rdsdata.model.SqlParameter;

import java.util.Arrays;

public class BatchExecuteExample {
  public static final String RESOURCE_ARN = "arn:aws:rds:us-
east-1:123456789012:cluster:mydbcluster";
  public static final String SECRET_ARN = "arn:aws:secretsmanager:us-
east-1:123456789012:secret:mysecret";

  public static void main(String[] args) {
      AWSRDSData rdsData = AWSRDSDataClient.builder().build();

    BatchExecuteStatementRequest request = new BatchExecuteStatementRequest()
            .withDatabase("test")
            .withResourceArn(RESOURCE_ARN)
            .withSecretArn(SECRET_ARN)
            .withSql("INSERT INTO test_table2 VALUES (:string, :number)")
            .withParameterSets(Arrays.asList(
                    Arrays.asList(
                            new SqlParameter().withName("string").withValue(new
 Field().withStringValue("Hello")),
                            new SqlParameter().withName("number").withValue(new
 Field().withLongValue(1L))
                    ),
                    Arrays.asList(
                            new SqlParameter().withName("string").withValue(new
 Field().withStringValue("World")),
                            new SqlParameter().withName("number").withValue(new
 Field().withLongValue(2L))
                    )
            ));

    rdsData.batchExecuteStatement(request);
  }
}
```

Contrôle du comportement en cas d’expiration de l’API de données

Tous les appels à l’API de données sont synchrones. Supposons que vous exécutiez une opération d’API de données qui exécute une instruction SQL comme `INSERT` ou `CREATE TABLE`. Si l’appel de l’API de données aboutit, le traitement SQL se termine lorsque l’appel est renvoyé.

Par défaut, l'API de données annule une opération et renvoie une erreur d'expiration du délai si le traitement de l'opération ne s'est pas terminé dans les 45 secondes. Dans ce cas, les données ne sont pas insérées, la table n'est pas créée, etc.

Vous pouvez utiliser l'API de données pour effectuer des opérations de longue durée qui ne peuvent pas être effectuées en 45 secondes. Si vous estimez qu'une opération, telle qu'une opération DDL ou `INSERT` en bloc sur une table de grande taille, durera plus de 45 secondes, vous pouvez spécifier le paramètre `continueAfterTimeout` de l'opération `ExecuteStatement`. Votre application recevra tout de même le message d'erreur d'expiration du délai. Toutefois, l'opération continuera et ne sera pas annulée. Pour obtenir un exemple, consultez Exécution d'une transaction SQL.

Si le kit AWS SDK correspondant à votre langage de programmation utilise son propre délai d'expiration pour les appels d'API ou les connexions au socket HTTP, assurez-vous que tous ces délais soient supérieurs à 45 secondes. Pour certains kits SDK, le délai d'expiration par défaut est inférieur à 45 secondes. Nous vous recommandons de spécifier des délais d'expiration spécifiques aux kits SDK ou aux clients d'au moins une minute. Cela permet d'éviter que votre application reçoive une erreur d'expiration du délai alors que l'opération de l'API de données aboutit. Et vous savez ainsi quand vous devez retenter l'opération ou non.

Supposons, par exemple, que le kit SDK renvoie une erreur d'expiration du délai à votre application, mais que l'opération de l'API de données se termine tout de même dans l'intervalle d'expiration de l'API. Dans ce cas, une nouvelle tentative d'exécution de cette opération risquerait d'insérer des données dupliquées ou de générer des résultats incorrects. Le kit SDK pourrait réessayer d'exécuter l'opération automatiquement, ce qui entraînerait des données incorrectes sans aucune action de la part de votre application.

L'intervalle d'expiration est particulièrement important pour le SDK Java 2. Dans ce kit SDK, le délai d'expiration des appels d'API et celui du socket HTTP sont tous deux de 30 secondes par défaut. Voici un exemple de procédure à suivre pour définir une valeur supérieure pour ces délais d'expiration :

```
public RdsDataClient createRdsDataClient() {
    return RdsDataClient.builder()
        .region(Region.US_EAST_1) // Change this to your desired Region
        .overrideConfiguration(createOverrideConfiguration())
        .httpClientBuilder(createHttpClientBuilder())
        .credentialsProvider(defaultCredentialsProvider()) // Change this to your
 desired credentials provider
        .build();
```

```
}

private static ClientOverrideConfiguration createOverrideConfiguration() {
    return ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofSeconds(60))
        .build();
}

private HttpClientBuilder createHttpClientBuilder() {
    return ApacheHttpClient.builder() // Change this to your desired HttpClient
        .socketTimeout(Duration.ofSeconds(60));
}
```

Voici un exemple équivalent utilisant le client de données asynchrone :

```
public static RdsDataAsyncClient createRdsDataAsyncClient() {
    return RdsDataAsyncClient.builder()
        .region(Region.US_EAST_1) // Change this to your desired Region
        .overrideConfiguration(createOverrideConfiguration())
        .credentialsProvider(defaultCredentialsProvider())  // Change this to your
 desired credentials provider
        .build();
}

private static ClientOverrideConfiguration createOverrideConfiguration() {
    return ClientOverrideConfiguration.builder()
        .apiCallAttemptTimeout(Duration.ofSeconds(60))
        .build();
}

private HttpClientBuilder createHttpClientBuilder() {
    return NettyNioAsyncHttpClient.builder() // Change this to your desired
 AsyncHttpClient
        .readTimeout(Duration.ofSeconds(60));
}
```

Utilisation de la bibliothèque cliente Java pour l’API de données RDS

Vous pouvez télécharger et utiliser une bibliothèque cliente Java pour l’API de données RDS (API de données). Cette bibliothèque cliente Java propose une autre manière d’utiliser l’API de données.

En utilisant cette bibliothèque, vous pouvez mapper vos classes côté client aux demandes et aux réponses de l'API de données. La prise en charge de ce mappage peut faciliter l'intégration à certains types Java précis, comme `Date`, `Time` et `BigDecimal`.

Téléchargement de la bibliothèque cliente Java pour l'API de données

La bibliothèque client Java de l'API de données est en open source dans GitHub à l'emplacement suivant :

https://github.com/awslabs/rds-data-api-client-library-java

Vous pouvez créer la bibliothèque manuellement à partir des fichiers sources, mais la bonne pratique consiste à la consommer en utilisant la gestion des dépendances Apache Maven. Ajoutez la dépendance suivante à votre fichier POM Maven.

Pour la version 2.x, compatible avec AWS SDK 2.x, utilisez ce qui suit :

```
<dependency>
   <groupId>software.amazon.rdsdata</groupId>
   <artifactId>rds-data-api-client-library-java</artifactId>
   <version>2.0.0</version>
</dependency>
```

Pour la version 1.x, compatible avec AWS SDK 1.x, utilisez ce qui suit :

```
<dependency>
    <groupId>software.amazon.rdsdata</groupId>
    <artifactId>rds-data-api-client-library-java</artifactId>
    <version>1.0.8</version>
</dependency>
```

Exemples relatifs à la bibliothèque cliente Java

Vous trouverez, ci-dessous, quelques exemples d'utilisation de la bibliothèque cliente Java de l'API de données. Ces exemples supposent que vous disposez d'un tableau `accounts` à deux colonnes : `accountId` et `name`. Vous disposez également de l'objet de transfert de données (DTO) suivant.

```
public class Account {
    int accountId;
```

```
    String name;
    // getters and setters omitted
}
```

La bibliothèque cliente vous permet de passer des DTO en tant que paramètres d'entrée. Les exemples suivants montrent comment les DTO des clients sont mappés à des jeux de paramètres d'entrée.

```
var account1 = new Account(1, "John");
var account2 = new Account(2, "Mary");
client.forSql("INSERT INTO accounts(accountId, name) VALUES(:accountId, :name)")
         .withParamSets(account1, account2)
         .execute();
```

Dans certains cas, il est plus facile de travailler avec des valeurs simples en tant que paramètres d'entrée. Pour cela, utilisez la syntaxe suivante.

```
client.forSql("INSERT INTO accounts(accountId, name) VALUES(:accountId, :name)")
         .withParameter("accountId", 3)
         .withParameter("name", "Zhang")
         .execute();
```

Voici un autre exemple qui fonctionne avec des valeurs simples en tant que paramètres d'entrée.

```

client.forSql("INSERT INTO accounts(accountId, name) VALUES(?, ?)", 4, "Carlos")
         .execute();

```

La bibliothèque cliente fournit le mappage automatique aux DTO lorsqu'un résultat est retourné. Les exemples suivants illustrent la manière dont le résultat est mappé à vos DTO.

```
List<Account> result = client.forSql("SELECT * FROM accounts")
          .execute()
          .mapToList(Account.class);

Account result = client.forSql("SELECT * FROM accounts WHERE account_id = 1")
          .execute()
          .mapToSingle(Account.class);
```

Dans de nombreux cas, l'ensemble de résultats de la base de données ne contient qu'une seule valeur. Afin de simplifier la récupération de ces résultats, la bibliothèque cliente propose l'API suivante :

```
int numberOfAccounts = client.forSql("SELECT COUNT(*) FROM accounts")
          .execute()
          .singleValue(Integer.class);
```

> **Note**
>
> La fonction `mapToList` convertit un ensemble de résultats SQL en liste d'objets définie par l'utilisateur. Nous ne prenons pas en charge l'utilisation de l'instruction `.withFormatRecordsAs(RecordsFormatType.JSON)` dans un appel `ExecuteStatement` pour la bibliothèque cliente Java, car elle a la même finalité. Pour plus d'informations, consultez Traitement des requêtes d'API Amazon RDS Data au format JSON.

Traitement des requêtes d'API Amazon RDS Data au format JSON

Lorsque vous appelez l'opération `ExecuteStatement`, vous pouvez choisir que les résultats de la requête soient retournés sous forme de chaîne au format JSON. Ainsi, vous pouvez utiliser les capacités d'analyse JSON de votre langage de programmation pour interpréter et reformater l'ensemble de résultats. Cela permet d'éviter d'écrire du code supplémentaire pour boucler sur l'ensemble de résultats et interpréter chaque valeur de colonne.

Pour demander l'ensemble de résultats au format JSON, vous transmettez le paramètre `formatRecordsAs` facultatif avec une valeur `JSON`. L'ensemble de résultats au format JSON est renvoyé dans le champ `formattedRecords` de la structure `ExecuteStatementResponse`.

L'action `BatchExecuteStatement` ne renvoie pas d'ensemble de résultats. Ainsi, l'option JSON ne s'applique pas à cette action.

Pour personnaliser les clés dans la structure de hachage JSON, définissez des alias de colonne dans l'ensemble de résultats. Vous pouvez le faire en utilisant la clause `AS` dans la liste des colonnes de votre requête SQL.

Vous pouvez utiliser la fonctionnalité JSON pour faciliter la lecture de l'ensemble des résultats et faire correspondre son contenu à des cadres spécifiques à la langue. Étant donné que le volume de l'ensemble des résultats codés en ASCII est plus important que celui de la représentation par défaut,

vous pouvez choisir cette dernière pour les requêtes qui renvoient un grand nombre de lignes ou des valeurs de colonnes importantes qui consomment plus de mémoire que celle dont dispose votre application.

Rubriques

Récupération des résultats des requêtes au format JSON

Pour recevoir l'ensemble de résultats sous forme de chaîne JSON, incluez `.withFormatRecordsAs(RecordsFormatType.JSON)` dans l'appel `ExecuteStatement`. La valeur de retour revient sous la forme d'une chaîne JSON dans le champ `formattedRecords`. Dans ce cas, le champ `columnMetadata` est `null`. Les étiquettes des colonnes sont les clés de l'objet qui représente chaque ligne. Ces noms de colonnes sont répétés pour chaque ligne de l'ensemble de résultats. Les valeurs des colonnes sont des chaînes de caractères entre guillemets, des valeurs numériques ou des valeurs spéciales représentant `true`, `false`, ou `null`. Les métadonnées des colonnes, telles que les contraintes de longueur et le type précis des nombres et des chaînes de caractères, ne sont pas conservées dans la réponse JSON.

Si vous omettez l'appel `.withFormatRecordsAs()` ou spécifiez un paramètre de `NONE`, l'ensemble de résultats est renvoyé au format binaire en utilisant les champs `Records` et `columnMetadata`.

Mappage des types de données

Les valeurs SQL de l'ensemble de résultats sont mises en correspondance avec un ensemble plus petit de types JSON. Les valeurs sont représentées dans JSON sous forme de chaînes de caractères, de nombres et de certaines constantes spéciales telles que `true`, `false` et `null`. Vous pouvez convertir ces valeurs en variables dans votre application, en utilisant un typage fort ou faible, selon le langage de programmation utilisé.

Type de données JDBC	Type de données JSON
`INTEGER`, `TINYINT`, `SMALLINT`, `BIGINT`	Numéro par défaut. Chaîne si l'option `LongReturnType` est définie sur `STRING`.

Type de données JDBC	Type de données JSON
`FLOAT`, `REAL`, `DOUBLE`	Nombre
`DECIMAL`	Chaîne par défaut. Nombre si l'option `DecimalReturnType` est définie sur `DOUBLE_OR_LONG` .
`STRING`	Chaîne
`BOOLEAN`, `BIT`	Booléen
`BLOB`, `BINARY`, `VARBINARY` , `LONGVARBINARY`	Chaîne avec encodage base64.
`CLOB`	Chaîne
`ARRAY`	Tableau
`NULL`	`null`
Autres types (y compris les types liés à la date et à l'heure)	Chaîne

Résolution des problèmes

La réponse JSON est limitée à 10 mégaoctets. Si la réponse est supérieure à cette limite, votre programme reçoit une erreur `BadRequestException`. Dans ce cas, vous pouvez résoudre l'erreur en utilisant l'une des techniques suivantes :

- Réduisez le nombre de lignes dans l'ensemble de résultats. Pour ce faire, ajoutez une clause `LIMIT`. Vous pouvez diviser un grand ensemble de résultats en plusieurs petits ensembles en envoyant plusieurs requêtes avec des clauses `LIMIT` et `OFFSET`.

  Si l'ensemble de résultats comprend des lignes qui sont filtrées par la logique d'application, vous pouvez supprimer ces lignes de l'ensemble de résultats en ajoutant d'autres conditions à la clause `WHERE`.
- Réduisez le nombre de colonnes dans l'ensemble de résultats. Pour ce faire, supprimez les éléments de la liste Select de la requête.

- Raccourcissez les étiquettes des colonnes en utilisant des alias de colonnes dans la requête. Chaque nom de colonne est répété dans la chaîne JSON pour chaque ligne de l'ensemble de résultats. Ainsi, un résultat de requête comportant de longs noms de colonnes et de nombreuses lignes pourrait dépasser la limite de taille. En particulier, utilisez des alias de colonne pour les expressions complexes afin d'éviter que l'expression entière ne soit répétée dans la chaîne JSON.
- Bien qu'en SQL, vous puissiez utiliser des alias de colonne pour produire un ensemble de résultats comportant plusieurs colonnes portant le même nom, les doublons de noms de clés ne sont pas autorisés en JSON. L'API de données RDS renvoie une erreur si vous demandez l'ensemble de résultats au format JSON et que plusieurs colonnes portent le même nom. Ainsi, assurez-vous que toutes les étiquettes de colonne portent des noms uniques.

Exemples

Les exemples Java suivants montrent comment appeler `ExecuteStatement` avec la réponse sous forme de chaîne formatée en JSON, puis interpréter l'ensemble de résultats. Remplacez les valeurs appropriées pour les paramètres *`databaseName`*, *`secretStoreArn`* et *`clusterArn`*.

L'exemple Java suivant illustre une requête qui renvoie une valeur numérique décimale dans l'ensemble de résultats. Les appels `assertThat` vérifient que les champs de la réponse présentent les propriétés attendues, conformément aux règles applicables aux ensembles de résultats JSON.

Cet exemple fonctionne avec le schéma et les exemples de données suivants :

```
create table test_simplified_json (a float);
insert into test_simplified_json values(10.0);
```

```
public void JSON_result_set_demo() {
    var sql = "select * from test_simplified_json";
    var request = new ExecuteStatementRequest()
      .withDatabase(databaseName)
      .withSecretArn(secretStoreArn)
      .withResourceArn(clusterArn)
      .withSql(sql)
      .withFormatRecordsAs(RecordsFormatType.JSON);
    var result = rdsdataClient.executeStatement(request);
}
```

La valeur du champ `formattedRecords` du programme précédent est :

```
[{"a":10.0}]
```

Les champs `Records` et `ColumnMetadata` dans la réponse sont tous deux nuls, en raison de la présence de l'ensemble de résultats JSON.

L'exemple Java suivant illustre une requête qui renvoie une valeur numérique entière dans l'ensemble de résultats. L'exemple appelle `getFormattedRecords` à ne retourner que la chaîne formatée en JSON et à ignorer les autres champs de réponse qui sont vides ou nuls. L'exemple désérialise le résultat dans une structure représentant une liste de registres. Chaque registre possède des champs dont les noms correspondent aux alias des colonnes de l'ensemble de résultats. Cette technique simplifie le code qui analyse l'ensemble des résultats. Votre application n'a pas besoin de boucler sur les lignes et les colonnes de l'ensemble de résultats et de convertir chaque valeur dans le type approprié.

Cet exemple fonctionne avec le schéma et les exemples de données suivants :

```
create table test_simplified_json (a int);
insert into test_simplified_json values(17);
```

```
public void JSON_deserialization_demo() {
    var sql = "select * from test_simplified_json";
    var request = new ExecuteStatementRequest()
      .withDatabase(databaseName)
      .withSecretArn(secretStoreArn)
      .withResourceArn(clusterArn)
      .withSql(sql)
      .withFormatRecordsAs(RecordsFormatType.JSON);
    var result = rdsdataClient.executeStatement(request)
      .getFormattedRecords();

/* Turn the result set into a Java object, a list of records.
   Each record has a field 'a' corresponding to the column
   labelled 'a' in the result set. */
    private static class Record { public int a; }
    var recordsList = new ObjectMapper().readValue(
        response, new TypeReference<List<Record>>() {
        });
}
```

La valeur du champ `formattedRecords` du programme précédent est :

```
[{"a":17}]
```

Pour récupérer la colonne a de la ligne de résultats 0, l'application doit se référer à `recordsList.get(0).a`.

En revanche, l'exemple Java suivant montre le type de code nécessaire pour créer une structure de données contenant l'ensemble de résultats lorsque vous n'utilisez pas le format JSON. Dans ce cas, chaque ligne de l'ensemble de résultats contient des champs comportant des informations sur un seul utilisateur. La création d'une structure de données pour représenter l'ensemble des résultats nécessite l'exécution en boucle des lignes. Pour chaque ligne, le code récupère la valeur de chaque champ, effectue une conversion de type appropriée et affecte le résultat au champ correspondant dans l'objet représentant la ligne. Ensuite, le code ajoute l'objet représentant chaque utilisateur à la structure de données représentant l'ensemble des résultats. Si la requête était modifiée pour réorganiser, ajouter ou supprimer des champs dans l'ensemble de résultats, le code de l'application devrait également être modifié.

```
/* Verbose result-parsing code that doesn't use the JSON result set format */
for (var row: response.getRecords()) {
    var user = User.builder()
      .userId(row.get(0).getLongValue())
      .firstName(row.get(1).getStringValue())
      .lastName(row.get(2).getStringValue())
      .dob(Instant.parse(row.get(3).getStringValue()))
      .build();
    result.add(user);
  }
```

Les exemples suivants montrent les valeurs du champ `formattedRecords` pour des ensembles de résultats comportant différents nombres de colonnes, alias de colonnes et types de données de colonnes.

Si l'ensemble de résultats comprend plusieurs lignes, chaque ligne est représentée par un objet qui constitue un élément de tableau. Chaque colonne de l'ensemble de résultats devient une clé dans l'objet. Les clés sont répétées pour chaque ligne de l'ensemble de résultats. Ainsi, pour les ensembles de résultats composés de nombreuses lignes et colonnes, vous devrez peut-être définir des alias de colonne courts pour éviter de dépasser la limite de longueur pour l'ensemble de la réponse.

Cet exemple fonctionne avec le schéma et les exemples de données suivants :

```
create table sample_names (id int, name varchar(128));
insert into sample_names values (0, "Jane"), (1, "Mohan"), (2, "Maria"), (3, "Bruce"),
 (4, "Jasmine");
```

```
[{"id":0,"name":"Jane"},{"id":1,"name":"Mohan"},
{"id":2,"name":"Maria"},{"id":3,"name":"Bruce"},{"id":4,"name":"Jasmine"}]
```

Si une colonne de l'ensemble de résultats est définie comme une expression, le texte de l'expression devient la clé JSON. Ainsi, il est généralement pratique de définir un alias de colonne descriptif pour chaque expression de la liste Select de la requête. Par exemple, la requête suivante inclut des expressions telles que des appels de fonction et des opérations arithmétiques dans sa liste Select.

```
select count(*), max(id), 4+7 from sample_names;
```

Ces expressions sont transmises à l'ensemble de résultats JSON en tant que clés.

```
[{"count(*)":5,"max(id)":4,"4+7":11}]
```

L'ajout de colonnes `AS` avec des étiquettes descriptives rend les clés plus simples à interpréter dans l'ensemble de résultats JSON.

```
select count(*) as rows, max(id) as largest_id, 4+7 as addition_result from
 sample_names;
```

Avec la requête SQL révisée, les étiquettes des colonnes définies par les clauses `AS` sont utilisées comme noms de clés.

```
[{"rows":5,"largest_id":4,"addition_result":11}]
```

La valeur de chaque paire clé-valeur dans la chaîne JSON peut être une chaîne entre guillemets. La chaîne peut contenir des caractères unicode. Si la chaîne contient des séquences d'échappement ou les caractères `"` ou `\`, ces caractères sont précédés de caractères d'échappement barre oblique inverse. Les exemples suivants de chaînes JSON illustrent ces possibilités. Par exemple, le résultat `string_with_escape_sequences` contient les caractères spéciaux retour arrière, saut de ligne, retour chariot, tabulation, saut de page et `\`.

```
[{"quoted_string":"hello"}]
[{"unicode_string":"####"}]
```

```
[{"string_with_escape_sequences":"\b \n \r \t \f \\ '"}]
```

La valeur de chaque paire clé-valeur dans la chaîne JSON peut également représenter un nombre. Le nombre peut être un entier, une valeur en virgule flottante, une valeur négative ou une valeur représentée en notation exponentielle. Les exemples suivants de chaînes JSON illustrent ces possibilités.

```
[{"integer_value":17}]
[{"float_value":10.0}]
[{"negative_value":-9223372036854775808,"positive_value":9223372036854775807}]
[{"very_small_floating_point_value":4.9E-324,"very_large_floating_point_value":1.79769313486231
```

Les valeurs booléennes et nulles sont représentées par les mots-clés spéciaux non entourés de guillemets `true`, `false` et `null`. Les exemples suivants de chaînes JSON illustrent ces possibilités.

```
[{"boolean_value_1":true,"boolean_value_2":false}]
[{"unknown_value":null}]
```

Si vous sélectionnez une valeur de type BLOB, le résultat est représenté dans la chaîne JSON sous la forme d'une valeur avec encodage base64. Pour reconvertir la valeur dans sa représentation originale, vous pouvez utiliser la fonction de décodage appropriée dans le langage de votre application. Par exemple, en Java, vous appelez la fonction `Base64.getDecoder().decode()`. L'exemple de sortie suivant montre le résultat de la sélection d'une valeur BLOB de `hello world` et du renvoi de l'ensemble de résultats sous forme de chaîne JSON.

```
[{"blob_column":"aGVsbG8gd29ybGQ="}]
```

L'exemple Python suivant montre comment accéder aux valeurs du résultat d'un appel à la fonction Python `execute_statement`. L'ensemble de résultats est une valeur de chaîne de caractères dans le champ `response['formattedRecords']`. Le code transforme la chaîne JSON en une structure de données en appelant la fonction `json.loads`. Ensuite, chaque ligne de l'ensemble de résultats est un élément de liste dans la structure de données ; dans chaque ligne, vous pouvez faire référence à chaque champ de l'ensemble de résultats par son nom.

```
import json

result = json.loads(response['formattedRecords'])
print (result[0]["id"])
```

L'exemple JavaScript suivant montre comment accéder aux valeurs du résultat d'un appel à la fonction JavaScript `executeStatement`. L'ensemble de résultats est une valeur de chaîne de caractères dans le champ `response.formattedRecords`. Le code transforme la chaîne JSON en une structure de données en appelant la fonction `JSON.parse`. Ensuite, chaque ligne de l'ensemble de résultats représente un élément de tableau dans la structure de données ; dans chaque ligne, vous pouvez faire référence à chaque champ de l'ensemble de résultats par son nom.

```
<script>
    const result = JSON.parse(response.formattedRecords);
    document.getElementById("display").innerHTML = result[0].id;
</script>
```

Dépannage de l'API de données Amazon RDS

Utilisez les sections suivantes, dont le titre correspond aux messages d'erreur courants, pour vous aider à résoudre les problèmes que vous rencontrez avec l'API de données Amazon RDS (API de données).

Rubriques

Transaction <transaction_ID> isn't found

Dans ce cas, l'ID de transaction spécifié dans un appel de l'API de données est introuvable. La cause de ce problème, parmi les suivantes, est ajoutée au message d'erreur :

- La transaction peut avoir expiré.

Assurez-vous que chaque appel transactionnel s’exécute dans un délai de trois minutes à la suite du précédent.

Il est également possible que l'identifiant de transaction spécifié n'ait pas été créé par un BeginTransactionappel. Assurez-vous que l’ID de transaction de votre appel est valide.

- Un appel précédent a entraîné l’arrêt de votre transaction.

  La transaction a déjà été arrêtée par votre appel `CommitTransaction` ou `RollbackTransaction`.

- La transaction a été abandonnée en raison d’une erreur issue d’un appel précédent.

  Vérifiez si vos appels précédents ont généré des exceptions.

Pour plus d’informations sur l’exécution des transactions, consultez Appel de l’API de données Amazon RDS.

Packet for query is too large

Dans ce cas, le jeu de résultats renvoyé pour une ligne était trop volumineux. La taille de l’API de données ne doit pas dépasser 64 Ko par ligne dans le jeu de résultat renvoyé par la base de données.

Pour résoudre ce problème, assurez-vous que la taille de chaque ligne d’un jeu de résultat est inférieure ou égale à 64 Ko.

Database Response Exceeded Size Limit

Dans ce cas, la taille du jeu de résultat renvoyé par la base de données était trop grande. La limite de l’API de données est de 1 Mio dans le jeu de résultats renvoyé par la base de données.

Pour résoudre ce problème, assurez-vous que les appels à l’API de données renvoient au maximum 1 Mio. Si vous avez besoin de renvoyer plus de 1 Mio, vous pouvez utiliser plusieurs appels `ExecuteStatement` avec la clause `LIMIT` dans votre requête.

Pour plus d’informations sur la clause `LIMIT`, consultez Syntaxe SELECT dans la documentation de MySQL.

HttpEndpointn'est pas activé pour le cluster <cluster_ID>

Vérifiez les causes potentielles suivantes de ce problème :

- Le cluster de bases de données Aurora ne prend pas en charge l'API de données. Pour plus d'informations sur les types de clusters de bases de données pris en charge par l'API de données RDS, consultez the section called "Disponibilité des régions et des versions de pour l'API de données Amazon RDS".
- L'API de données n'est pas activée pour le cluster de bases de données Aurora. Pour utiliser l'API de données avec un cluster de bases de données Aurora, l'API de données doit être activée pour le cluster de bases de données. Pour plus d'informations sur l'activation de l'API de données, consultez Activation de l'API de données Amazon RDS.
- Le cluster de bases de données a été renommé après l'activation de l'API de données. Dans ce cas, désactivez l'API de données pour ce cluster, puis réactivez-la.
- L'ARN que vous avez spécifié ne correspond pas précisément à l'ARN du cluster. Vérifiez que l'ARN renvoyé par une autre source ou créé par la logique du programme correspond exactement à l'ARN du cluster. Par exemple, assurez-vous que l'ARN que vous utilisez respecte la casse adéquate pour tous les caractères alphabétiques.

DatabaseErrorException: La transaction exécute toujours une requête

Lorsque votre application envoie une demande avec un ID de transaction qui exécute déjà une autre requête, l'API de données renvoie immédiatement cette erreur à votre application. Cette situation peut se produire si votre application envoie des demandes de façon asynchrone, par exemple à l'aide d'un mécanisme comme les « promesses » en Javascript.

Pour résoudre ce problème, attendez que la demande précédente soit terminée, puis réessayez. Vous pouvez répéter la tentative jusqu'à ce que l'erreur disparaisse ou qu'un autre type d'erreur soit renvoyé à l'application.

Cette condition peut se produire avec l'API de données pour Aurora Serverless v2 et les instances provisionnées. Dans l'API de données pour Aurora Serverless v1, les demandes envoyées avec le même ID de transaction sont mises en attente jusqu'à ce que la précédente soit terminée. Cependant, cet ancien comportement peut entraîner des délais d'expiration si la demande précédente met trop de temps à s'exécuter. Si vous migrez une ancienne application d'API de données qui envoie des demandes simultanées, adaptez votre logique de gestion des exceptions afin de prendre en compte ce nouveau type d'erreur.

Exception de résultat non pris en charge

L'API de données ne prend pas en charge tous les types de données. Cette erreur se produit lorsque vous exécutez une requête qui renvoie un type de données non pris en charge.

Pour contourner ce problème, convertissez le type de données non pris en charge en `TEXT`. Par exemple :

```
SELECT custom_type::TEXT FROM my_table;
-- OR
SELECT CAST(custom_type AS TEXT) FROM my_table;
```

Les instructions multiples ne sont pas prises en charge

Les instructions multiples ne sont pas prises en charge dans l'API de données pour Aurora sans serveur v2 et les clusters provisionnés. Toute tentative d'exécuter plusieurs instructions dans un seul appel d'API entraîne cette erreur.

Pour exécuter des instructions multiples, utilisez des appels d'API `ExecuteStatement` distincts ou utilisez l'API `BatchExecuteStatement` pour le traitement par lots.

Le paramètre de schéma n'est pas pris en charge

Aurora sans serveur v1 ignore silencieusement le paramètre de schéma. Cependant, Aurora sans serveur v2 et les clusters provisionnés rejettent explicitement les appels d'API contenant le paramètre de schéma.

Pour résoudre ce problème, supprimez le paramètre de schéma de tous les appels à l'API de données lorsque vous utilisez Aurora sans serveur v2 ou des clusters provisionnés.

IPv6 problèmes de connectivité

Si vous rencontrez des problèmes lors de la connexion à l'API de données à l'aide de IPv6 points de terminaison, vérifiez les causes potentielles suivantes :

- Le réseau n'est pas compatible IPv6 : vérifiez que votre infrastructure réseau est compatible IPv6 et que IPv6 le routage est correctement configuré.
- Problèmes de résolution DNS : assurez-vous que votre résolveur DNS peut résoudre les enregistrements AAAA pour les points de terminaison Dual-Stack (par ex., `rds-data.us-east-1.api.aws`).

- Configuration du groupe de sécurité : mettez à jour les règles du groupe de sécurité pour autoriser le IPv6 trafic sur le port 443 (HTTPS). Ajoutez des règles pour les blocs IPv6 CIDR (par exemple, `::/0` pour toutes les IPv6 adresses).
- Configuration de l'ACL réseau : assurez-vous que le réseau ACLs autorise le IPv6 trafic sur les ports requis.
- Compatibilité avec les bibliothèques clientes : vérifiez que vos bibliothèques clientes HTTP AWS SDKs prennent en charge IPv6 la connectivité à double pile.
- Configuration du point de terminaison VPC : si vous l'utilisez PrivateLink, assurez-vous que votre point de terminaison VPC est configuré pour prendre en charge IPv6 et que des blocs d'adresse CIDR sont attribués aux sous-réseaux associés. IPv6

Pour résoudre les problèmes de IPv6 connectivité, procédez comme suit :

1. Testez la connectivité à l'aide des points de terminaison IPv4 -only (`.amazonaws.com`) pour vérifier que le problème est spécifique à. IPv6
2. Utilisez les outils de diagnostic réseau pour vérifier la IPv6 connectivité aux points de terminaison à double pile.
3. Vérifiez les CloudTrail journaux pour détecter toute erreur d'authentification ou d'autorisation lors de l'utilisation de IPv6 points de terminaison.
4. Vérifiez que votre application est correctement configurée pour utiliser le nouveau point de terminaison URLs à double pile.

Journalisation des appels d’API de données Amazon RDS avec AWS CloudTrail

L’API de données RDS (API de données) est intégrée avec AWS CloudTrail, un service qui fournit un enregistrement des actions effectuées par un utilisateur, un rôle ou un service AWS dans l’API de données. CloudTrail capture tous les appels d’API pour l’API de données en tant qu’événements, y compris les appels à partir de la console Amazon RDS et les appels de code à des opérations d’API de données. Si vous créez un journal de suivi, vous pouvez transmettre en continu les événements CloudTrail vers un compartiment Amazon S3, y compris les événements concernant l’API de données. Grâce aux données collectées par CloudTrail, vous pouvez déterminer de nombreuses informations. Ces informations comprennent la demande qui a été faite à l’API de

données, l'adresse IP à partir de laquelle la demande a été faite, qui a effectué la demande, quand elle a eu lieu, ainsi que des détails supplémentaires.

Pour en savoir plus sur CloudTrail, consultez le Guide de l'utilisateur AWS CloudTrail.

Utilisation des informations de l'API de données dans CloudTrail

CloudTrail est activé dans votre compte AWS lors de la création de ce dernier. Lorsqu'une activité prise en charge (événements de gestion) se produit dans l'API de données, cette activité est enregistrée dans un événement CloudTrail avec d'autres événements de service AWS dans l'Historique des événements. Vous pouvez afficher, rechercher et télécharger les événements de gestion récents dans votre compte AWS. Pour plus d'informations, consultez Travailler avec l'historique des événements CloudTrail dans le AWS CloudTrailGuide de l'utilisateur.

Pour un enregistrement continu des événements dans votre compte AWS, y compris des événements concernant l'API de données, créez un journal de suivi. Un journal de suivi permet à CloudTrail de livrer des fichiers journaux dans un compartiment Amazon S3. Par défaut, lorsque vous créez un journal de suivi dans la console, il s'applique à toutes les régions AWS. Le journal d'activité consigne les événements de toutes les Régions dans la partition AWSAWS et transfère les fichiers journaux dans le compartiment Amazon S3 de votre choix. En outre, vous pouvez configurer d'autres services AWS pour analyser plus en profondeur les données d'événement collectées dans les journaux CloudTrail et agir sur celles-ci. Pour plus d'informations, consultez les rubriques suivantes dans le AWS CloudTrailGuide de l'utilisateur :

- Présentation de la création d'un journal d'activité
- Intégrations et services pris en charge par CloudTrail
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers journaux CloudTrail de plusieurs régions et Réception de fichiers journaux CloudTrail de plusieurs comptes

Toutes les opérations de l'API de données sont enregistrées par CloudTrail et documentées dans la référence API du service de données Amazon RDS. Par exemple, les appels aux opérations `BatchExecuteStatement`, `BeginTransaction`, `CommitTransaction` et `ExecuteStatement` génèrent des entrées dans les fichiers journaux CloudTrail.

Chaque événement ou entrée du journal contient des informations sur la personne qui a généré la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été effectuée avec les informations d'identification utilisateur racine ou .
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la demande a été effectuée par un autre service AWS.

Pour plus d'informations, consultez l'élément userIdentity CloudTrail.

Inclusion et exclusion d'événements d'API de données d'un journal d'activité AWS CloudTrail

La plupart des utilisateurs d'API de données s'appuient sur les événements d'un journal d'activité AWS CloudTrail pour fournir un enregistrement des opérations d'API de données. Les données d'événements ne révèlent pas le nom de la base de données, le nom du schéma ni les instructions SQL dans les demandes adressées à l'API de données. Toutefois, identifier quel utilisateur a exécuté un type d'appel sur un cluster de bases de données spécifique à un instant donné peut contribuer à repérer les modèles d'accès anormaux.

Inclusion d'événements d'API de données dans un journal d'activité AWS CloudTrail

Pour Aurora PostgreSQL sans serveur v2 et les bases de données provisionnées, les opérations d'API de données suivantes sont enregistrées en tant qu'événements de données sur AWS CloudTrail. Les événements de données sont des opérations d'API de plan de données à volume élevé que CloudTrail ne consigne pas par défaut. Des frais supplémentaires s'appliquent pour les événements de données. Pour en savoir plus sur la tarification de CloudTrail, consultez Tarification AWS CloudTrail.

- BatchExecuteStatement
- BeginTransaction
- CommitTransaction
- ExecuteStatement
- RollbackTransaction

Vous pouvez utiliser la console CloudTrail, l'AWS CLI, ou les opérations d'API CloudTrail pour journaliser les opérations d'API de données. Dans la console CloudTrail, choisissez API de données RDS - Cluster de bases de données pour le type d'événement de données. Pour plus d'informations,

consultez Journalisation des événements de données avec la AWS Management Console dans le Guide de l'utilisateur AWS CloudTrail.

À l'aide de l'AWS CLI, exécutez la commande `aws cloudtrail put-event-selectors` pour enregistrer ces opérations d'API de données pour votre journal d'activité. Pour enregistrer tous les événements de l'API de données sur les clusters de bases de données, indiquez `AWS::RDS::DBCluster` pour le type de ressource. L'exemple suivant illustre la journalisation de tous les événements de l'API de données sur les clusters de bases de données. Pour plus d'informations, consultez Journalisation des événements de données avec la AWS Command Line Interface dans le Guide de l'utilisateur AWS CloudTrail.

```
aws cloudtrail put-event-selectors --trail-name trail_name --advanced-event-selectors \
'{
   "Name": "RDS Data API Selector",
   "FieldSelectors": [
      {
         "Field": "eventCategory",
         "Equals": [
            "Data"
         ]
      },
      {
         "Field": "resources.type",
         "Equals": [
            "AWS::RDS::DBCluster"
         ]
      }
   ]
}'
```

Vous pouvez configurer des sélecteurs d'événements avancés pour appliquer un filtre supplémentaire sur les champs `readOnly`, `eventName`, et `resources.ARN`. Pour plus d'informations sur ces champs, consultez AdvancedFieldSelector.

Exclusion d'événements d'API de données d'un journal d'activité AWS CloudTrail (Aurora Serverless v1 uniquement)

Les événements de l'API de données sont des événements de gestion dans Aurora Serverless v1. Par défaut, tous les événements de l'API de données sont inclus dans un journal d'activité AWS CloudTrail. Cependant, comme l'API de données peut générer un grand nombre d'événements,

vous pouvez exclure ces événements de votre journal d'activité CloudTrail. Le paramètre Exclure les événements de l'API de données Amazon RDS exclut tous les événements de l'API de données du journal d'activité. Vous ne pouvez pas exclure des événements d'API de données spécifiques.

Pour exclure des événements d'API de données d'un journal d'activité, procédez comme suit :

- Dans la console CloudTrail, choisissez le paramètre Exclude Amazon RDS Data API events (Exclure les événements d'API de données Amazon RDS) lorsque vous créez un journal d'activité ou mettez à jour un journal d'activité.
- Dans l'API CloudTrail API, utilisez l'opération PutEventSelectors. Si vous utilisez des sélecteurs d'événements avancés, vous pouvez exclure les événements de l'API de données en configurant le champ `eventSource` avec une valeur différente de `rdsdata.amazonaws.com`. Si vous utilisez des sélecteurs d'événements basiques, vous pouvez exclure les événements de l'API de données en définissant la valeur de l'attribut `ExcludeManagementEventSources` sur `rdsdata.amazonaws.com`. Pour plus d'informations, consultez Journalisation des événements avec l'AWS Command Line Interface dans le Guide de l'utilisateur AWS CloudTrail.

> **Warning**
>
> L'exclusion d'événements d'API de données d'un journal CloudTrail peut masquer des actions d'API de données. Soyez prudent lorsque vous accordez aux principaux l'autorisation `cloudtrail:PutEventSelectors` nécessaire pour effectuer cette opération.

Vous pouvez désactiver cette exclusion à tout moment en modifiant le paramétrage de la console ou les sélecteurs d'événements pour un journal d'activité. Le journal d'activité commencera alors à enregistrer les événements d'API de données. Toutefois, il ne pourra pas récupérer les événements d'API de données survenus pendant que l'exclusion était effective.

Lorsque vous excluez des événements d'API de données à l'aide de la console ou de l'API, l'opération d'API `PutEventSelectors` CloudTrail qui en résulte est également enregistrée dans vos journaux CloudTrail. Si des événements d'API de données n'apparaissent pas dans vos journaux CloudTrail, recherchez un événement `PutEventSelectors` dont l'attribut `ExcludeManagementEventSources` est défini sur `rdsdata.amazonaws.com`.

Pour plus d'informations, consultez Journalisation des événements de gestion pour les journaux d'activité dans le Guide de l'utilisateur AWS CloudTrail.

Présentation des entrées des fichiers journaux de l’API de données

Un journal d’activité est une configuration qui permet la livraison d’événements sous forme de fichiers journaux vers un compartiment Amazon S3 que vous spécifiez. Les fichiers journaux CloudTrail peuvent contenir une ou plusieurs entrées. Un événement représente une demande individuelle émise à partir d’une source quelconque et comprend des informations sur l’action demandée, la date et l’heure de l’action, les paramètres de la demande, etc. Les fichiers journaux CloudTrail ne constituent pas une série ordonnée retraçant les appels d’API publics. Ils ne suivent aucun ordre précis.

Aurora PostgreSQL sans serveur v2 et les bases de données provisionnées

L’exemple suivant présente une entrée de journal CloudTrail qui illustre l’opération `ExecuteStatement` d’Aurora PostgreSQL sans serveur v2 et des bases de données provisionnées. Pour ces bases de données, tous les événements de l’API de données sont des événements de données dont la source est rdsdataapi.amazonaws.com et le type d’événement est Rds Data Service.


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T00:49:34Z",
    "eventSource": "rdsdataapi.amazonaws.com",
    "eventName": "ExecuteStatement",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.102 Python/3.7.2 Windows/10 botocore/1.12.92",
    "requestParameters": {
        "continueAfterTimeout": false,
        "database": "**********",
        "includeResultMetadata": false,
        "parameters": [],
        "resourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-database-1",
        "schema": "**********",
```

```
        "secretArn": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:dataapisecret-ABC123",
        "sql": "**********"
    },
    "responseElements": null,
    "requestID": "6ba9a36e-b3aa-4ca8-9a2e-15a9eada988e",
    "eventID": "a2c7a357-ee8e-4755-a0d0-aed11ed4253a",
    "eventType": "Rds Data Service",
    "recipientAccountId": "123456789012"
}
```

Aurora Serverless v1

L'exemple suivant montre comment apparaît l'entrée de journal CloudTrail de l'exemple précédent pour Aurora Serverless v1. Pour Aurora Serverless v1, tous les événements sont des événements de gestion dont la source est rdsdata.amazonaws.com et le type d'événement est AwsApiCall.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/johndoe",
        "accountId": "123456789012",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "johndoe"
    },
    "eventTime": "2019-12-18T00:49:34Z",
    "eventSource": "rdsdata.amazonaws.com",
    "eventName": "ExecuteStatement",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/1.16.102 Python/3.7.2 Windows/10 botocore/1.12.92",
    "requestParameters": {
        "continueAfterTimeout": false,
        "database": "**********",
        "includeResultMetadata": false,
        "parameters": [],
        "resourceArn": "arn:aws:rds:us-east-1:123456789012:cluster:my-database-1",
        "schema": "**********",
        "secretArn": "arn:aws:secretsmanager:us-
east-1:123456789012:secret:dataapisecret-ABC123",
```

```
        "sql": "**********"
    },
    "responseElements": null,
    "requestID": "6ba9a36e-b3aa-4ca8-9a2e-15a9eada988e",
    "eventID": "a2c7a357-ee8e-4755-a0d0-aed11ed4253a",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

Surveillance des requêtes de l'API de données RDS avec Performance Insights

Si votre cluster Aurora exécute Aurora Serverless v2 ou des instances provisionnées, vous pouvez utiliser Performance Insights avec l'API de données RDS.

Pour plus d'informations sur l'utilisation de Performance Insights avec Aurora, consultez Surveillance de la charge de la base de données avec Performance Insights sur .

Représentation des requêtes de l'API de données RDS dans Performance Insights

Avec l'API de données, votre cluster Aurora traite les requêtes en fonction des appels d'API de données que vous soumettez depuis votre application. L'API de données exécute également certaines instructions SQL dans le cadre de son propre fonctionnement interne, telles que l'annulation de requêtes dépassant le seuil du délai d'exécution. Les deux types d'opérations SQL sont présentés dans les statistiques et les graphiques de Performance Insights.

- Lorsqu'une requête de l'API de données est envoyée à un cluster Aurora, le champ Hôte dans le tableau de bord PI apparaît comme API de données RDS. Pour Aurora PostgreSQL, le champ application_name présente la valeur `rds-data-api`. Recherchez ces étiquettes lorsque vous analysez la charge de la base de données en utilisant la dimension Principaux hôtes ou Principales applications.
- Toutes les requêtes internes exécutées par l'API de données pour gérer les aspects de la base de données tels que le pool de connexions et les délais d'expiration des requêtes sont annotées avec un préfixe API de données RDS. Exemple : `/* RDS Data API */ select * from my_table;` recherche ces préfixes lorsque vous analysez la charge de la base de données en

fonction des Principaux éléments SQL en tant que dimension. utilisés comme dimension. Les instructions sont annotées à l’aide d’un commentaire SQL de `/* RDS Data API */`.

Utilisation de l’éditeur de requêtes Aurora

L’éditeur de requêtes Aurora vous permet d’exécuter des instructions SQL sur votre cluster de bases de données Aurora via la AWS Management Console. Vous pouvez exécuter des requêtes SQL, des instructions de manipulation de données (DML) et des instructions de définition de données (DDL). L’interface de console vous permet d’effectuer la maintenance de la base de données, de produire des rapports et de réaliser des tests SQL. Vous pouvez éviter de configurer la configuration réseau pour vous connecter à votre cluster de base de données à partir d'un système client distinct tel qu'une EC2 instance ou un ordinateur portable.

L’éditeur de requête nécessite un cluster de bases de données Aurora pour lequel l’API RDS de données (API de données) est activée. Pour en savoir plus sur les clusters de bases de données qui prennent en charge l’API de données et sur son activation, consultez Utilisation de l’API de données Amazon RDS. Le SQL que vous pouvez exécuter est limité par les contraintes de l’API de données. Pour plus d’informations, consultez the section called “Limitations”.

Disponibilité de l’éditeur de requête

L'éditeur de requêtes est disponible pour les clusters de base de données Aurora utilisant les versions des moteurs Aurora MySQL et Aurora PostgreSQL qui prennent en charge l'API Data, et dans Régions AWS le répertoire où l'API Data est disponible. Pour de plus amples informations, veuillez consulter Régions et moteurs de base de données Aurora pris en charge pour l’API de données RDS.

Autorisation de l’accès à l’éditeur de requête

Pour exécuter des requêtes dans l’éditeur de requête, les utilisateurs doivent y être autorisés. Vous pouvez autoriser un utilisateur à exécuter des requêtes dans l'éditeur de requêtes en ajoutant la `AmazonRDSDataFullAccess` politique, une stratégie prédéfinie Gestion des identités et des accès AWS (IAM), à cet utilisateur.

> **Note**
>
> Lorsque vous créez l’utilisateur IAM, assurez-vous d’utiliser le même nom d’utilisateur et le même mot de passe que ceux utilisés pour l’utilisateur de base de données, à savoir

le nom d'utilisateur et le mot de passe administrateur. Pour plus d'informations, consultez Création d'un utilisateur IAM dans votre Compte AWS dans le Guide de l'utilisateur Gestion des identités et des accès AWS.

Vous pouvez aussi créer une politique IAM qui accorde l'accès à l'éditeur de requête. Après la création de la politique, ajoutez-la à chaque utilisateur ayant besoin d'accéder à l'éditeur de requête.

La politique suivante fournit aux utilisateurs les autorisations minimales requises pour accéder à l'éditeur de requête.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "QueryEditor0",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:PutResourcePolicy",
                "secretsmanager:PutSecretValue",
                "secretsmanager:DeleteSecret",
                "secretsmanager:DescribeSecret",
                "secretsmanager:TagResource"
            ],
            "Resource": "arn:aws:secretsmanager:*:*:secret:rds-db-credentials/*"
        },
        {
            "Sid": "QueryEditor1",
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetRandomPassword",
                "tag:GetResources",
                "secretsmanager:CreateSecret",
                "secretsmanager:ListSecrets",
                "dbqms:CreateFavoriteQuery",
                "dbqms:DescribeFavoriteQueries",
                "dbqms:UpdateFavoriteQuery",
                "dbqms:DeleteFavoriteQueries",
```

```
                "dbqms:GetQueryString",
                "dbqms:CreateQueryHistory",
                "dbqms:UpdateQueryHistory",
                "dbqms:DeleteQueryHistory",
                "dbqms:DescribeQueryHistory",
                "rds-data:BatchExecuteStatement",
                "rds-data:BeginTransaction",
                "rds-data:CommitTransaction",
                "rds-data:ExecuteStatement",
                "rds-data:RollbackTransaction"
            ],
            "Resource": "*"
        }
    ]
}
```

Pour plus d'informations sur la création d'une stratégie IAM, consultez Création de stratégies IAM dans le Guide de l'utilisateur Gestion des identités et des accès AWS.

Pour plus d'informations sur l'ajout d'une stratégie IAM, consultez Ajout et suppression de stratégies IAM dans le Guide de l'utilisateur Gestion des identités et des accès AWS.

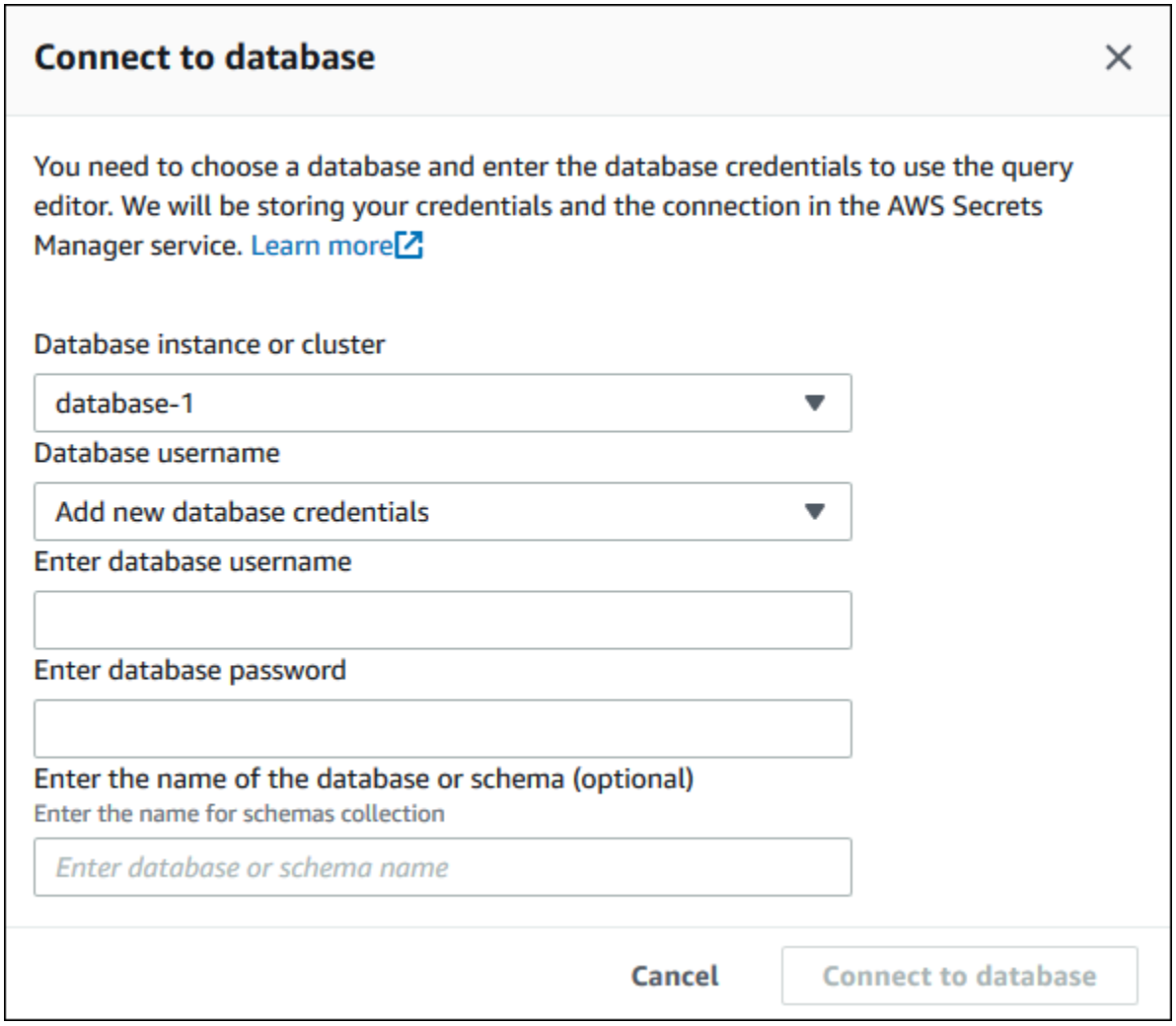
Exécution de requêtes dans l'éditeur de requête

Avec l'éditeur de requête, vous pouvez exécuter des requêtes SQL dans un cluster de bases de données Aurora. Le SQL que vous pouvez exécuter est limité par les contraintes de l'API de données. Pour plus d'informations, consultez the section called "Limitations".

Pour exécuter une requête dans l'éditeur de requête

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le coin supérieur droit duAWS Management Console, choisissez celui Région AWS dans lequel vous avez créé les clusters de base de données Aurora que vous souhaitez interroger.
3. Dans le panneau de navigation, choisissez Databases (Bases de données).
4. Choisissez le cluster de bases de données Aurora sur lequel vous souhaitez exécuter des requêtes SQL.

5. Pour Actions, choisissez Query (Requête). Si vous ne vous êtes pas connecté à la base de données au préalable, la page Connect to database (Se connecter à la base de données) s'ouvre.



6. Entrez les informations suivantes :

   a. Pour l'instance ou le cluster de bases de données, choisissez le cluster de bases de données Aurora pour lequel vous souhaitez exécuter des requêtes SQL.

   b. Pour Database username (Nom d'utilisateur de la base de données), choisissez le nom de l'utilisateur de la base de données avec lequel vous souhaitez vous connecter, ou choisissez Add new database credentials (Ajouter de nouvelles informations d'identification pour la base de données). Si vous choisissez Add new database credentials (Ajouter de nouvelles informations d'identification pour la base de données), entrez le nom d'utilisateur pour les nouvelles informations d'identification de la base de données dans le champ Enter database username (Entrer le nom d'utilisateur de la base de données).

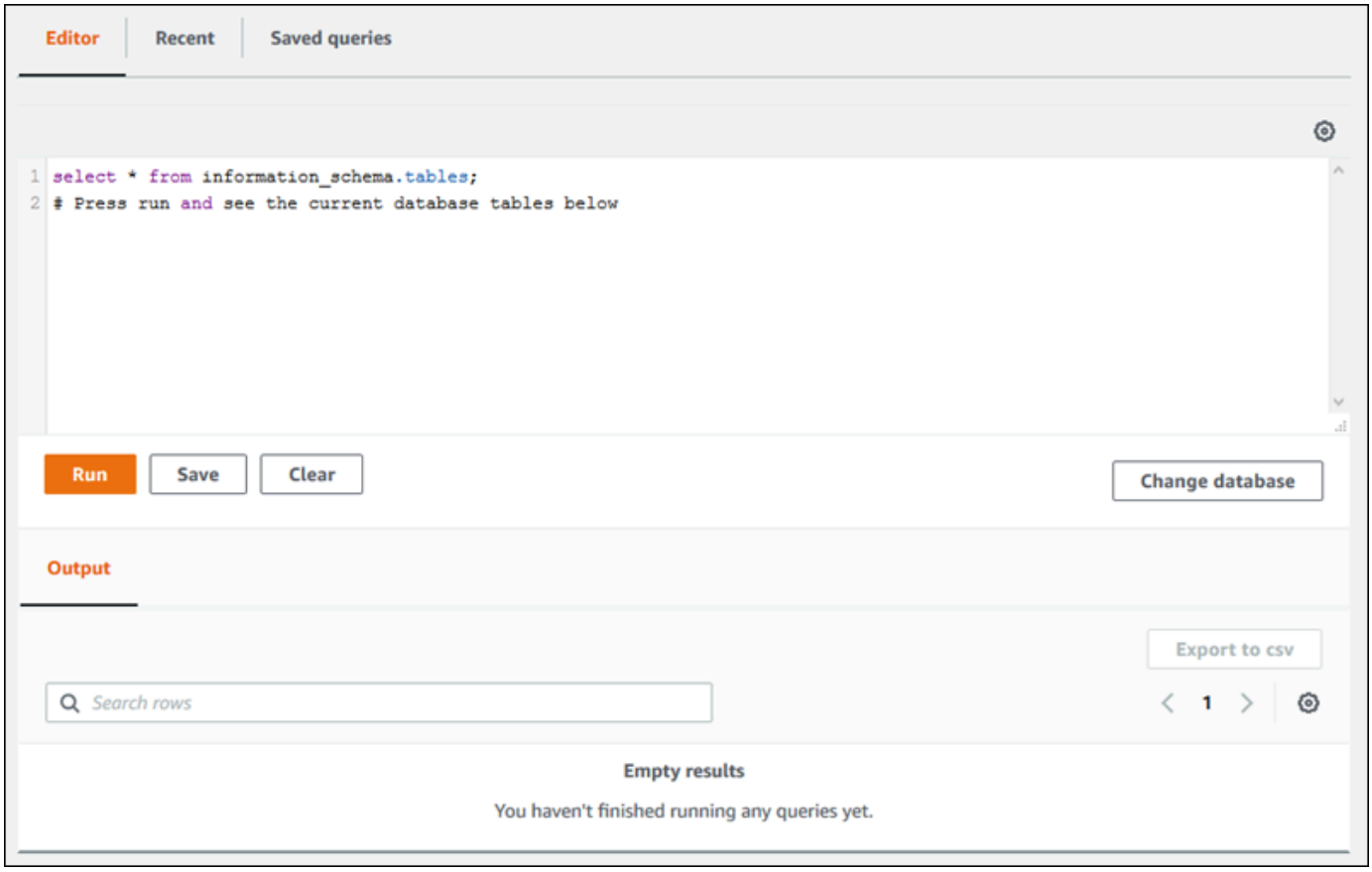
c. Pour Enter database password (Entrer le mot de passe de la base de données), saisissez le mot de passe de l'utilisateur de base de données que vous avez choisi.

d. Dans la dernière case, entrez le nom de la base de données ou du schéma que vous souhaitez utiliser pour le cluster de bases de données Aurora.

e. Choisissez Connect to database (Se connecter à la base de données).

> **Note**
>
> Si la connexion aboutit, vos informations de connexion et d'authentification sont stockées dans AWS Secrets Manager. Vous n'avez pas besoin d'entrer à nouveau les informations de connexion.

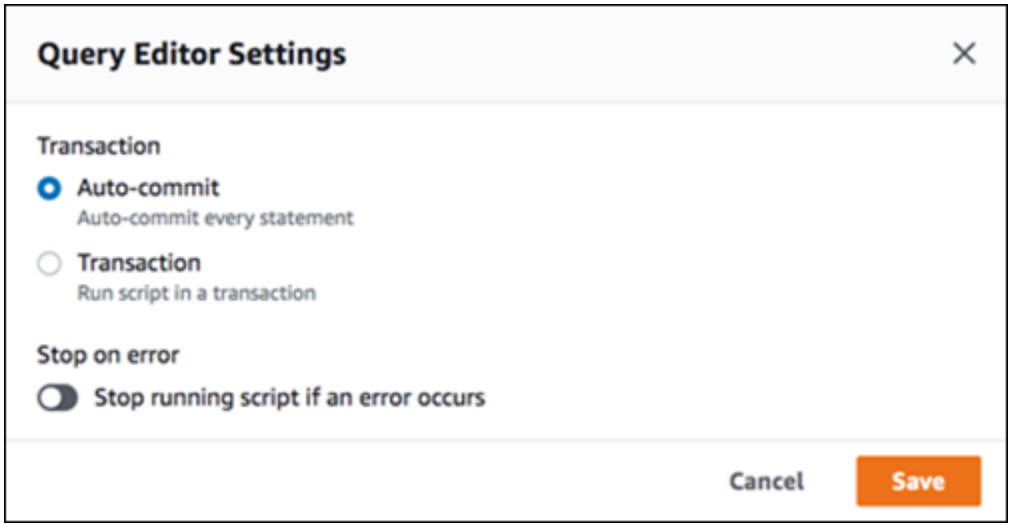
7. Dans l'éditeur de requête, entrez la requête SQL que vous souhaitez exécuter au niveau de la base de données.



Chaque instruction SQL peut être validée automatiquement, ou vous pouvez exécuter les instructions SQL dans un script dans le cadre d'une transaction. Pour contrôler ce

comportement, choisissez l'icône représentant un engrenage située au-dessus de la fenêtre de requête.

La fenêtre Query Editor Settings (Paramètres de l'éditeur de requête) apparaît.



Si vous choisissez Auto-commit, chaque instruction SQL est automatiquement validée. Si vous choisissez Transaction, vous pouvez exécuter un groupe d'instructions dans un script. Les instructions sont automatiquement validées à la fin du script, sauf si elles sont explicitement validées ou annulées avant. En outre, vous pouvez également choisir d'arrêter un script en cours si une erreur se produit en activant Stop on error (Arrêter en cas d'erreur).

> **Note**
>
> Dans un groupe d’instructions, les instructions en langage de définition de données (DDL) peuvent provoquer la validation d’anciennes instructions en langage de manipulation de données (DML). Vous pouvez également inclure des instructions `COMMIT` et `ROLLBACK` dans un groupe d’instructions au sein d’un script.

Une fois que vous avez terminé la configuration dans la fenêtre Query Editor Settings (Paramètres de l’éditeur de requête), choisissez Save (Enregistrer).

8. Choisissez Run (Exécuter) ou appuyez sur Ctrl + Entrée pour que l’éditeur de requête affiche les résultats de la requête.

   Une fois la requête exécutée, choisissez Save (Enregistrer) pour l’enregistrer dans Saved queries (Requêtes enregistrées).

   Choisissez Export to csv (Exporter au format csv) pour enregistrer les résultats de la requête dans une feuille de calcul.

Vous pouvez rechercher, modifier et exécuter de nouveau des anciennes requêtes. Pour ce faire, choisissez l’onglet Recent (Récent) ou Saved queries (Requêtes enregistrées), sélectionnez le texte de la requête, puis choisissez Run (Exécuter).

Pour changer de base de données, choisissez Change database (Changer de base de données).

Référence de l’API DBQMS (Database Query Metadata Service)

Le service `dbqms` (Database Query Metadata Service, service de métadonnées de requête de base de données) est un service interne uniquement. Il fournit vos requêtes récentes et enregistrées pour l’éditeur de requêtes sur l’AWS Management Console pour plusieurs services AWS, dont Amazon RDS.

Actions DBQMS prises en charge

- CreateFavoriteQuery
- CreateQueryHistory
- CreateTab

- DeleteFavoriteQueries
- DeleteQueryHistory
- DeleteTab
- DescribeFavoriteQueries
- DescribeQueryHistory
- DescribeTabs
- GetQueryString
- UpdateFavoriteQuery
- UpdateQueryHistory
- UpdateTab

CreateFavoriteQuery

Crée une nouvelle requête préférée. Chaque utilisateur peut créer jusqu'à 1 000 requêtes enregistrées. Cette limite est sujette à changement à l'avenir.

CreateQueryHistory

Enregistre une nouvelle entrée d'historique des requêtes.

CreateTab

Enregistre un nouvel onglet de requête. Chaque utilisateur peut créer jusqu'à 10 onglets de requête.

DeleteFavoriteQueries

Supprime une ou plusieurs requêtes enregistrées.

DeleteQueryHistory

Supprime les entrées d'historique des requêtes.

DeleteTab

Supprime les entrées de l'onglet requête.

DescribeFavoriteQueries

Répertorie les requêtes enregistrées créées par un utilisateur dans un compte donné.

DescribeQueryHistory

Liste les entrées de l’historique des requêtes.

DescribeTabs

Répertorie les onglets de requête créés par un utilisateur dans un compte donné.

GetQueryString

Récupérer le texte complet de la requête à partir d’un ID de requête.

UpdateFavoriteQuery

Met à jour la chaîne de requête, la description, le nom ou la date d’expiration.

UpdateQueryHistory

Met à jour le statut de l’historique des requêtes.

UpdateTab

Met à jour le nom de l’onglet de requête et la chaîne de requête.

Utilisation du machine learning Amazon Aurora

Avec le machine learning Amazon Aurora, vous pouvez intégrer votre cluster de bases de données Aurora à l'un des services de machine learning AWS suivants, en fonction de vos besoins. Ils prennent chacun en charge des cas d'utilisation propres au machine learning.

Amazon Bedrock

Amazon Bedrock est un service entièrement géré qui met à disposition les modèles de fondation de premier plan des entreprises d'IA à l'aide d'une API, ainsi que des outils de développement pour aider à créer et à mettre à l'échelle des applications d'IA générative. Avec Amazon Bedrock, vous payez pour exécuter l'inférence sur n'importe quel modèle de fondation tiers. La tarification dépend du volume de jetons d'entrée et de jetons de sortie, et du fait que vous ayez acheté ou non du débit provisionné pour le modèle. Pour plus d'informations, consultez Qu'est-ce qu'Amazon Bedrock ? dans le Guide de l'utilisateur Amazon Bedrock.

Amazon Comprehend

Amazon Comprehend est un service de traitement du langage naturel (NLP) utilisé pour extraire des informations à partir de documents. Avec Amazon Comprehend, vous pouvez déduire des sentiments en fonction du contenu des documents, en analysant les entités, les phrases clés, le langage et d'autres fonctions. Pour en savoir plus, consultez Qu'est-ce qu'Amazon Comprehend ? dans le Guide du développeur Amazon Comprehend.

SageMaker AI

Amazon SageMaker AI est un service de machine learning entièrement géré. Les scientifiques des données et les développeurs utilisent Amazon SageMaker AI pour créer, entraîner et tester des modèles de machine learning pour diverses tâches d'inférence, telles que la détection des fraudes et la recommandation de produits. Lorsqu'un modèle de machine learning est prêt à être utilisé en production, il peut être déployé dans l'environnement hébergé Amazon SageMaker AI. Pour plus d'informations, consultez Qu'est-ce qu'Amazon SageMaker AI ? dans le Guide du développeur Amazon SageMaker AI.

L'utilisation d'Amazon Comprehend avec votre cluster de bases de données Aurora nécessite moins de configuration préliminaire que l'utilisation de SageMaker AI. Si vous êtes novice en matière de machine learning AWS, nous vous recommandons de commencer par explorer Amazon Comprehend.

Rubriques

- Utilisation du machine learning Amazon Aurora avec Aurora MySQL
- Utilisation du machine learning Amazon Aurora avec Aurora PostgreSQL

Utilisation du machine learning Amazon Aurora avec Aurora MySQL

En utilisant le machine learning Amazon Aurora avec votre cluster de bases de données Aurora MySQL, vous pouvez utiliser Amazon Bedrock, Amazon Comprehend, ou Amazon SageMaker AI, selon vos besoins. Ils prennent chacun en charge différents cas d'utilisation de machine learning.

Table des matières

Exigences pour l’utilisation du machine learning Aurora avec Aurora MySQL

Les services de machine learning AWS sont des services gérés que vous configurez et exécutez dans leurs propres environnements de production. Le machine learning Aurora prend en charge l’intégration avec Amazon Bedrock, Amazon Comprehend et SageMaker AI. Avant d’essayer de configurer votre cluster de bases de données Aurora MySQL pour utiliser le machine learning Aurora, assurez-vous de comprendre les exigences et prérequis suivants.

- Les services de machine learning doivent être exécutés dans la même Région AWS que votre cluster de bases de données Aurora MySQL. Vous ne pouvez pas utiliser les services de machine learning à partir d’un cluster de bases de données Aurora MySQL situé dans une autre région.
- Si votre cluster de bases de données Aurora MySQL se trouve dans un autre cloud public virtuel (VPC) que celui de vos services Amazon Bedrock, Amazon Comprehend ou SageMaker AI, le groupe de sécurité du VPC doit autoriser les connexions sortantes vers le service de machine learning Aurora cible. Pour plus d’informations, consultez Contrôler le trafic vers vos ressources AWS à l’aide de groupes de sécurité dans le Guide de l’utilisateur Amazon VPC.
- Vous pouvez mettre à niveau un cluster Aurora s’exécutant sur une version plus ancienne d’Aurora MySQL vers une version plus récente si vous souhaitez utiliser le machine learning Aurora avec ce cluster. Pour plus d’informations, consultez Mises à jour du moteur de base de données pour Amazon Aurora MySQL.
- Votre cluster de bases de données Aurora MySQL doit utiliser un groupe de paramètres de cluster de bases de données personnalisé. À la fin du processus de configuration de chaque service de machine learning Aurora que vous souhaitez utiliser, vous ajoutez l’Amazon Resource Name (ARN) du rôle IAM associé qui a été créé pour le service. Nous vous recommandons de créer à l’avance un groupe de paramètres de cluster de bases de données personnalisé pour votre Aurora MySQL et de configurer votre cluster de bases de données Aurora MySQL pour qu’il soit prêt à être modifié à la fin du processus de configuration.

- Pour SageMaker AI :
  - Les composants de machine learning que vous souhaitez utiliser pour les inférences doivent être configurés et prêts à être utilisés. Au cours du processus de configuration de votre cluster de bases de données Aurora MySQL, vous devez disposer de l'ARN du point de terminaison SageMaker AI disponible. Les scientifiques des données de votre équipe sont probablement les mieux placés pour travailler avec SageMaker AI afin de préparer les modèles et effectuer les autres tâches de ce type. Pour commencer à utiliser Amazon SageMaker AI, consultez Commencer à utiliser Amazon SageMaker AI. Pour plus d'informations sur les inférences et les points de terminaison, consultez Inférence en temps réel.
  - Pour utiliser SageMaker AI avec vos propres données de formation, vous devez configurer un compartiment Amazon S3 dans le cadre de votre configuration Aurora MySQL pour le machine learning Aurora. Pour ce faire, vous devez suivre le même processus général que pour configurer l'intégration SageMaker AI. Pour un résumé de ce processus de configuration facultatif, consultez Configuration de votre cluster de bases de données Aurora MySQL pour utiliser Amazon S3 pour SageMaker AI (facultatif).
- Pour les bases de données globales Aurora, vous configurez les services de machine learning Aurora que vous souhaitez utiliser dans toutes les Régions AWS de votre base de données globale Aurora. Par exemple, si vous souhaitez utiliser le machine learning Aurora avec SageMaker AI pour votre base de données globale Aurora, procédez comme suit pour chaque cluster de bases de données Aurora MySQL de chaque Région AWS :
  - Configurez les services Amazon SageMaker AI avec les mêmes modèles de formation et points de terminaison SageMaker AI. Ils doivent également porter les mêmes noms.
  - Créez les rôles IAM comme indiqué dans Configuration de votre cluster de bases de données Aurora MySQL de façon à utiliser le machine learning Aurora.
  - Ajoutez l'ARN du rôle IAM au groupe de paramètres de cluster de bases de données personnalisé pour chaque cluster de bases de données Aurora MySQL dans chaque Région AWS.

  Ces tâches nécessitent que le machine learning Aurora soit disponible pour votre version d'Aurora MySQL dans toutes les Régions AWS qui composent votre base de données globale Aurora.

Disponibilité des régions et des versions

La disponibilité et la prise en charge des fonctions varient selon les versions spécifiques de chaque moteur de base de données Aurora, et selon les Régions AWS.

- Pour en savoir plus sur les régions et les versions disponibles d'Amazon Comprehend et d'Amazon SageMaker AI avec Aurora MySQL, consultez Machine Learning Aurora avec Aurora MySQL.
- Amazon Bedrock est uniquement pris en charge sur Aurora MySQL 3.06 ou version ultérieure.

  Pour plus d'informations sur les régions disponibles pour Amazon Bedrock, consultez Modèles pris en charge par Région AWS dans le Guide de l'utilisateur Amazon Bedrock.

Fonctions prises en charge et limitations du machine learning Aurora avec Aurora MySQL

Lorsque vous utilisez Aurora MySQL avec le machine learning Aurora, les limitations suivantes s'appliquent :

- L'extension de machine learning Aurora ne prend pas en charge les interfaces vectorielles.
- Les intégrations de machine learning Aurora ne sont pas prises en charge lorsqu'elles sont utilisées dans un déclencheur.
- Les fonctions de machine learning Aurora ne sont pas compatibles avec la réplication de journaux binaires (binlog).
  - Le paramètre `--binlog-format=STATEMENT` envoie une exception pour les appels vers des fonctions Machine Learning Aurora.
  - Les fonctions du machine learning Aurora sont non déterministes, et les fonctions stockées non déterministes ne sont pas compatibles avec ce format binlog.

  Pour plus d'informations, consultez Formats de journalisation binaire dans la documentation MySQL.
- Les fonctions stockées qui appellent des tables dont les colonnes sont toujours générées ne sont pas prises en charge. Cela s'applique à toutes les fonctions stockées dans Aurora MySQL. Pour en savoir plus sur ce type de colonne, consultez CREATE TABLE et colonnes générées dans la documentation MySQL.
- Les fonctions Amazon Bedrock ne prennent pas en charge `RETURNS JSON`. Vous pouvez utiliser `CONVERT` ou `CAST` afin d'effectuer une conversion de `TEXT` vers `JSON` si nécessaire.
- Amazon Bedrock ne prend pas en charge les demandes par lots.
- Aurora MySQL prend en charge tous les points de terminaison SageMaker AI pouvant lire et écrire un format CSV (valeurs séparées par des virgules) via un `ContentType` de `text/csv`. Ce format est accepté par les algorithmes SageMaker AI intégrés suivants :

- Linear Learner
- Random Cut Forest
- XGBoost

Pour en savoir plus sur ces algorithmes, consultez Choisir un algorithme dans le Guide du développeur Amazon SageMaker AI.

Configuration de votre cluster de bases de données Aurora MySQL de façon à utiliser le machine learning Aurora

Dans les rubriques suivantes, vous pouvez trouver des procédures de configuration distinctes pour chacun de ces services de machine learning Aurora.

Rubriques

- Configuration de votre cluster de bases de données Aurora MySQL pour utiliser Amazon Bedrock
- Configuration de votre cluster de bases de données Aurora MySQL pour utiliser Amazon Comprehend
- Configuration de votre cluster de bases de données Aurora MySQL pour utiliser SageMaker AI
  - Configuration de votre cluster de bases de données Aurora MySQL pour utiliser Amazon S3 pour SageMaker AI (facultatif)
- Autorisation d'accès aux utilisateurs de base de données pour le machine learning Aurora
  - Autorisation d'accès aux fonctions Amazon Bedrock
  - Autorisation d'accès aux fonctions Amazon Comprehend
  - Autorisation d'accès aux fonctions SageMaker AI

Configuration de votre cluster de bases de données Aurora MySQL pour utiliser Amazon Bedrock

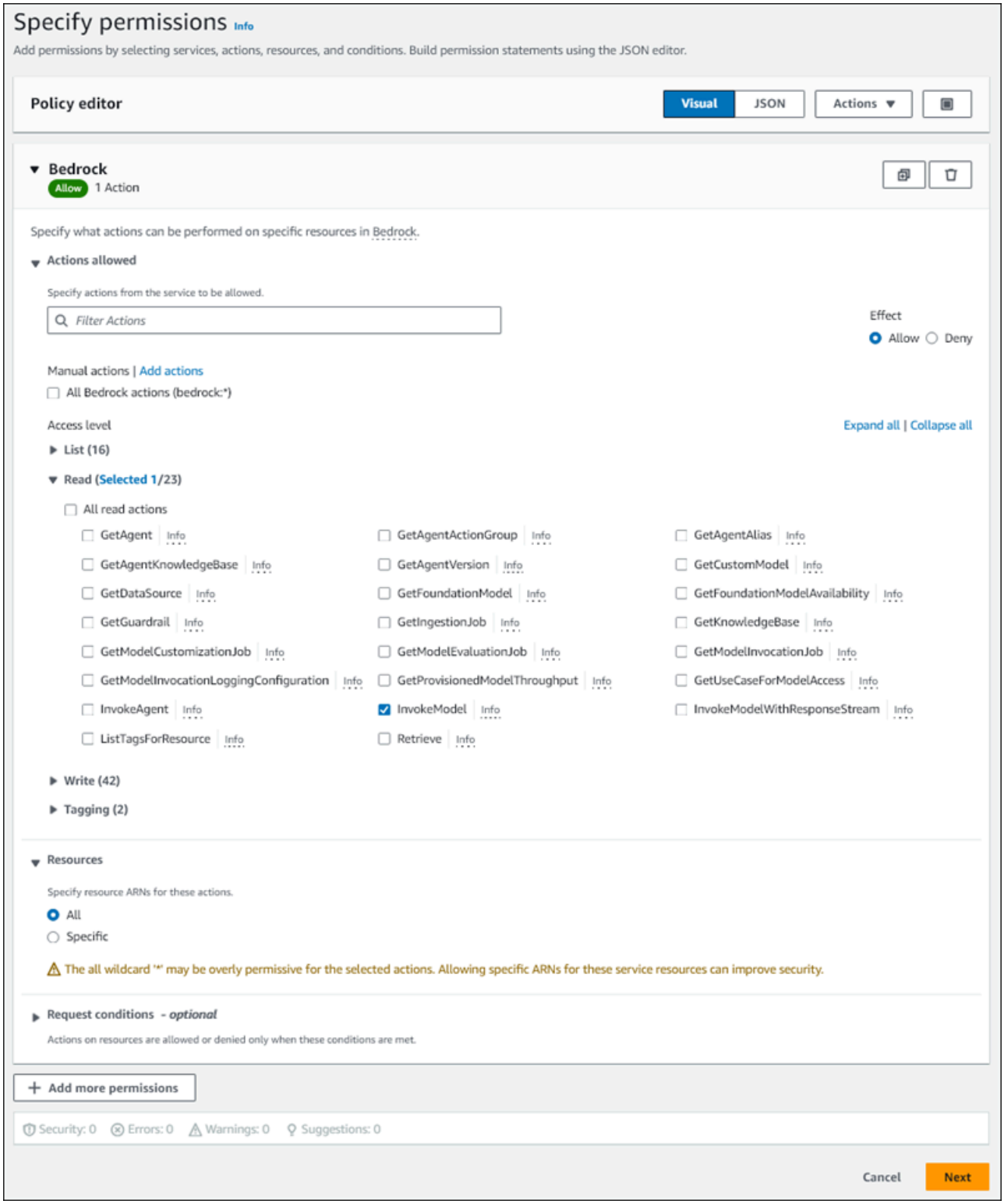
Le machine learning Aurora s'appuie sur des rôles et des politiques Gestion des identités et des accès AWS (IAM) pour permettre à votre cluster de bases de données Aurora MySQL d'accéder aux services Amazon Bedrock et de les utiliser. Les procédures suivantes créent une stratégie d'autorisation et un rôle IAM afin que votre cluster de bases de données puisse s'intégrer à Amazon Bedrock.

Pour créer la stratégie IAM

1. Connectez-vous à la AWS Management Console, puis ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation, choisissez Stratégies.
3. Choisissez Créer une stratégie.
4. Sur la page Spécifier les autorisations, choisissez Bedrock dans Sélectionner un service.

   Affichage des autorisations Amazon Bedrock.
5. Développez Lecture, puis sélectionnez InvokeModel.
6. Sélectionnez Tout dans Ressources.

   La page Spécifier les autorisations doit ressembler à la figure suivante.



7. Choisissez Suivant.
8. Sur la page Examiner et créer, entrez un nom pour votre stratégie, par exemple **`BedrockInvokeModel`**.

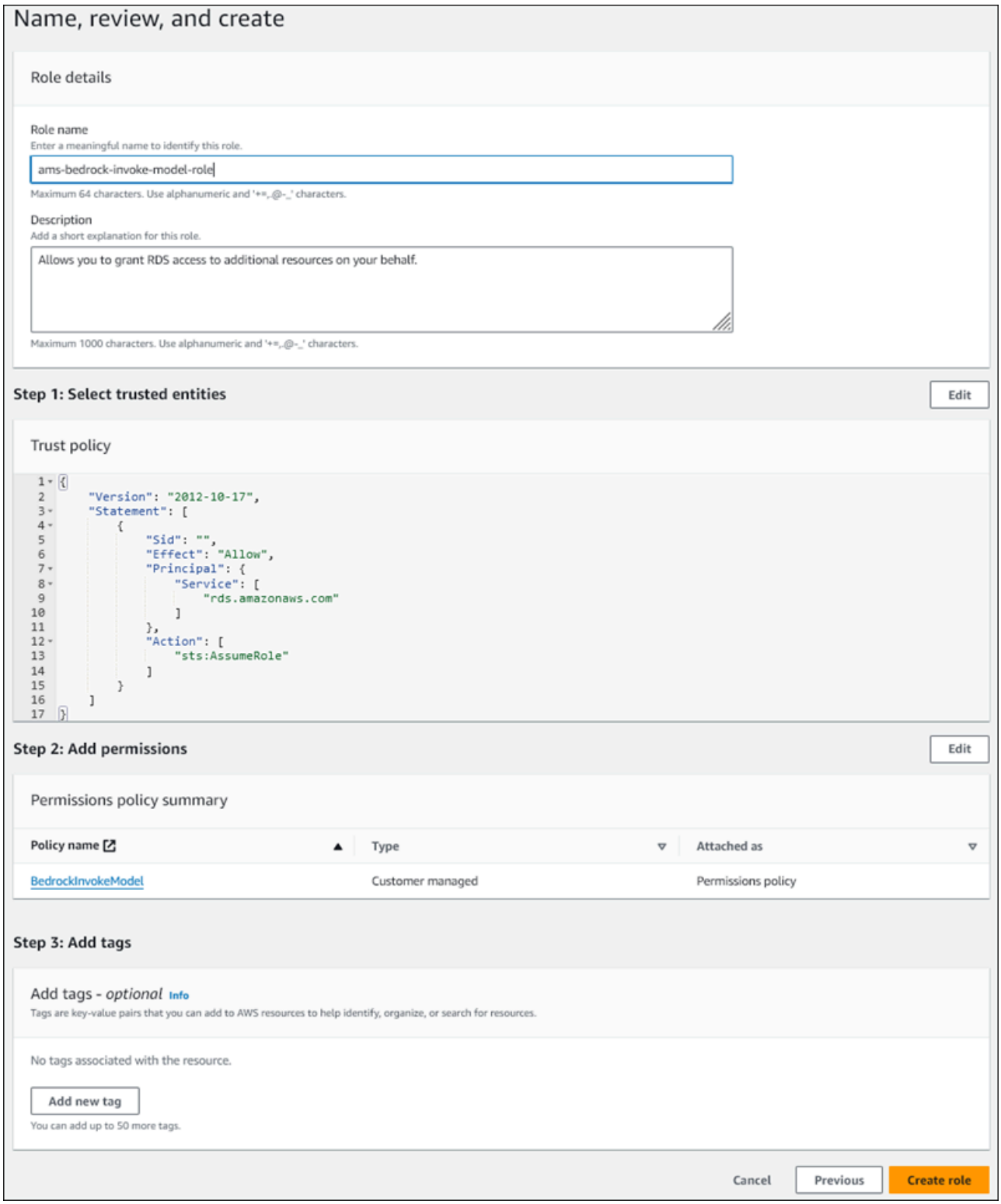
9. Examinez votre stratégie, puis choisissez Créer une stratégie.

Créez ensuite le rôle IAM qui utilise la stratégie d'autorisation d'Amazon Bedrock.

Pour créer le rôle IAM

1. Connectez-vous à la AWS Management Console, puis ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Rôles dans le panneau de navigation.
3. Sélectionnez Créer un rôle.
4. Sur la page Sélectionner une entité de confiance, choisissez RDS dans Cas d'utilisation.
5. Sélectionnez RDS - Ajouter un rôle à la base de données, puis cliquez sur Suivant.
6. Sur la page Ajouter des autorisations, sélectionnez la politique IAM que vous avez créée dans Politiques d'autorisations, puis choisissez Suivant.
7. Sur la page Nommer, vérifier et créer, entrez un nom pour votre rôle, par exemple **`ams-bedrock-invoke-model-role`**.

   Le rôle doit ressembler à la figure suivante.



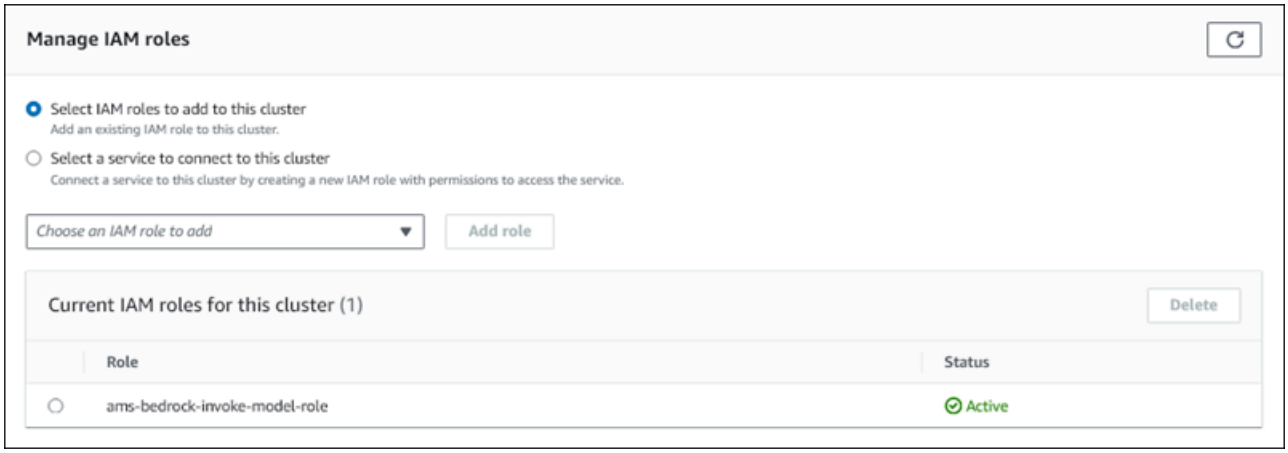
8. Passez en revue les informations de votre rôle, puis choisissez Créer un rôle.

Associez ensuite le rôle IAM d'Amazon Bedrock à votre cluster de bases de données.

Pour associer le rôle IAM à votre cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données dans le panneau de navigation.
3. Choisissez le cluster de bases de données Aurora MySQL que vous souhaitez connecter aux services Amazon Bedrock.
4. Choisissez l'onglet Connectivity & security (Connectivité et sécurité).
5. Dans la section Gérer les rôles IAM, choisissez Sélectionner les rôles IAM à ajouter à ce cluster.
6. Choisissez le rôle IAM que vous avez créé, puis choisissez Ajouter un rôle.

   Le rôle IAM est associé à votre cluster de bases de données, d'abord avec l'état En attente, puis Actif. Une fois le processus terminé, vous pouvez trouver le rôle dans la liste Current IAM roles for this cluster (Rôles IAM actuels pour ce cluster).



Vous devez ajouter l'ARN de ce rôle IAM au paramètre `aws_default_bedrock_role` dans le groupe de paramètres du cluster de bases de données associé au cluster Aurora MySQL. Si votre cluster de bases de données Aurora MySQL n'utilise pas de groupe de paramètres de cluster de bases de données personnalisé, vous devez en créer un à utiliser avec votre cluster de bases de données Aurora MySQL pour terminer l'intégration. Pour plus d'informations, consultez Groupes de paramètres de cluster de bases de données pour les clusters de bases de données Amazon Aurora.

Pour configurer le paramètre du cluster de bases de données

1. Dans la console Amazon RDS, ouvrez l'onglet Configuration de votre cluster de bases de données Aurora MySQL.

2. Localisez le groupe de paramètres du cluster de bases de données configuré pour votre cluster. Choisissez le lien pour ouvrir votre groupe de paramètres de cluster de bases de données personnalisé, puis sélectionnez Modifier.
3. Trouvez le paramètre `aws_default_bedrock_role` dans votre groupe de paramètres du cluster de bases de données personnalisé.
4. Dans le champ Valeur, saisissez l'ARN du rôle IAM.
5. Choisissez Save changes (Enregistrer les modifications) pour sauvegarder le paramètre.
6. Redémarrez l'instance principale de votre cluster de bases de données Aurora MySQL afin que ce paramètre prenne effet.

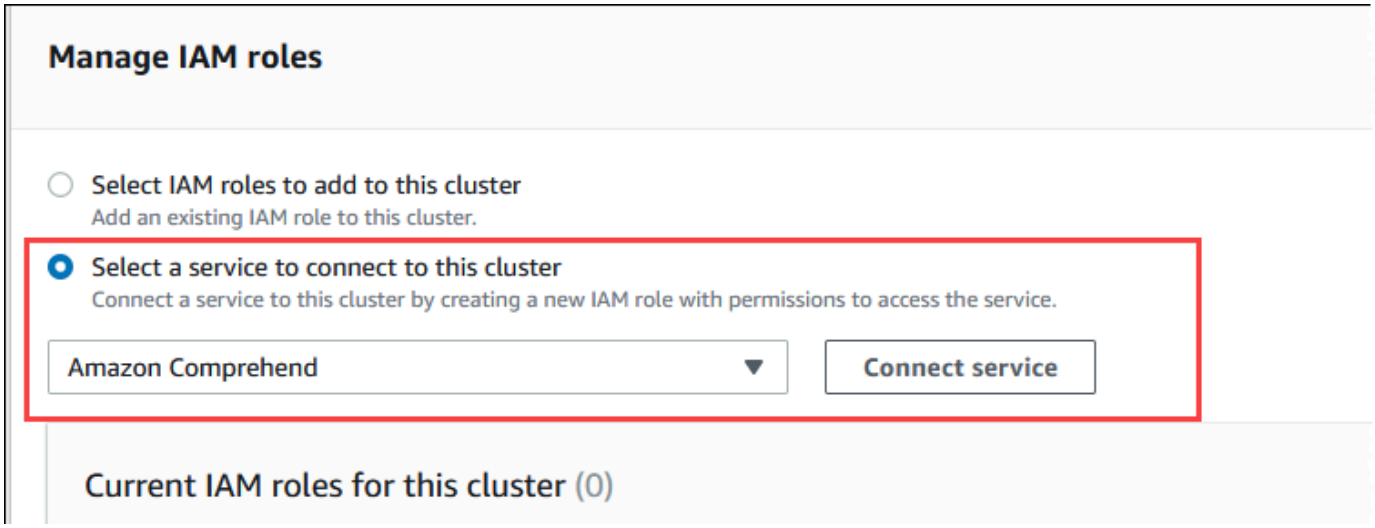
L'intégration IAM pour Amazon Bedrock est terminée. Poursuivez la configuration de votre cluster de bases de données Aurora MySQL pour qu'il fonctionne avec Amazon Bedrock en Autorisation d'accès aux utilisateurs de base de données pour le machine learning Aurora.

Configuration de votre cluster de bases de données Aurora MySQL pour utiliser Amazon Comprehend

Le machine learning Aurora s'appuie sur des rôles et des stratégies Gestion des identités et des accès AWS pour permettre à votre cluster de bases de données Aurora MySQL d'accéder aux services Amazon Comprehend et de les utiliser. La procédure suivante crée automatiquement un rôle et une stratégie IAM pour votre cluster afin qu'il puisse utiliser Amazon Comprehend.

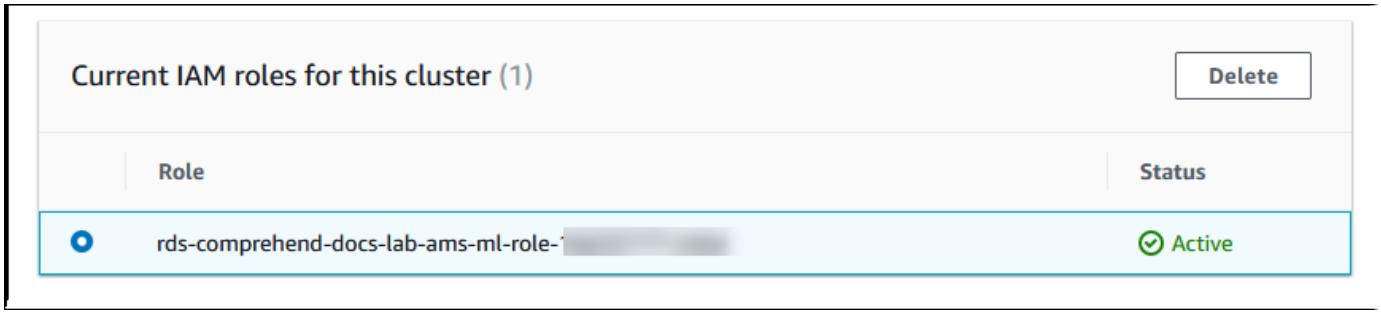
Configurer votre cluster de bases de données Aurora MySQL pour utiliser Amazon Comprehend

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données dans le panneau de navigation.
3. Choisissez le cluster de bases de données Aurora MySQL que vous souhaitez connecter aux services Amazon Comprehend.
4. Choisissez l'onglet Connectivity & security (Connectivité et sécurité).
5. Dans la section Gérer les rôles IAM, choisissez Sélectionner un service à connecter à ce cluster.
6. Choisissez Amazon Comprehend dans le menu, puis choisissez Connecter un service.



7. La boîte de dialogue Connect cluster to Amazon Comprehend (Connecter le cluster à Amazon Comprehend) ne nécessite aucune information supplémentaire. Toutefois, il se peut qu'un message vous avertisse que l'intégration entre Aurora et Amazon Comprehend est actuellement en version préliminaire. Assurez-vous de lire le message avant de continuer. Vous pouvez choisir Annuler si vous préférez ne pas continuer.
8. Choisissez Connect service (Connecter un service) pour terminer le processus d'intégration.

   Aurora crée le rôle IAM. Il crée également la stratégie qui permet au cluster de bases de données Aurora MySQL d'utiliser les services Amazon Comprehend et associe la stratégie au rôle. Une fois le processus terminé, vous pouvez trouver le rôle dans la liste Current IAM roles for this cluster (Rôles IAM actuels pour ce cluster), comme indiqué dans l'image suivante.



   Vous devez ajouter l'ARN de ce rôle IAM au paramètre `aws_default_comprehend_role` dans le groupe de paramètres du cluster de bases de données associé au cluster de bases de données Aurora MySQL. Si votre cluster de bases de données Aurora MySQL n'utilise pas de groupe de paramètres de cluster de bases de données personnalisé, vous devez en créer un à utiliser avec votre cluster de bases de données Aurora MySQL pour terminer l'intégration. Pour plus d'informations, consultez Groupes de paramètres de cluster de bases de données pour les clusters de bases de données Amazon Aurora.

Après avoir créé votre groupe de paramètres de cluster de bases de données personnalisé et l'avoir associé à votre cluster de bases de données Aurora MySQL, vous pouvez continuer à suivre ces étapes.

Si votre cluster utilise un groupe de paramètres de base de données personnalisé, procédez comme suit.

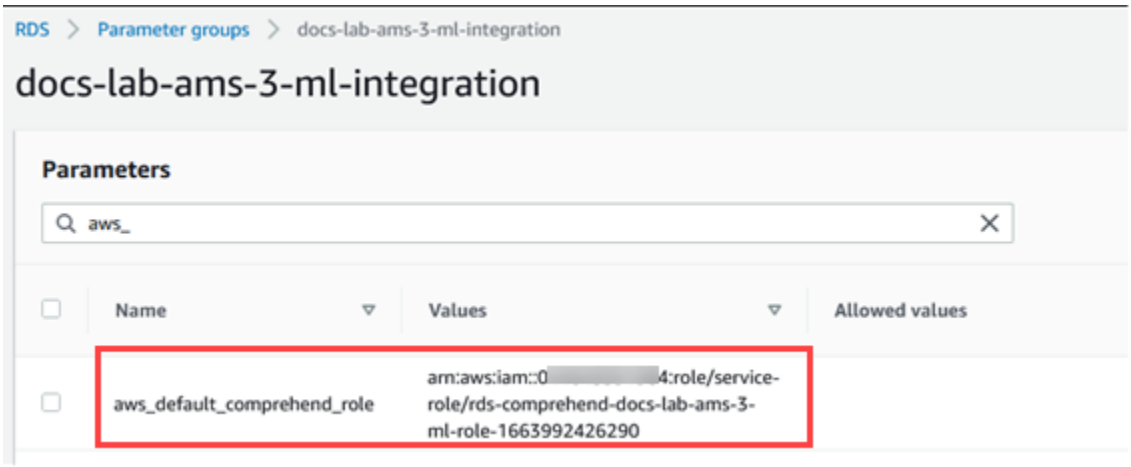
a. Dans la console Amazon RDS, ouvrez l'onglet Configuration de votre cluster de bases de données Aurora MySQL.

b. Localisez le groupe de paramètres du cluster de bases de données configuré pour votre cluster. Choisissez le lien pour ouvrir votre groupe de paramètres de cluster de bases de données personnalisé, puis sélectionnez Modifier.

c. Trouvez le paramètre `aws_default_comprehend_role` dans votre groupe de paramètres du cluster de bases de données personnalisé.

d. Dans le champ Valeur, saisissez l'ARN du rôle IAM.

e. Choisissez Save changes (Enregistrer les modifications) pour sauvegarder le paramètre. Dans l'image suivante, vous pouvez voir un exemple.

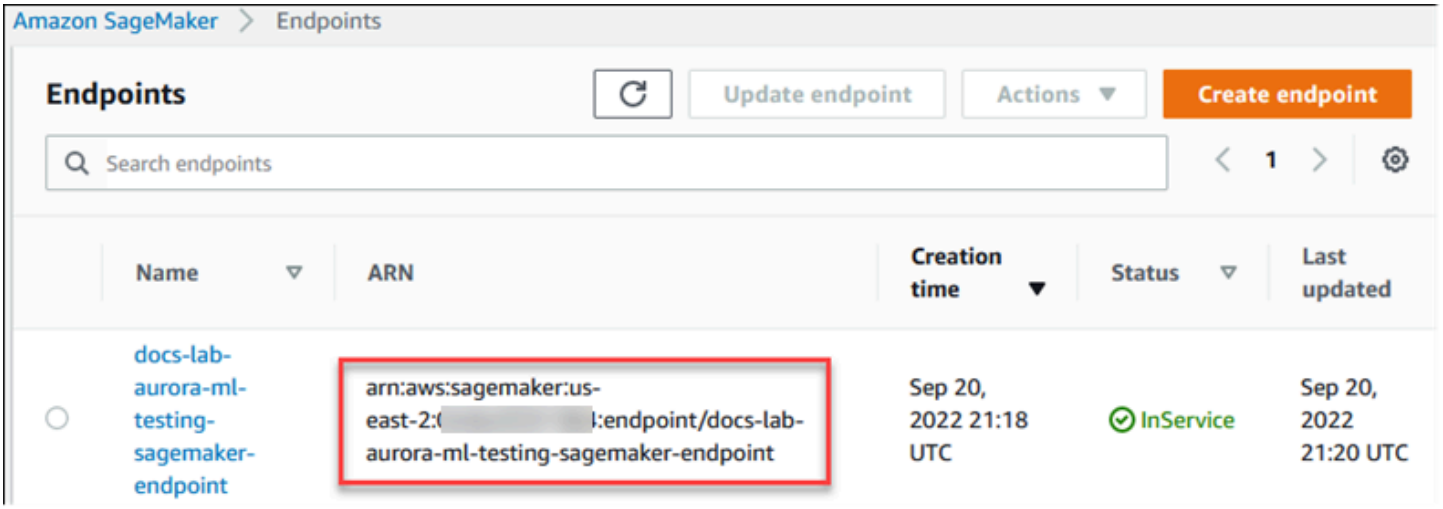


Redémarrez l'instance principale de votre cluster de bases de données Aurora MySQL afin que ce paramètre prenne effet.

L'intégration IAM pour Amazon Comprehend est terminée. Continuez à configurer votre cluster de bases de données Aurora MySQL pour qu'il fonctionne avec Amazon Comprehend en accordant l'accès aux utilisateurs de base de données appropriés.

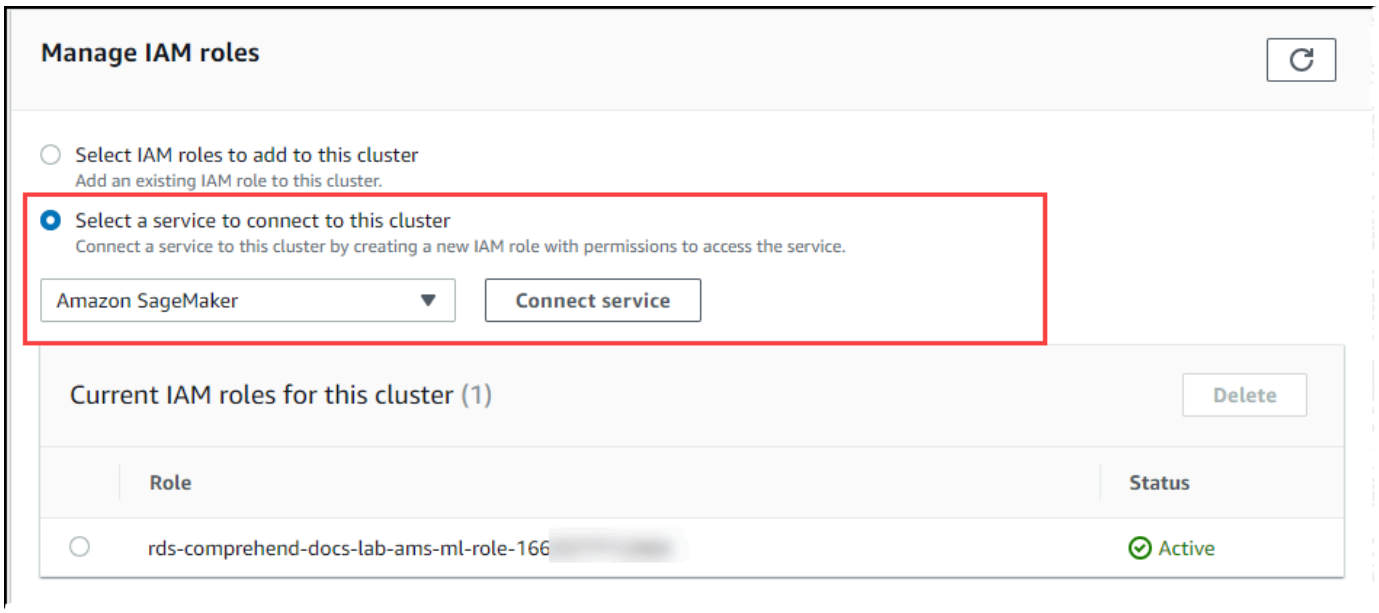
Configuration de votre cluster de bases de données Aurora MySQL pour utiliser SageMaker AI

La procédure suivante crée automatiquement un rôle et une stratégie IAM pour votre cluster de bases de données Aurora MySQL afin qu'il puisse utiliser SageMaker AI. Avant d'essayer de suivre cette procédure, assurez-vous que le point de terminaison SageMaker AI est disponible afin de pouvoir le saisir si nécessaire. En général, les scientifiques des données de votre équipe se chargent de produire un point de terminaison que vous pouvez utiliser à partir de votre cluster de bases de données Aurora MySQL. Vous pouvez trouver ces points de terminaison dans la console SageMaker AI. Dans le panneau de navigation, ouvrez Inference (Inférence), puis choisissez Endpoints (Points de terminaison). Dans l'image suivante, vous pouvez voir un exemple.

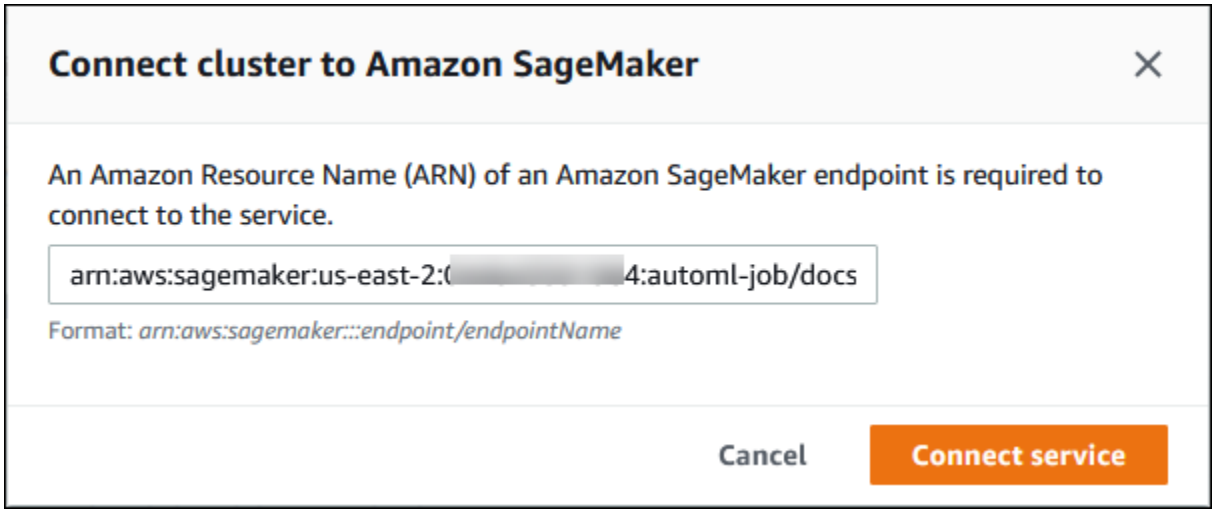


Configurer votre cluster de bases de données Aurora MySQL pour utiliser SageMaker AI

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Choisissez Bases de données dans le menu de navigation Amazon RDS, puis le cluster de bases de données Aurora MySQL que vous souhaitez connecter aux services SageMaker AI.
3. Choisissez l'onglet Connectivity & security (Connectivité et sécurité).
4. Choisissez Select a service to connect to this cluster (Sélectionner un service à connecter à ce cluster) dans la section Manage IAM roles (Gérer les rôles IAM). Choisissez SageMaker Ai dans le sélecteur.



5. Choisissez Connect service (Connecter un service).
6. Dans la boîte de dialogue Connecter le cluster à SageMaker AI, saisissez l'ARN du point de terminaison SageMaker AI.



7. Aurora crée le rôle IAM. Il crée également la stratégie qui permet au cluster de bases de données Aurora MySQL d'utiliser les services SageMaker AI et associe la stratégie au rôle. Une fois le processus terminé, vous pouvez trouver le rôle dans la liste Current IAM roles for this cluster (Rôles IAM actuels pour ce cluster).
8. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
9. Choisissez Roles (Rôles) dans la section Gestion des accès du menu de navigation Gestion des identités et des accès AWS.
10. Trouvez le rôle parmi ceux qui figurent dans la liste. Son nom utilise le modèle suivant.

```
rds-sagemaker-your-cluster-name-role-auto-generated-digits
```

11. Ouvrez la page Résumé du rôle et recherchez l'ARN. Notez l'ARN ou copiez-le à l'aide du widget de copie.
12. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
13. Choisissez votre cluster de bases de données Aurora MySQL, puis son onglet Configuration.
14. Localisez le groupe de paramètres du cluster de bases de données et choisissez le lien pour ouvrir votre groupe de paramètres du cluster de bases de données personnalisé. Recherchez le paramètre `aws_default_sagemaker_role` et saisissez l'ARN du rôle IAM dans le champ Valeur, puis enregistrez le paramètre.
15. Redémarrez l'instance principale de votre cluster de bases de données Aurora MySQL afin que ce paramètre prenne effet.

La configuration IAM est maintenant terminée. Continuez à configurer votre cluster de bases de données Aurora MySQL pour qu'il fonctionne avec SageMaker AI en accordant l'accès aux utilisateurs de base de données appropriés.

Si vous souhaitez utiliser vos modèles SageMaker AI à des fins de formation plutôt que d'utiliser des composants SageMaker AI prédéfinis, vous devez également ajouter le compartiment Amazon S3 à votre cluster de bases de données Aurora MySQL, comme indiqué dans Configuration de votre cluster de bases de données Aurora MySQL pour utiliser Amazon S3 pour SageMaker AI (facultatif) ci-dessous.

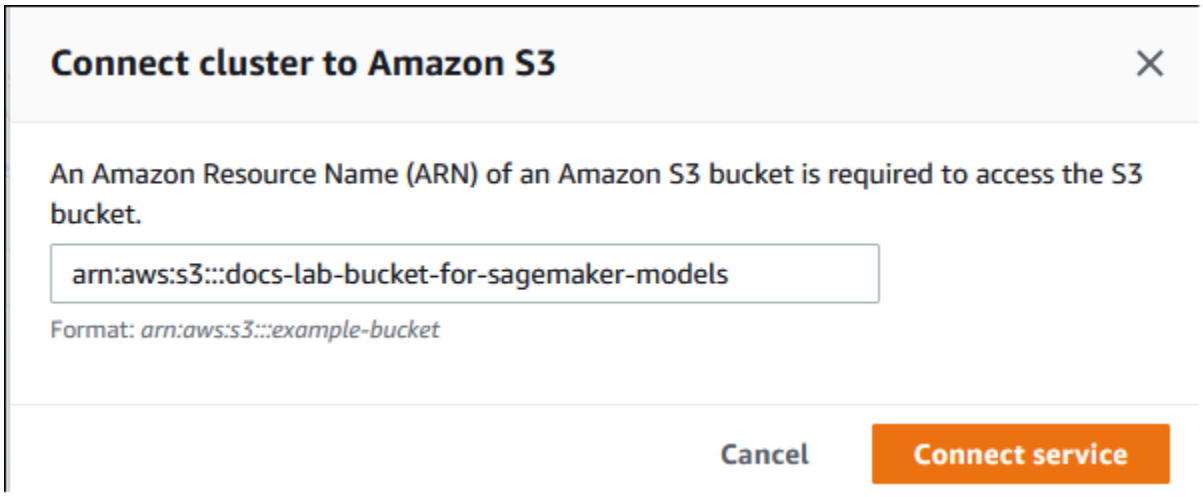
Configuration de votre cluster de bases de données Aurora MySQL pour utiliser Amazon S3 pour SageMaker AI (facultatif)

Pour utiliser SageMaker AI avec vos propres modèles plutôt que d'utiliser les composants prédéfinis fournis par SageMaker AI, vous devez configurer un compartiment Amazon S3 destiné au cluster de bases de données Aurora MySQL. Pour plus d'informations sur la création d'un compartiment Amazon S3, consultez Créer un compartiment dans le Guide de l'utilisateur Amazon Simple Storage Service.

Configurer votre cluster de bases de données Aurora MySQL pour utiliser un compartiment Amazon S3 pour SageMaker AI

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

2. Choisissez Bases de données dans le menu de navigation Amazon RDS, puis le cluster de bases de données Aurora MySQL que vous souhaitez connecter aux services SageMaker AI.
3. Choisissez l'onglet Connectivity & security (Connectivité et sécurité).
4. Choisissez Select a service to connect to this cluster (Sélectionner un service à connecter à ce cluster) dans la section Manage IAM roles (Gérer les rôles IAM). Choisissez Amazon S3 dans le sélecteur.
5. Choisissez Connect service (Connecter un service).
6. Dans la boîte de dialogue Connecter un cluster à Amazon S3, saisissez l'ARN du compartiment Amazon S3, comme indiqué dans l'image suivante.



7. Choisissez Connect service (Connecter un service) pour terminer ce processus.

Pour plus d'informations sur l'utilisation de compartiments Amazon S3 avec SageMaker AI, consultez Spécifier un compartiment S3 pour télécharger des jeux de données d'entraînement et stocker des données de sortie dans le Guide du développeur Amazon SageMaker AI. Pour en savoir plus sur l'utilisation de SageMaker AI, consultez Démarrer avec les instances de bloc-notes Amazon SageMaker AI dans le Guide du développeur Amazon SageMaker AI.

Autorisation d'accès aux utilisateurs de base de données pour le machine learning Aurora

Les utilisateurs de la base de données doivent être autorisés à invoquer ces fonctions de machine learning Aurora. La manière dont vous accordez l'autorisation dépend de la version de MySQL que vous utilisez pour votre cluster de bases de données Aurora MySQL, comme indiqué ci-dessous. La

manière de procéder dépend de la version de MySQL utilisée par votre cluster de bases de données Aurora MySQL.

- Pour Aurora MySQL version 3 (compatible avec MySQL 8.0), les utilisateurs de base de données doivent disposer du rôle de base de données approprié. Pour plus d'informations, consultez Utilisation de rôles dans le Manuel de référence MySQL 8.0.
- Pour Aurora MySQL version 2 (compatible avec MySQL 5.7), les utilisateurs de base de données disposent de privilèges. Pour plus d'informations, consultez Contrôle d'accès et gestion des comptes dans le Manuel de référence MySQL 5.7.

Le tableau suivant indique les rôles et les privilèges dont les utilisateurs de base de données ont besoin pour utiliser les fonctions de machine learning.

Aurora MySQL version 3 (rôle)	Aurora MySQL version 2 (privilège)
AWS_BEDROCK_ACCESS	–
AWS_COMPREHEND_ACCESS	INVOKE COMPREHEND
AWS_SAGEMAKER_ACCESS	INVOKE SAGEMAKER

Autorisation d'accès aux fonctions Amazon Bedrock

Pour permettre aux utilisateurs de base de données d'accéder aux fonctions d'Amazon Bedrock, utilisez l'instruction SQL suivante :

```
GRANT AWS_BEDROCK_ACCESS TO user@domain-or-ip-address;
```

Les utilisateurs de base de données doivent également disposer d'autorisations EXECUTE pour les fonctions que vous créez pour travailler avec Amazon Bedrock :

```
GRANT EXECUTE ON FUNCTION database_name.function_name TO user@domain-or-ip-address;
```

Enfin, les rôles des utilisateurs de base de données doivent être définis sur AWS_BEDROCK_ACCESS :

```
SET ROLE AWS_BEDROCK_ACCESS;
```

Les fonctions Amazon Bedrock peuvent désormais être utilisées.

Autorisation d'accès aux fonctions Amazon Comprehend

Pour permettre aux utilisateurs de la base de données d'accéder aux fonctions d'Amazon Comprehend, utilisez l'instruction appropriée à votre version d'Aurora MySQL.

- Aurora MySQL version 3 (compatible avec MySQL 8.0)

```
GRANT AWS_COMPREHEND_ACCESS TO user@domain-or-ip-address;
```

- Aurora MySQL version 2 (compatible avec MySQL 5.7)

```
GRANT INVOKE COMPREHEND ON *.* TO user@domain-or-ip-address;
```

Les fonctions Amazon Comprehend peuvent désormais être utilisées. Pour obtenir des exemples d'utilisation, consultez Utiliser Amazon Comprehend avec votre cluster de bases de données Aurora MySQL.

Autorisation d'accès aux fonctions SageMaker AI

Pour permettre aux utilisateurs de base de données d'accéder aux fonctions de SageMaker AI, utilisez l'instruction appropriée à votre version d'Aurora MySQL.

- Aurora MySQL version 3 (compatible avec MySQL 8.0)

```
GRANT AWS_SAGEMAKER_ACCESS TO user@domain-or-ip-address;
```

- Aurora MySQL version 2 (compatible avec MySQL 5.7)

```
GRANT INVOKE SAGEMAKER ON *.* TO user@domain-or-ip-address;
```

Les utilisateurs de base de données doivent également disposer d'autorisations `EXECUTE` pour les fonctions que vous créez pour travailler avec SageMaker AI. Supposons que vous ayez créé deux fonctions, `db1.anomoly_score` et `db2.company_forecasts`, pour invoquer les services de votre point de terminaison SageMaker AI. Vous accordez des privilèges d'exécution comme indiqué dans l'exemple ci-dessous.

```
GRANT EXECUTE ON FUNCTION db1.anomaly_score TO user1@domain-or-ip-address1;
```

```
GRANT EXECUTE ON FUNCTION db2.company_forecasts TO user2@domain-or-ip-address2;
```

Les fonctions de SageMaker AI peuvent désormais être utilisées. Pour obtenir des exemples d'utilisation, consultez Utilisation de SageMaker AI avec votre cluster de bases de données Aurora MySQL.

Utilisation d'Amazon Bedrock avec votre cluster de bases de données Aurora MySQL

Pour utiliser Amazon Bedrock, vous devez créer une fonction définie par l'utilisateur (UDF) dans votre base de données Aurora MySQL qui invoque un modèle. Pour plus d'informations, consultez Modèles pris en charge dans Amazon Bedrock dans le Guide de l'utilisateur Amazon Bedrock.

Un UDF utilise la syntaxe suivante :

```
CREATE FUNCTION function_name (argument type)
        [DEFINER = user]
    RETURNS mysql_data_type
    [SQL SECURITY {DEFINER | INVOKER}]
    ALIAS AWS_BEDROCK_INVOKE_MODEL
    MODEL ID 'model_id'
    [CONTENT_TYPE 'content_type']
    [ACCEPT 'content_type']
    [TIMEOUT_MS timeout_in_milliseconds];
```

- Les fonctions Amazon Bedrock ne prennent pas en charge `RETURNS JSON`. Vous pouvez utiliser `CONVERT` ou `CAST` afin d'effectuer une conversion de `TEXT` vers `JSON` si nécessaire.
- Si vous ne spécifiez pas la valeur `CONTENT_TYPE` ou `ACCEPT`, la valeur par défaut est `application/json`.
- Si vous ne spécifiez pas la valeur `TIMEOUT_MS`, la valeur `aurora_ml_inference_timeout` est utilisée.

Par exemple, l'UDF suivant invoque le modèle Amazon Titan Text Express :

```
CREATE FUNCTION invoke_titan (request_body TEXT)
    RETURNS TEXT
    ALIAS AWS_BEDROCK_INVOKE_MODEL
    MODEL ID 'amazon.titan-text-express-v1'
    CONTENT_TYPE 'application/json'
```

```
    ACCEPT 'application/json';
```

Pour autoriser un utilisateur de base de données à utiliser cette fonction, utilisez la commande SQL suivante :

```
GRANT EXECUTE ON FUNCTION database_name.invoke_titan TO user@domain-or-ip-address;
```

L'utilisateur peut ensuite appeler `invoke_titan` comme n'importe quelle autre fonction, comme illustré dans l'exemple suivant. Assurez-vous de formater le corps de la demande conformément aux modèles de texte Amazon Titan.

```
CREATE TABLE prompts (request varchar(1024));
INSERT INTO prompts VALUES (
'{
    "inputText": "Generate synthetic data for daily product sales in various categories
 - include row number, product name, category, date of sale and price. Produce output
 in JSON format. Count records and ensure there are no more than 5.",
    "textGenerationConfig": {
        "maxTokenCount": 1024,
        "stopSequences": [],
        "temperature":0,
        "topP":1
    }
}');

SELECT invoke_titan(request) FROM prompts;

{"inputTextTokenCount":44,"results":[{"tokenCount":296,"outputText":"
```tabular-data-json
{
    "rows": [
        {
            "Row Number": "1",
            "Product Name": "T-Shirt",
            "Category": "Clothing",
            "Date of Sale": "2024-01-01",
            "Price": "$20"
        },
        {
            "Row Number": "2",
            "Product Name": "Jeans",
            "Category": "Clothing",
```
```

```
            "Date of Sale": "2024-01-02",
            "Price": "$30"
        },
        {
            "Row Number": "3",
            "Product Name": "Hat",
            "Category": "Accessories",
            "Date of Sale": "2024-01-03",
            "Price": "$15"
        },
        {
            "Row Number": "4",
            "Product Name": "Watch",
            "Category": "Accessories",
            "Date of Sale": "2024-01-04",
            "Price": "$40"
        },
        {
            "Row Number": "5",
            "Product Name": "Phone Case",
            "Category": "Accessories",
            "Date of Sale": "2024-01-05",
            "Price": "$25"
        }
    ]
}
```","completionReason":"FINISH"}]}

Pour les autres modèles que vous utilisez, veillez à appliquer le format approprié. Pour plus d'informations, consultez Paramètres d'inférence pour les modèles de fondation dans le Guide de l'utilisateur Amazon Bedrock.

Utiliser Amazon Comprehend avec votre cluster de bases de données Aurora MySQL

Pour Aurora MySQL, le machine learning Aurora fournit les deux fonctions intégrées suivantes pour travailler avec Amazon Comprehend et vos données texte. Vous fournissez le texte à analyser (`input_data`) et vous spécifiez la langue (`language_code`).

aws_comprehend_detect_sentiment

Cette fonction identifie le texte comme ayant une posture émotionnelle positive, négative, neutre ou mixte. La documentation de référence de cette fonction est la suivante.

```
aws_comprehend_detect_sentiment(
  input_text,
  language_code
  [,max_batch_size]
)
```

Pour en savoir plus, consultez Sentiment dans le Guide du développeur Amazon Comprehend.

aws_comprehend_detect_sentiment_confidence

Cette fonction mesure le niveau de confiance du sentiment détecté pour un texte donné. Elle renvoie une valeur (type `double`) qui indique la confiance du sentiment attribué par la fonction aws_comprehend_detect_sentiment au texte. La confiance est une mesure statistique comprise entre 0 et 1. Plus le niveau de confiance est élevé, plus vous pouvez donner de poids au résultat. Voici un résumé de la documentation de la fonction.

```
aws_comprehend_detect_sentiment_confidence(
  input_text,
  language_code
  [,max_batch_size]
)
```

Pour les deux fonctions (aws_comprehend_detect_sentiment_confidence, aws_comprehend_detect_sentiment), `max_batch_size` utilise une valeur par défaut de 25 si aucune valeur n'est spécifiée. La taille de lot doit toujours être supérieure à 0. Le paramètre `max_batch_size` vous permet d'ajuster les performances des appels de fonction Amazon Comprehend. Une taille de lot importante est l'assurance de performances plus rapides pour une meilleure utilisation de la mémoire sur le cluster de bases de données Aurora MySQL. Pour plus d'informations, consultez Considérations sur les performances du machine learning Aurora avec Aurora MySQL.

Pour obtenir des informations sur les paramètres et retourner des types pour les fonctions de détection des sentiments dans Amazon Comprehend, consultez DetectSentiment

Example Exemple : une requête simple utilisant les fonctions Amazon Comprehend

Voici un exemple de requête simple qui invoque ces deux fonctions pour voir dans quelle mesure vos clients sont satisfaits de votre équipe d'assistance. Supposons que vous disposiez d'une table de base de données (`support`) qui enregistre les commentaires des clients après chaque demande d'aide. Cet exemple de requête applique les deux fonctions intégrées au texte de la colonne `feedback` du tableau et génère les résultats. Les valeurs de confiance renvoyées par la fonction sont des valeurs de confiance doubles comprises entre 0,0 et 1,0. Pour une sortie plus lisible, cette requête arrondit les résultats à 6 décimales. Pour faciliter les comparaisons, cette requête trie également les résultats par ordre décroissant, en commençant par le résultat présentant le degré de confiance le plus élevé.

```
SELECT feedback AS 'Customer feedback',
    aws_comprehend_detect_sentiment(feedback, 'en') AS Sentiment,
    ROUND(aws_comprehend_detect_sentiment_confidence(feedback, 'en'), 6)
    AS Confidence FROM support
    ORDER BY Confidence DESC;
+-----------------------------------------------------------+-----------+------------+
| Customer feedback                                         | Sentiment | Confidence |
+-----------------------------------------------------------+-----------+------------+
| Thank you for the excellent customer support!             | POSITIVE  |   0.999771 |
| The latest version of this product stinks!                | NEGATIVE  |   0.999184 |
| Your support team is just awesome! I am blown away.       | POSITIVE  |   0.997774 |
| Your product is too complex, but your support is great.   | MIXED     |   0.957958 |
| Your support tech helped me in fifteen minutes.           | POSITIVE  |   0.949491 |
| My problem was never resolved!                            | NEGATIVE  |   0.920644 |
| When will the new version of this product be released?    | NEUTRAL   |   0.902706 |
| I cannot stand that chatbot.                              | NEGATIVE  |   0.895219 |
| Your support tech talked down to me.                      | NEGATIVE  |   0.868598 |
| It took me way too long to get a real person.             | NEGATIVE  |   0.481805 |
+-----------------------------------------------------------+-----------+------------+
 10 rows in set (0.1898 sec)
```

Example Exemple : détermination du sentiment moyen pour un texte supérieur à un niveau de confiance spécifique

Généralement, une requête Amazon Comprehend recherche les lignes dans lesquelles le sentiment représente une certaine valeur, avec un niveau de confiance supérieur à un certain nombre. Par exemple, la requête suivante illustre comment vous pouvez déterminer le sentiment moyen des documents dans votre base de données. La requête prend uniquement en compte les documents dans lesquels la confiance de l'évaluation est de 80 % minimum.

```
SELECT AVG(CASE aws_comprehend_detect_sentiment(productTable.document, 'en')
  WHEN 'POSITIVE' THEN 1.0
  WHEN 'NEGATIVE' THEN -1.0
  ELSE 0.0 END) AS avg_sentiment, COUNT(*) AS total
FROM productTable
WHERE productTable.productCode = 1302 AND
    aws_comprehend_detect_sentiment_confidence(productTable.document, 'en') >= 0.80;
```

Utilisation de SageMaker AI avec votre cluster de bases de données Aurora MySQL

Pour utiliser les fonctionnalités de SageMaker AI depuis votre cluster de bases de données Aurora MySQL, vous devez créer des fonctions stockées qui intègrent vos appels au point de terminaison SageMaker AI et à ses fonctions d'inférence. Pour ce faire, vous utilisez `CREATE FUNCTION` de MySQL de la même manière que vous le faites pour les autres tâches de traitement sur votre cluster de bases de données Aurora MySQL.

Pour utiliser des modèles déployés dans SageMaker AI pour l'inférence, vous créez des fonctions définies par l'utilisateur à l'aide d'instructions Langage de définition de données (DDL) MySQL pour les fonctions stockées. Chaque fonction stockée représente le point de terminaison SageMaker AI hébergeant le modèle. Lorsque vous définissez une telle fonction, vous spécifiez les paramètres d'entrée dans le modèle, le point de terminaison SageMaker AI spécifique à invoquer et le type de retour. La fonction retourne l'inférence calculée par le point de terminaison SageMaker AI après avoir appliqué le modèle aux paramètres d'entrée.

Toutes les fonctions stockées de machine learning Aurora retournent des types numériques ou `VARCHAR`. Vous pouvez utiliser tous les types numériques à l'exception de `BIT`. Les autres types, tels que `JSON`, `BLOB`, `TEXT` et `DATE` ne sont pas autorisés.

L'exemple suivant illustre la syntaxe `CREATE FUNCTION` utilisée pour travailler avec SageMaker AI.

```
CREATE FUNCTION function_name (
     arg1 type1,
     arg2 type2, ...)
     [DEFINER = user]
  RETURNS mysql_type
  [SQL SECURITY { DEFINER | INVOKER } ]
  ALIAS AWS_SAGEMAKER_INVOKE_ENDPOINT
  ENDPOINT NAME 'endpoint_name'
```

```
  [MAX_BATCH_SIZE max_batch_size];
```

Il s'agit d'une extension de l'instruction DDL `CREATE FUNCTION` normale. Dans l'instruction `CREATE FUNCTION` qui définit la fonction SageMaker AI, vous ne spécifiez pas un corps de la fonction. Au lieu de cela, vous spécifiez le mot-clé `ALIAS` à la place du corps de la fonction. Actuellement, le machine learning Aurora prend uniquement en charge `aws_sagemaker_invoke_endpoint` pour cette syntaxe étendue. Vous devez spécifier le paramètre `endpoint_name`. Un point de terminaison SageMaker AI peut disposer de caractéristiques différentes pour chaque modèle.

> **Note**
>
> Pour plus d'informations sur `CREATE FUNCTION`, consultez CREATE PROCEDURE et CREATE FUNCTION, dans le Manuel de référence MySQL 8.0.

Le paramètre `max_batch_size` est facultatif. Par défaut, la taille maximale du lot est de 10 000. Vous pouvez utiliser ce paramètre dans votre fonction pour limiter le nombre maximum d'entrées traitées dans une demande par lot adressée à SageMaker AI. Le paramètre `max_batch_size` permet d'éviter une erreur causée par des entrées trop larges ou d'éviter à SageMaker AI de retourner une réponse plus rapidement. Ce paramètre affecte la taille d'un tampon interne utilisé pour le traitement des demandes SageMaker AI. Une valeur trop importante pour `max_batch_size` peut entraîner une surcharge de mémoire substantielle sur votre instance de base de données.

Nous recommandons de laisser le paramètre `MANIFEST` sur sa valeur par défaut de `OFF`. Bien que vous puissiez utiliser l'option `MANIFEST ON`, certaines fonctionnalités SageMaker AI ne peuvent pas directement utiliser le CSV exporté avec cette option. Le format de manifeste n'est pas compatible avec le format de manifeste prévu par SageMaker AI.

Vous créez une fonction stockée séparée pour chacun de vos modèles SageMaker AI. Ce mappage de fonctions vers des modèles est obligatoire, car un point de terminaison est associé à un modèle spécifique, et chaque modèle accepte différents paramètres. L'utilisation de types SQL pour les entrées de modèle et le type de sortie de modèle permet d'éviter les erreurs de conversion de types lors du transfert des données entre les services AWS. Vous pouvez contrôler qui peut appliquer le modèle. Vous pouvez également contrôler les caractéristiques d'exécution en spécifiant un paramètre représentant la taille de lot maximum.

Actuellement, toutes les fonctions de machine learning Aurora disposent de la propriété `NOT DETERMINISTIC`. Si vous ne spécifiez pas cette propriété de manière explicite, Aurora définit

automatiquement `NOT DETERMINISTIC`. Cela s'explique, car le modèle SageMaker AI peut être modifié sans aucune notification à la base de données. Dans ce cas, les appels à une fonction de machine learning Aurora peuvent retourner différents résultats pour la même entrée au sein d'une seule transaction.

Vous ne pouvez pas utiliser les caractéristiques `CONTAINS SQL`, `NO SQL`, `READS SQL DATA` ou `MODIFIES SQL DATA` dans votre instruction `CREATE FUNCTION`.

Voici un exemple d'appel d'un point de terminaison SageMaker AI pour détecter des anomalies. Il existe un point de terminaison SageMaker AI `random-cut-forest-model`. Le modèle correspondant est déjà entraîné par l'algorithme `random-cut-forest`. Pour chaque entrée, le modèle retourne un score d'anomalie. Cet exemple illustre les points de données dont le score dépasse de 3 déviations standard (environ le 99,9e centile) le score moyen.

```
CREATE FUNCTION anomaly_score(value real) returns real
  alias aws_sagemaker_invoke_endpoint endpoint name 'random-cut-forest-model-demo';

set @score_cutoff = (select avg(anomaly_score(value)) + 3 * std(anomaly_score(value))
 from nyc_taxi);

select *, anomaly_detection(value) score from nyc_taxi
  where anomaly_detection(value) > @score_cutoff;
```

Exigences du jeu de caractères pour les fonctions SageMaker AI qui retournent des chaînes

Nous recommandons de spécifier un jeu de caractères de `utf8mb4` comme type de retour pour vos fonctions SageMaker AI qui retournent des valeurs de chaîne. Si cela n'est pas pratique, utilisez une chaîne suffisamment longue pour que le type de retour conserve une valeur représentée dans le jeu de caractères `utf8mb4`. L'exemple suivant illustre comment déclarer le jeu de caractères `utf8mb4` pour votre fonction.

```
CREATE FUNCTION my_ml_func(...) RETURNS VARCHAR(5) CHARSET utf8mb4 ALIAS ...
```

Actuellement, chaque fonction SageMaker AI qui retourne une chaîne utilise le jeu de caractères `utf8mb4` pour la valeur de retour. La valeur de retour utilise ce jeu de caractères même si votre fonction SageMaker AI déclare de manière implicite ou explicite un jeu de caractères différent pour son type de retour. Si votre fonction SageMaker AI déclare un jeu de caractères différent pour la valeur de retour, les données retournées peuvent être tronquées en mode silencieux si vous les

stockez dans une colonne de tableau pas assez longue. Par exemple, une requête avec une clause `DISTINCT` crée un tableau temporaire. Ainsi, le résultat de la fonction SageMaker AI peut être tronqué en raison de la gestion interne des chaînes lors d'une requête.

Exportation des données vers Amazon S3 pour l'entraînement des modèles SageMaker AI (avancé)

Nous vous recommandons de commencer à utiliser le machine learning Aurora et SageMaker AI avec certains des algorithmes fournis, et de demander aux scientifiques des données de votre équipe de vous fournir les points de terminaison SageMaker AI que vous pouvez utiliser avec votre code SQL. Vous trouverez ci-dessous des informations minimales sur l'utilisation de votre propre compartiment Amazon S3 avec vos propres modèles SageMaker AI et votre cluster de bases de données Aurora MySQL.

Le machine learning comprend deux étapes principales : la formation et l'inférence. Pour entraîner des modèles SageMaker AI, vous exportez des données vers un compartiment Amazon S3. Le compartiment Amazon S3 est utilisé par une instance de bloc-note Jupyter SageMaker AI, afin de former votre modèle avant de le déployer. Vous pouvez utiliser l'instruction `SELECT INTO OUTFILE S3` pour interroger les données d'un cluster de bases de données Aurora MySQL et les enregistrer directement dans des fichiers texte stockés dans un compartiment Amazon S3. Ensuite, l'instance de bloc-bote consomme les données du compartiment Amazon S3 dans le cadre de la formation.

Le machine learning Aurora étend la syntaxe `SELECT INTO OUTFILE` existante dans Aurora MySQL pour exporter les données au format CSV. Le fichier CSV généré peut être directement consommé par des modèles ayant besoin de ce format dans le cadre de la formation.

```
SELECT * INTO OUTFILE S3 's3_uri' [FORMAT {CSV|TEXT} [HEADER]] FROM table_name;
```

L'extension prend en charge le format CSV standard.

- Le format `TEXT` est identique au format d'exportation MySQL existant. Il s'agit du format par défaut.
- Le format `CSV` est inédit et suit la spécification dans RFC-4180.
- Si vous spécifiez le mot-clé facultatif `HEADER`, le fichier de sortie contient une ligne d'en-tête. Les étiquettes de la ligne d'en-tête correspondent aux noms de colonnes de l'instruction `SELECT`.
- Vous pouvez toujours utiliser les mots-clés `CSV` et `HEADER` comme identifiants.

La syntaxe étendue et la grammaire de `SELECT INTO` sont désormais comme suit :

```
INTO OUTFILE S3 's3_uri'
[CHARACTER SET charset_name]
[FORMAT {CSV|TEXT} [HEADER]]
[{FIELDS | COLUMNS}
  [TERMINATED BY 'string']
  [[OPTIONALLY] ENCLOSED BY 'char']
  [ESCAPED BY 'char']
]
[LINES
  [STARTING BY 'string']
  [TERMINATED BY 'string']
]
```

Considérations sur les performances du machine learning Aurora avec Aurora MySQL

Les services Amazon Bedrock, Amazon Comprehend et SageMaker AI effectuent la majeure partie du travail lorsqu'ils sont invoqués par une fonction de machine learning Aurora. Cela signifie que vous pouvez adapter ces ressources selon vos besoins, de manière indépendante. Pour votre cluster de bases de données Aurora MySQL, vous pouvez rendre vos appels de fonctions aussi efficaces que possible. Vous trouverez ci-dessous quelques considérations relatives aux performances à prendre en compte lors de l'utilisation du machine learning Aurora.

Modèle et invite

Les performances obtenues avec Amazon Bedrock varient considérablement selon le modèle et l'invite utilisés. Choisissez le modèle et l'invite qui conviennent le mieux à votre cas d'utilisation.

Cache de requête

Le cache de requête MySQL Aurora ne fonctionne pas pour les fonctions de machine learning Aurora. Aurora MySQL ne stocke pas de résultats de requête dans le cache de requête pour aucune instruction SQL appelant des fonctions de machine learning Aurora.

Optimisation par lots pour les appels de fonction de machine learning Aurora

L'élément principal des performances de machine learning Aurora que vous pouvez influencer depuis votre cluster Aurora est le paramètre du mode par lots pour les appels vers les fonctions stockées de machine learning Aurora. Les fonctions de machine learning occasionnant généralement une

surcharge conséquente, l'appel d'un service externe séparément pour chaque ligne est impraticable. Le machine learning Aurora peut réduire cette surcharge en combinant dans un seul lot les appels au service de machine learning Aurora externe pour de nombreuses lignes. Le machine learning Aurora reçoit les réponses pour toutes les lignes d'entrée, puis transmet les réponses, ligne par ligne, à la requête en cours d'exécution. Cette optimisation améliore le débit et la latence de vos requêtes Aurora sans en modifier les résultats.

Lorsque vous créez une fonction stockée Aurora connectée à un point de terminaison SageMaker AI, vous définissez le paramètre de taille de lot. Ce paramètre influence le nombre de ligne transféré pour chaque appel sous-jacent vers SageMaker AI. Pour les requêtes qui traitent un grand nombre de lignes, le nombre de ressources permettant d'exécuter un appel SageMaker AI séparé pour chaque ligne peut être substantiel. Plus l'ensemble de données traité par la procédure stockée est important, plus grande sera la taille de lot.

Pour savoir si l'optimisation du mode par lots peut être appliquée à une fonction SageMaker AI, vérifiez le plan de requête produit par l'instruction `EXPLAIN PLAN`. Dans ce cas, la colonne `extra` du plan d'exécution inclut `Batched machine learning`. L'exemple suivant illustre un appel vers une fonction SageMaker AI qui utilise un mode par lots.

```
mysql> CREATE FUNCTION anomaly_score(val real) returns real alias
 aws_sagemaker_invoke_endpoint endpoint name 'my-rcf-model-20191126';
Query OK, 0 rows affected (0.01 sec)

mysql> explain select timestamp, value, anomaly_score(value) from nyc_taxi;
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len |
 ref  | rows | filtered | Extra                   |
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
|  1 | SIMPLE      | nyc_taxi | NULL       | ALL  | NULL          | NULL | NULL    |
 NULL |   48 |   100.00 | Batched machine learning |
+----+-------------+----------+------------+------+---------------+------+---------
+------+------+----------+-------------------------+
1 row in set, 1 warning (0.01 sec)
```

Lorsque vous appelez l'une des fonctions Amazon Comprehend intégrées, vous pouvez contrôler le taille de lot en spécifiant le paramètre `max_batch_size` facultatif. Ce paramètre limite le nombre maximum de valeurs `input_text` traitées dans chaque lot. En envoyant plusieurs objets à la fois, cela réduit le nombre d'allers-retours entre Aurora et Amazon Comprehend. Limiter la taille de lot

s'avère utile dans les situations impliquant une requête avec une clause `LIMIT`. En utilisant une petite valeur pour `max_batch_size`, vous pouvez éviter d'appeler Amazon Comprehend plus de fois que vous n'avez de textes d'entrée.

L'optimisation par lots pour l'évaluation des fonctions de machine learning Aurora s'applique dans les cas suivants :

- Appels de fonction dans la liste sélectionnée ou la clause `WHERE` des instructions `SELECT`.
- Appels de fonction dans la liste `VALUES` des instructions `INSERT` et `REPLACE`.
- Fonctions SageMaker AI dans les valeurs `SET` des instructions `UPDATE` :

```
INSERT INTO MY_TABLE (col1, col2, col3) VALUES
  (ML_FUNC(1), ML_FUNC(2), ML_FUNC(3)),
  (ML_FUNC(4), ML_FUNC(5), ML_FUNC(6));
UPDATE MY_TABLE SET col1 = ML_FUNC(col2), SET col3 = ML_FUNC(col4) WHERE ...;
```

Surveillance du machine learning Aurora

Vous pouvez surveiller les opérations par lots de machine learning Aurora en interrogeant plusieurs variables globales, comme indiqué dans l'exemple ci-dessous.

```
show status like 'Aurora_ml%';
```

Vous pouvez réinitialiser ces variables d'état en utilisant une instruction `FLUSH STATUS`. Ainsi, tous les chiffres représentent des totaux, des moyennes, etc. depuis la dernière réinitialisation de la variable.

`Aurora_ml_logical_request_cnt`

Nombre de demandes logiques que l'instance de base de données a évaluées pour qu'elles soient envoyées aux services de machine learning Aurora depuis la dernière réinitialisation de l'état. Si un traitement par lots a été utilisé, cette valeur peut être supérieure à `Aurora_ml_actual_request_cnt`.

`Aurora_ml_logical_response_cnt`

Le nombre total de réponses reçues par Aurora MySQL de la part des services de machine learning Aurora dans toutes les requêtes exécutées par des utilisateurs de l'instance de base de données.

`Aurora_ml_actual_request_cnt`

Nombre total de demandes effectuées par Aurora MySQL auprès des services de machine learning Aurora sur l'ensemble des requêtes exécutées par les utilisateurs de l'instance de base de données.

`Aurora_ml_actual_response_cnt`

Le nombre total de réponses reçues par Aurora MySQL de la part des services de machine learning Aurora dans toutes les requêtes exécutées par des utilisateurs de l'instance de base de données.

`Aurora_ml_cache_hit_cnt`

Le nombre total d'accès au cache interne reçus par Aurora MySQL de la part des services de machine learning Aurora dans toutes les requêtes exécutées par des utilisateurs de l'instance de base de données.

`Aurora_ml_retry_request_cnt`

Nombre de nouvelles tentatives de demande que l'instance de base de données a envoyées aux services de machine learning Aurora depuis la dernière réinitialisation de l'état.

`Aurora_ml_single_request_cnt`

Le nombre total de fonctions de machine learning Aurora évaluées par un mode autre que par lots dans toutes les requêtes exécutées par des utilisateurs de l'instance de base de données.

Pour obtenir des informations sur la surveillance des performances des opérations SageMaker AI appelées depuis des fonctions de machine learning Aurora, consultez Surveillance Amazon SageMaker AI.

Utilisation du machine learning Amazon Aurora avec Aurora PostgreSQL

En utilisant l'apprentissage automatique Amazon Aurora avec votre cluster de bases de données Aurora PostgreSQL, vous pouvez utiliser Amazon Comprehend, Amazon SageMaker AI ou Amazon Bedrock, selon vos besoins. Ces services prennent chacun en charge des cas d'utilisation propres au machine learning.

L'apprentissage automatique Aurora est pris en charge dans certaines versions Régions AWS et pour des versions spécifiques d'Aurora PostgreSQL uniquement. Avant d'essayer de configurer le

machine learning Aurora, vérifiez la disponibilité pour votre version d'Aurora PostgreSQL et votre région. Pour en savoir plus, consultez Machine Learning Aurora avec Aurora PostgreSQL.

Rubriques

Exigences pour l'utilisation du machine learning Aurora avec Aurora PostgreSQL

AWS les services d'apprentissage automatique sont des services gérés qui sont configurés et exécutés dans leurs propres environnements de production. L'apprentissage automatique Aurora prend en charge l'intégration avec Amazon Comprehend, SageMaker AI et Amazon Bedrock. Avant d'essayer de configurer votre cluster de bases de données Aurora PostgreSQL pour utiliser le machine learning Aurora, assurez-vous de comprendre les exigences et prérequis suivants.

- Les services Amazon Comprehend, SageMaker AI et Amazon Bedrock doivent être exécutés de la même manière que Région AWS votre cluster de base de données Aurora PostgreSQL. Vous ne pouvez pas utiliser les services Amazon Comprehend, SageMaker AI ou Amazon Bedrock à partir d'un cluster de base de données Aurora PostgreSQL situé dans une autre région.
- Si votre cluster de base de données Aurora PostgreSQL se trouve dans un cloud public virtuel (VPC) basé sur le service Amazon VPC différent de celui de vos services Amazon Comprehend SageMaker et AI, le groupe de sécurité du VPC doit autoriser les connexions sortantes vers le service d'apprentissage automatique Aurora cible. Pour de plus amples informations, veuillez consulter Activation de la communication réseau entre Amazon Aurora et d'autres services AWS.

- Pour l' SageMaker IA, les composants d'apprentissage automatique que vous souhaitez utiliser pour les inférences doivent être configurés et prêts à être utilisés. Pendant le processus de configuration de votre cluster de base de données Aurora PostgreSQL, vous devez disposer de l'Amazon Resource Name (ARN) SageMaker du point de terminaison AI. Les data scientists de votre équipe sont probablement les mieux placés pour travailler avec l' SageMaker IA pour préparer les modèles et effectuer les autres tâches de ce type. Pour commencer à utiliser Amazon SageMaker AI, consultez Get Started with Amazon SageMaker AI. Pour plus d'informations sur les inférences et les points de terminaison, consultez Inférence en temps réel.
- Pour Amazon Bedrock, vous devez disposer de l'ID des modèles Bedrock que vous souhaitez utiliser pour les inférences, disponible au cours du processus de configuration de votre cluster de bases de données Aurora PostgreSQL. Les scientifiques des données de votre équipe sont probablement les mieux placés pour travailler avec Bedrock afin de décider des modèles à utiliser, les optimiser si nécessaire et effectuer d'autres tâches de ce type. Pour commencer à utiliser Amazon Bedrock, consultez Comment configurer Bedrock.
- Les utilisateurs d'Amazon Bedrock doivent demander l'accès aux modèles avant de pouvoir s'en servir. Si vous souhaitez ajouter des modèles supplémentaires pour la génération de texte, de discussion instantanée et d'images, vous devez demander l'accès aux modèles dans Amazon Bedrock. Pour plus d'informations, consultez Accès aux modèles.

Fonctions prises en charge et limitations du machine learning Aurora avec Aurora PostgreSQL

L'apprentissage automatique Aurora prend en charge tout point de terminaison d' SageMaker IA capable de lire et d'écrire le format CSV (valeurs séparées par des virgules) via une `ContentType` valeur de. `text/csv` Les algorithmes d' SageMaker IA intégrés qui acceptent actuellement ce format sont les suivants.

- Linear Learner
- Random Cut Forest
- XGBoost

  Pour en savoir plus sur ces algorithmes, consultez la section Choisir un algorithme dans le manuel Amazon SageMaker AI Developer Guide.

Lorsque vous utilisez Amazon Bedrock avec le machine learning Aurora, les limitations suivantes s’appliquent :

- Les fonctions définies par l'utilisateur (UDFs) fournissent un moyen natif d'interagir avec Amazon Bedrock. Ils UDFs n'ont pas d'exigences spécifiques en matière de demande ou de réponse, ils peuvent donc utiliser n'importe quel modèle.
- Vous pouvez l'utiliser UDFs pour créer le flux de travail souhaité. Par exemple, vous pouvez combiner des primitives de base, telles que `pg_cron` pour exécuter une requête, récupérer des données, générer des inférences et écrire dans des tables pour répondre directement aux requêtes.
- UDFs ne prennent pas en charge les appels par lots ou en parallèle.
- L’extension de machine learning Aurora ne prend pas en charge les interfaces vectorielles. Dans le cadre de l’extension, une fonction est disponible pour générer les vectorisations de la réponse du modèle au format `float8[]` permettant de stocker ces vectorisations dans Aurora. Pour plus d’informations sur l’utilisation de `float8[]`, consultez Utilisations d’Amazon Bedrock avec votre cluster de bases de données Aurora PostgreSQL.

Configuration de votre cluster de bases de données Aurora PostgreSQL de façon à utiliser le machine learning Aurora

Pour que l'apprentissage automatique Aurora fonctionne avec votre cluster de bases de données Aurora PostgreSQL, vous devez créer Gestion des identités et des accès AWS un rôle (IAM) pour chacun des services que vous souhaitez utiliser. Le rôle IAM permet à votre cluster de bases de données Aurora PostgreSQL d’utiliser le service de machine learning Aurora pour le compte du cluster. Vous devez également installer l’extension du machine learning Aurora. Dans les rubriques suivantes, vous pouvez trouver des procédures de configuration pour chacun de ces services de machine learning Aurora.

Rubriques

- Configuration d’Aurora PostgreSQL pour utiliser Amazon Bedrock
- Configuration d’Aurora PostgreSQL pour utiliser Amazon Comprehend
- Configuration d'Aurora PostgreSQL pour utiliser Amazon AI SageMaker
  - Configuration d'Aurora PostgreSQL pour utiliser Amazon S3 SageMaker pour l'IA (version avancée)
- Installation de l’extension du machine learning Aurora

Configuration d’Aurora PostgreSQL pour utiliser Amazon Bedrock

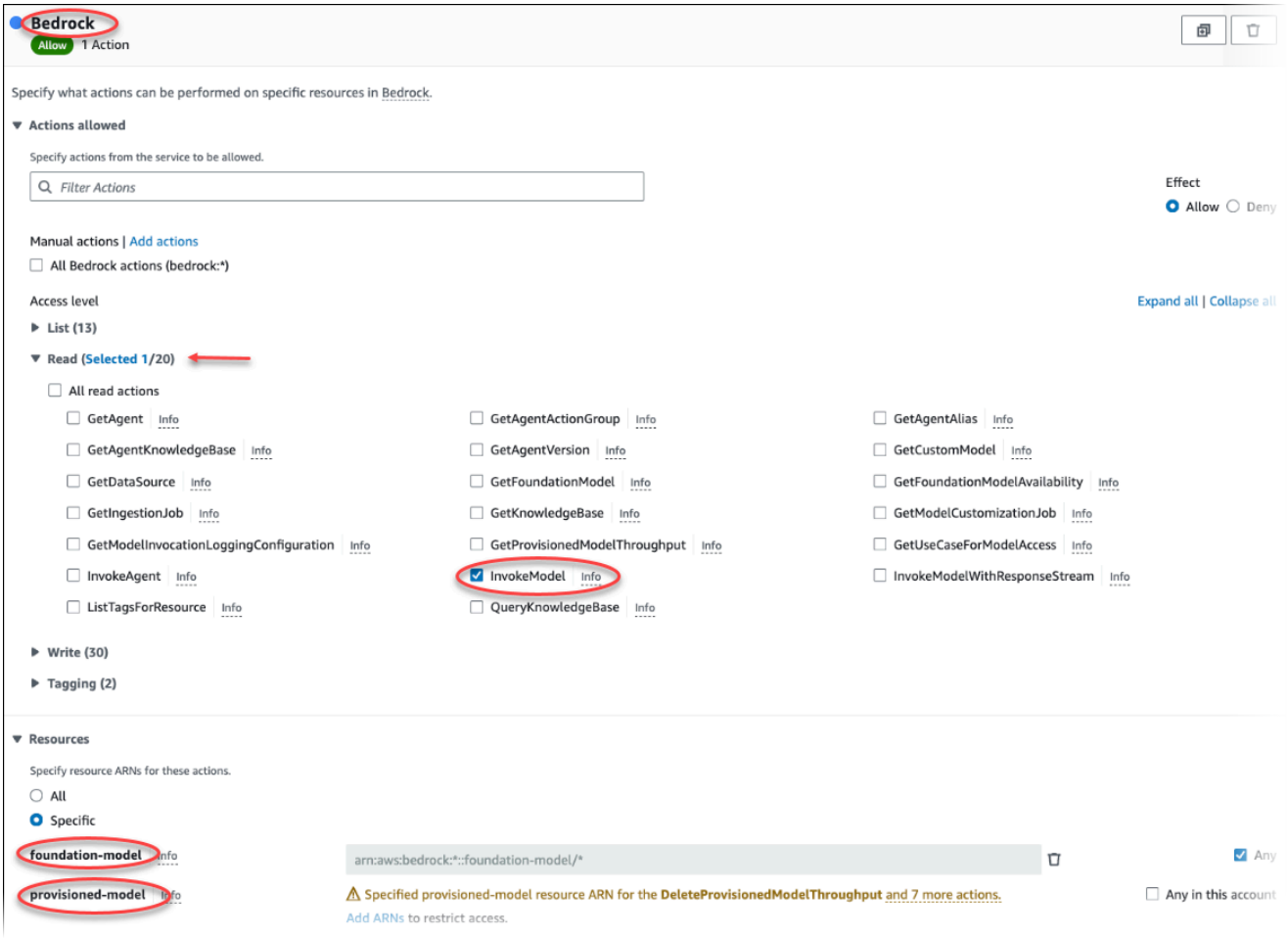
Dans la procédure suivante, vous devez d’abord créer le rôle et la stratégie IAM qui donnent à votre Aurora PostgreSQL l’autorisation d’utiliser Amazon Bedrock pour le compte du cluster. Vous associez ensuite la stratégie à un rôle IAM que votre cluster de bases de données Aurora PostgreSQL utilise pour fonctionner avec Amazon Bedrock. Pour des raisons de simplicité, cette procédure utilise la AWS Management Console pour effectuer toutes les tâches.

Pour configurer votre cluster de bases de données Aurora PostgreSQL pour utiliser Amazon Bedrock

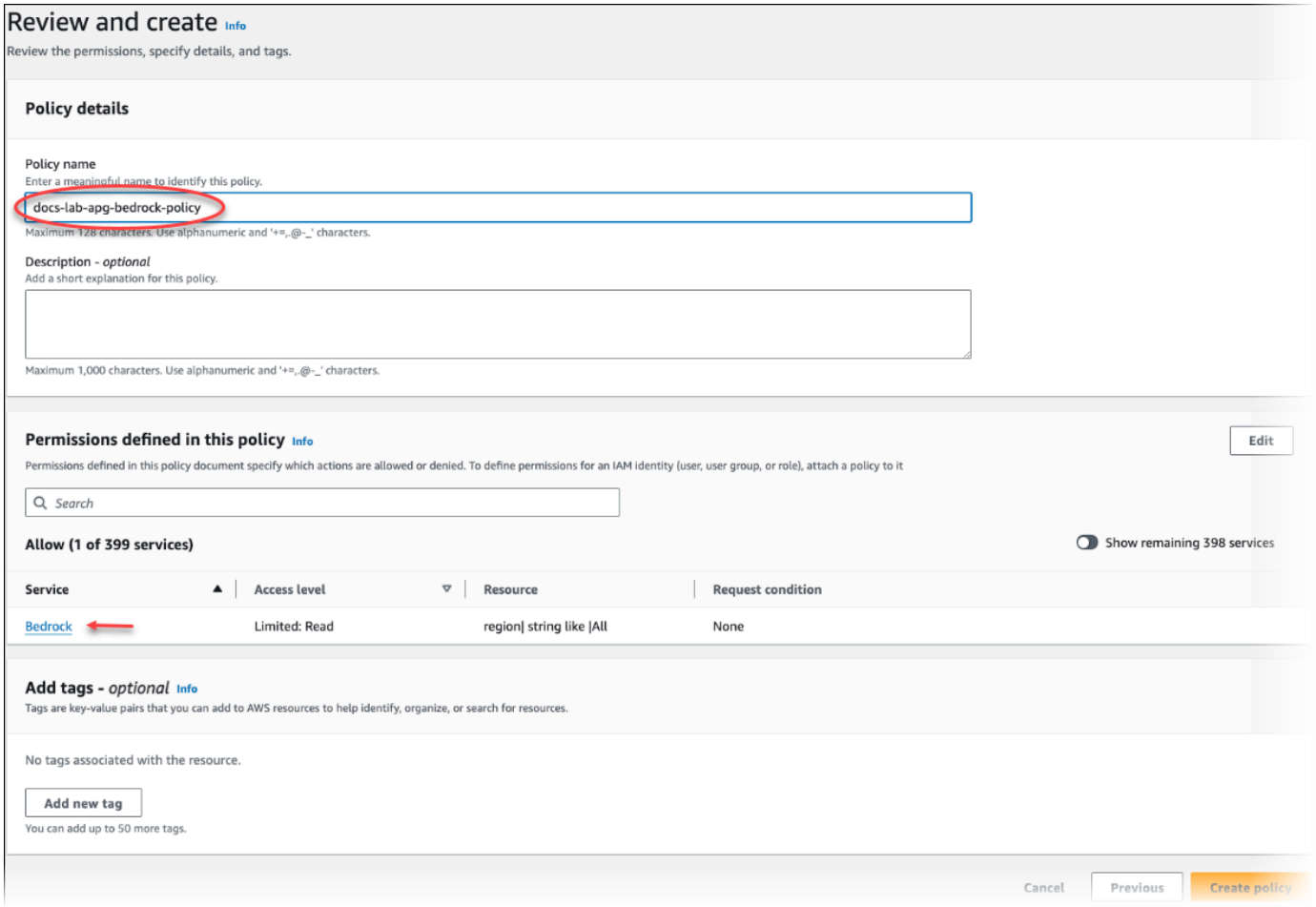
1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
3. Choisissez Politiques (sous Gestion des accès) dans le menu de la console Gestion des identités et des accès AWS (IAM).

    a. Choisissez Create Policy (Créer une politique). Sur la page de l’éditeur visuel, choisissez Service, puis saisissez Bedrock dans le champ Sélectionner un service. Développez le niveau d’accès en lecture. Choisissez InvokeModelparmi les paramètres de lecture d'Amazon Bedrock.

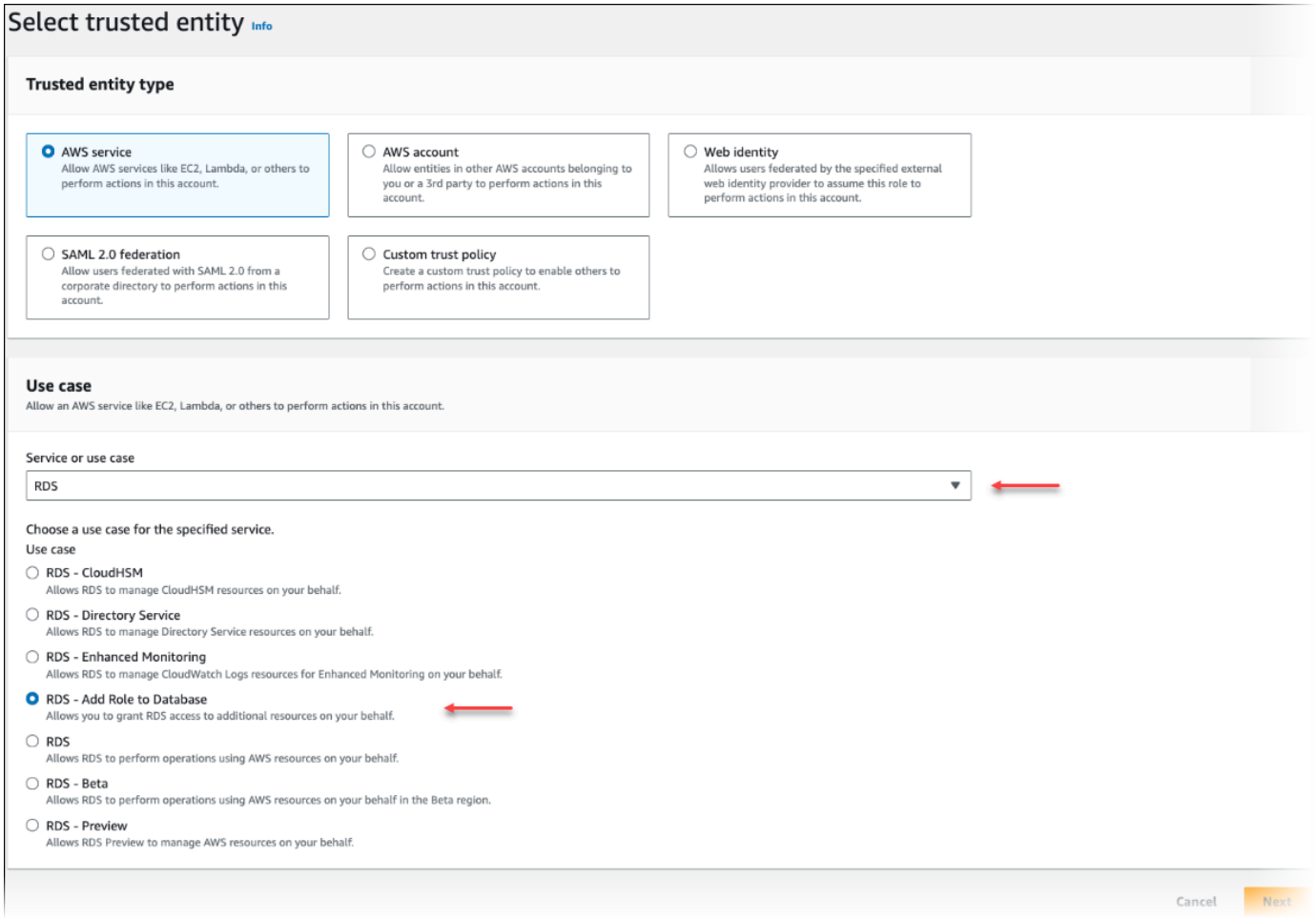
    b. Choisissez le Foundation/Provisioned modèle auquel vous souhaitez accorder l'accès en lecture via la politique.



4. Choisissez Next: Tags (Suivant : Balises) et définissez toutes les balises (facultatif). Choisissez Suivant : Vérification. Saisissez un nom et une description pour la stratégie, comme indiqué dans l'image.

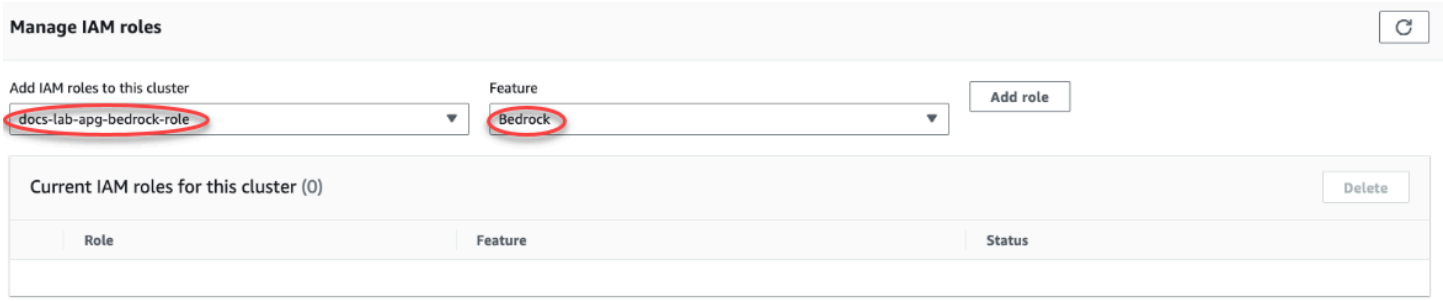


5. Choisissez Create Policy (Créer une politique). La console affiche une alerte lorsque la stratégie a été enregistrée. Vous pouvez la trouver dans la liste des stratégies.
6. Choisissez Roles (under Access management) (Rôles (sous Gestion des accès)) sur la console IAM.
7. Choisissez Créer un rôle.
8. Sur la page Sélectionner une entité de confiance, choisissez la vignette Service AWS , puis choisissez RDS pour ouvrir le sélecteur.
9. Choisissez RDS – Add Role to Database (RDS – Ajoutez un rôle à la base de données).



10. Choisissez Suivant. Sur la page Ajouter des autorisations, recherchez la stratégie que vous avez créée à l'étape précédente et choisissez-la parmi celles répertoriées. Choisissez Suivant.
11. Next: Review (Suivant : Vérification). Saisissez un nom pour le rôle IAM et une description.
12. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
13. Accédez à l'emplacement Région AWS où se trouve votre cluster de base de données Aurora PostgreSQL.
14. Dans le panneau de navigation, choisissez Bases de données, puis le cluster de bases de données Aurora PostgreSQL que vous souhaitez utiliser avec Bedrock.
15. Choisissez l'onglet Connectivity & security (Connectivité et sécurité) et faites défiler la page pour accéder à la section Manage IAM roles (Gérer les rôles IAM). Dans le sélecteur Add IAM roles to this cluster (Ajouter des rôles IAM à ce cluster), choisissez le rôle que vous avez créé dans les étapes précédentes. Dans le sélecteur Fonction, choisissez Bedrock, puis choisissez Ajouter un rôle.

Le rôle (avec sa stratégie) est associé au cluster de bases de données Aurora PostgreSQL. Une fois le processus terminé, vous pouvez trouver le rôle dans la liste Rôles IAM actuels pour ce cluster, comme indiqué dans l'image suivante.



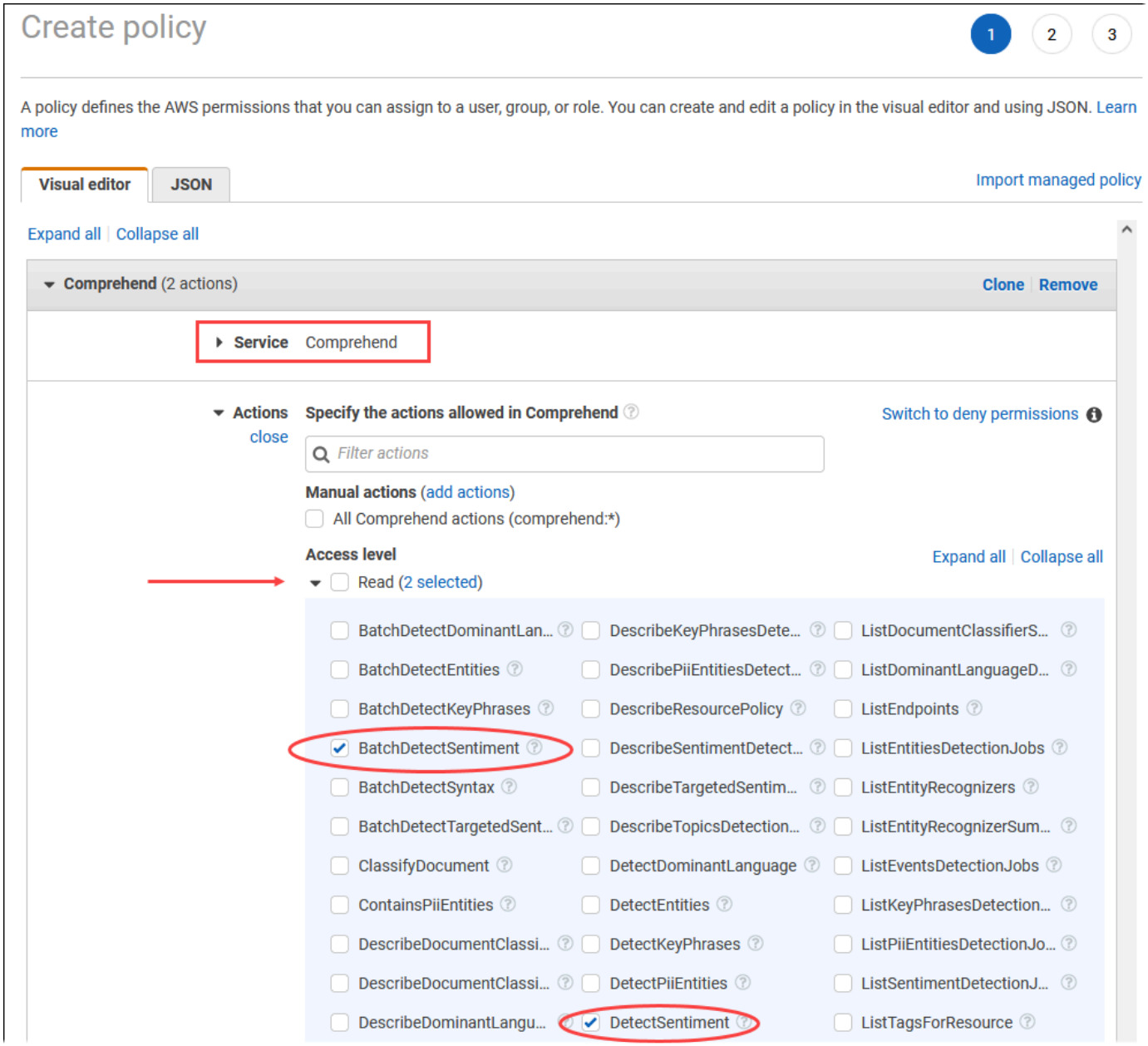
La configuration IAM pour Amazon Bedrock est terminée. Continuez à configurer votre Aurora PostgreSQL pour qu'il fonctionne avec le machine learning Aurora en installant l'extension comme indiqué dans Installation de l'extension du machine learning Aurora.

Configuration d'Aurora PostgreSQL pour utiliser Amazon Comprehend

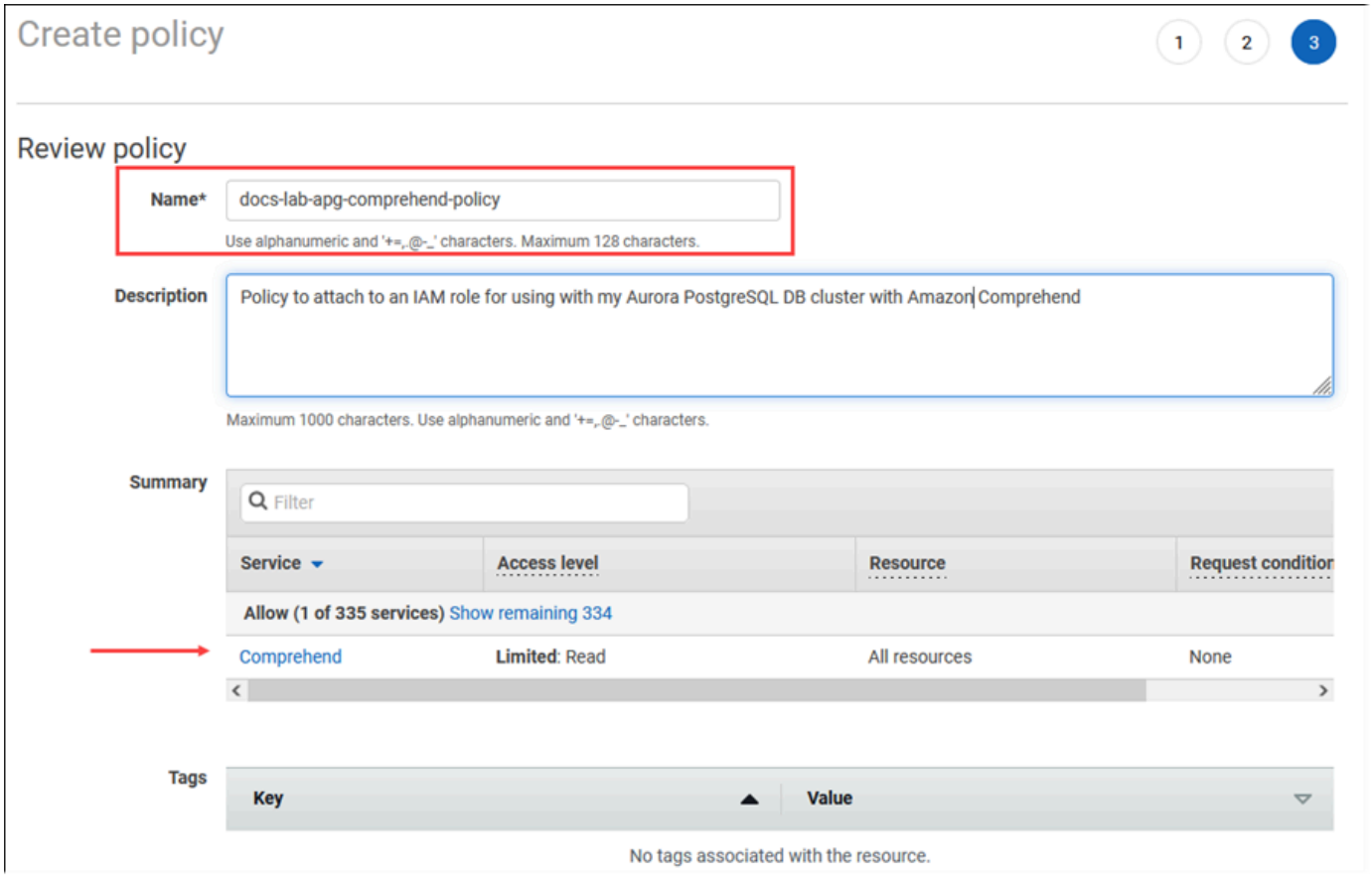
Dans la procédure suivante, vous devez d'abord créer le rôle et la stratégie IAM qui donnent à votre Aurora PostgreSQL l'autorisation d'utiliser Amazon Comprehend pour le compte du cluster. Vous associez ensuite la stratégie à un rôle IAM que votre cluster de bases de données Aurora PostgreSQL utilise pour fonctionner avec Amazon Comprehend. Par souci de simplicité, cette procédure utilise la AWS Management Console pour effectuer toutes les tâches.

Configurer votre cluster de bases de données Aurora PostgreSQL pour utiliser Amazon Comprehend

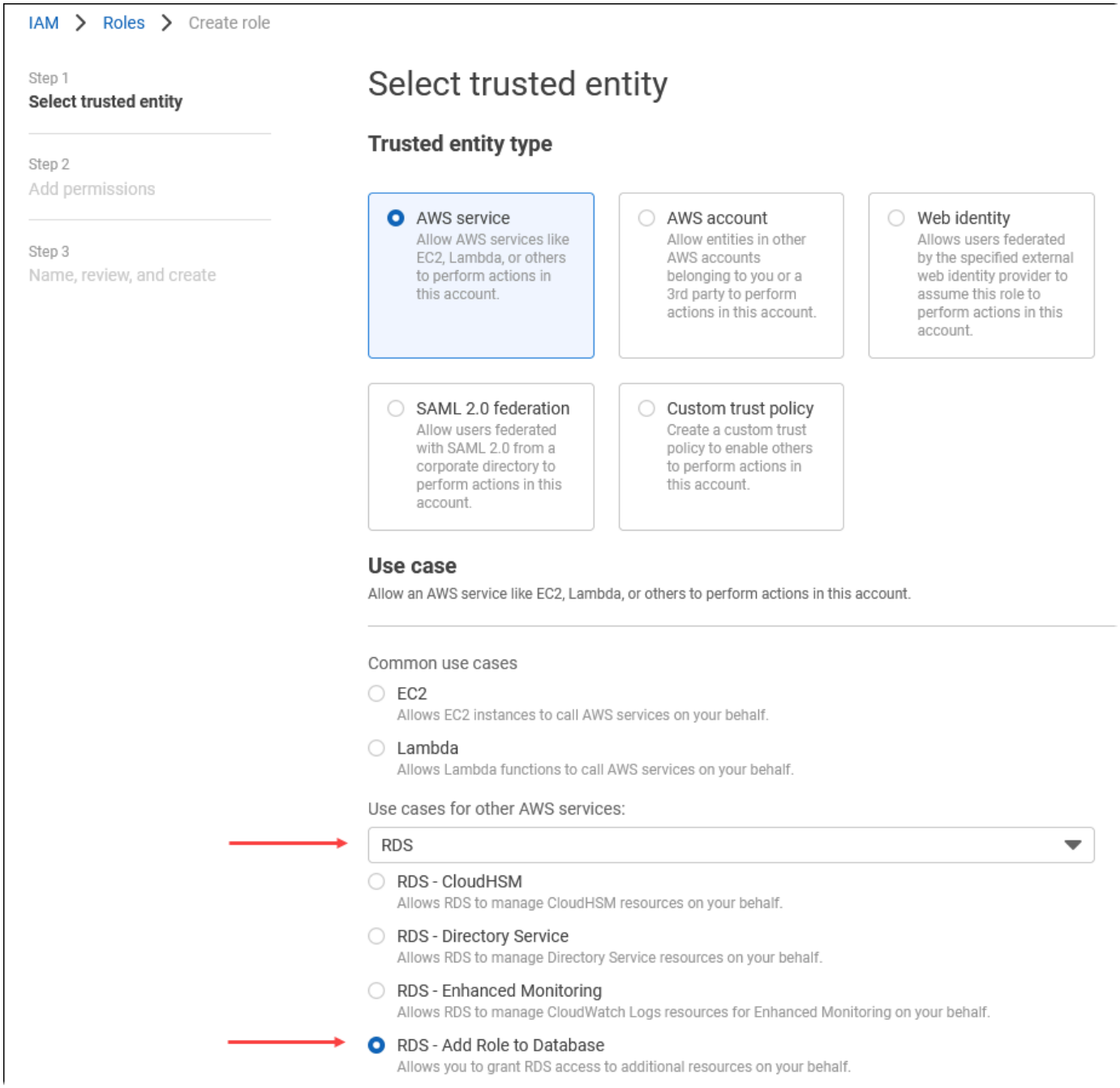
1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
3. Choisissez Politiques (sous Gestion des accès) dans le menu de la console Gestion des identités et des accès AWS (IAM).



4. Choisissez Create Policy (Créer une politique). Sur la page de l'éditeur visuel, choisissez Service, puis saisissez Comprehend dans le champ Sélectionner un service. Développez le niveau d'accès en lecture. Choisissez BatchDetectSentimentet DetectSentimentparmi les paramètres de lecture d'Amazon Comprehend.
5. Choisissez Next: Tags (Suivant : Balises) et définissez toutes les balises (facultatif). Choisissez Suivant : Vérification. Saisissez un nom et une description pour la stratégie, comme indiqué dans l'image.



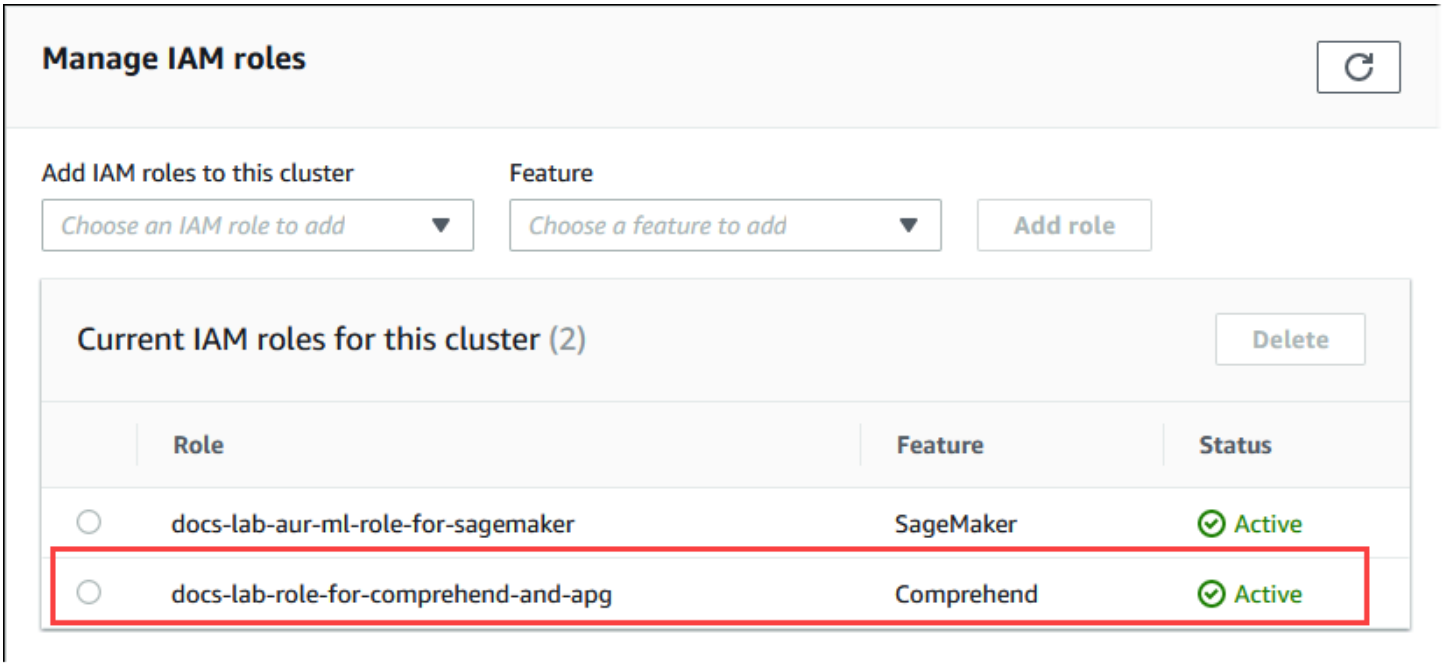
6. Choisissez Create Policy (Créer une politique). La console affiche une alerte lorsque la stratégie a été enregistrée. Vous pouvez la trouver dans la liste des stratégies.
7. Choisissez Roles (under Access management) (Rôles (sous Gestion des accès)) sur la console IAM.
8. Choisissez Créer un rôle.
9. Sur la page Sélectionner une entité de confiance, choisissez la vignette Service AWS , puis choisissez RDS pour ouvrir le sélecteur.
10. Choisissez RDS – Add Role to Database (RDS – Ajoutez un rôle à la base de données).



11. Choisissez Suivant. Sur la page Ajouter des autorisations, recherchez la stratégie que vous avez créée à l'étape précédente et choisissez-la parmi celles répertoriées. Choisissez Next (Suivant).
12. Next: Review (Suivant : Vérification). Saisissez un nom pour le rôle IAM et une description.
13. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
14. Accédez à l'emplacement Région AWS où se trouve votre cluster de base de données Aurora PostgreSQL.

15. Dans le panneau de navigation, choisissez Bases de données, puis le cluster de bases de données Aurora PostgreSQL que vous souhaitez utiliser avec Amazon Comprehend.
16. Choisissez l'onglet Connectivity & security (Connectivité et sécurité) et faites défiler la page pour accéder à la section Manage IAM roles (Gérer les rôles IAM). Dans le sélecteur Add IAM roles to this cluster (Ajouter des rôles IAM à ce cluster), choisissez le rôle que vous avez créé dans les étapes précédentes. Dans le sélecteur Fonction, choisissez Comprehend, puis Ajouter un rôle.

Le rôle (avec sa stratégie) est associé au cluster de bases de données Aurora PostgreSQL. Une fois le processus terminé, vous pouvez trouver le rôle dans la liste Rôles IAM actuels pour ce cluster, comme indiqué dans l'image suivante.



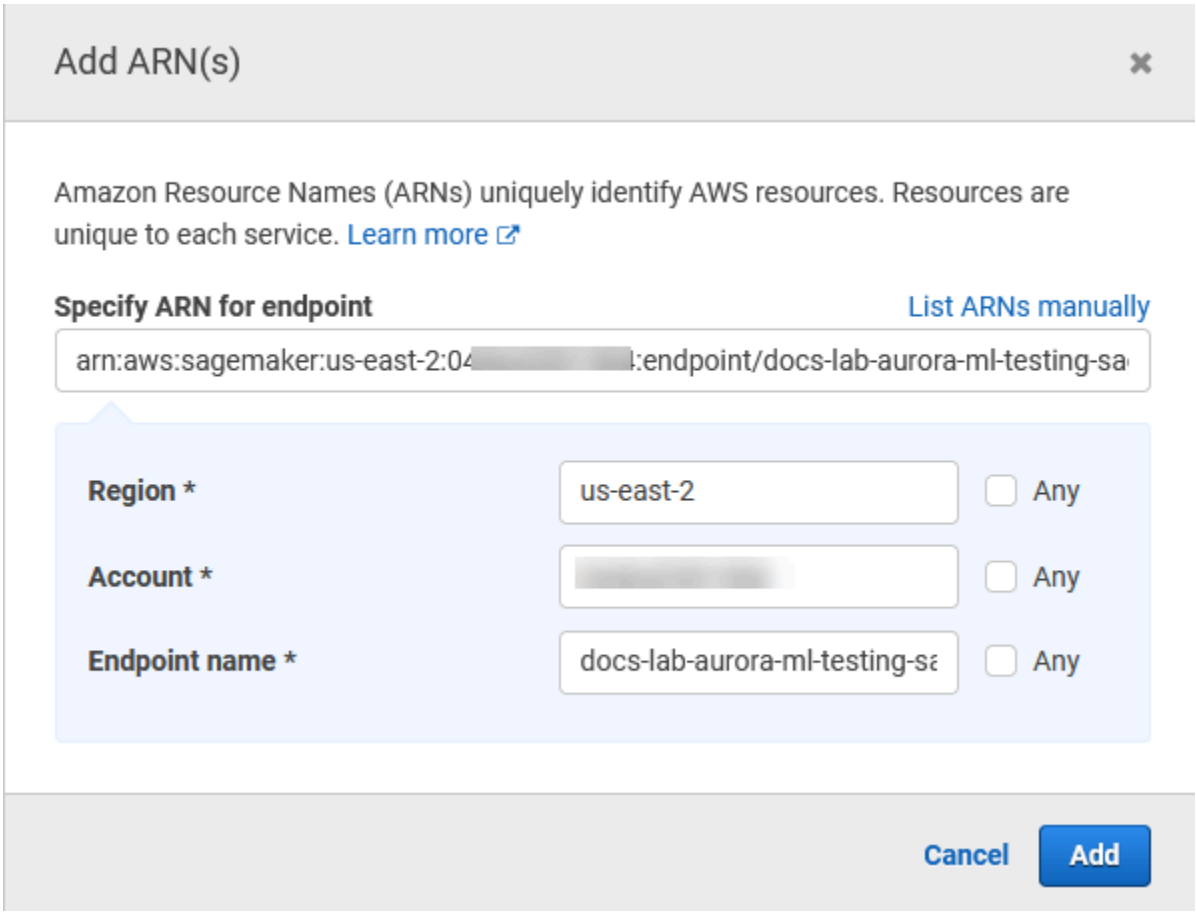
La configuration IAM pour Amazon Comprehend est terminée. Continuez à configurer votre Aurora PostgreSQL pour qu'il fonctionne avec le machine learning Aurora en installant l'extension comme indiqué dans Installation de l'extension du machine learning Aurora.

Configuration d'Aurora PostgreSQL pour utiliser Amazon AI SageMaker

Avant de pouvoir créer la politique et le rôle IAM pour votre cluster de base de données Aurora PostgreSQL, vous devez disposer de la configuration de votre modèle d'IA et de SageMaker votre point de terminaison.

Pour configurer votre cluster de base de données Aurora PostgreSQL afin d'utiliser l'IA SageMaker

1. Connectez-vous à la console IAM AWS Management Console et ouvrez-la à https://console.aws.amazon.com/iam/l'adresse.
2. Choisissez Politiques (sous Gestion des accès) dans le menu de la console Gestion des identités et des accès AWS (IAM), puis choisissez Créer une politique. Dans l'éditeur visuel, choisissez SageMakerle service. Pour Actions, ouvrez le sélecteur de lecture (sous Niveau d'accès) et choisissez InvokeEndpoint. Dans ce cas, une icône d'avertissement s'affiche.
3. Ouvrez le sélecteur de ressources et choisissez le lien Ajouter un ARN pour restreindre l'accès sous Spécifier l'ARN de la ressource du point de terminaison pour l' InvokeEndpoint action.
4. Entrez vos ressources Région AWS d' SageMaker IA et le nom de votre point de terminaison. Votre AWS compte est prérempli.



5. Choisissez Add (Ajouter) pour enregistrer. Choisissez Next: Tags (Suivant : Balises) et Next: Review (Suivant : Vérification) pour accéder à la dernière page du processus de création de la stratégie.

6. Saisissez un nom et une description pour la stratégie, puis choisissez Create policy (Créer une stratégie). La stratégie est créée et ajoutée à la liste des stratégies. Une alerte s'affiche dans la console lorsque cela se produit.
7. Dans la console IAM, choisissez Roles (Rôles).
8. Choisissez Créer un rôle.
9. Sur la page Sélectionner une entité de confiance, choisissez la vignette Service AWS , puis choisissez RDS pour ouvrir le sélecteur.
10. Choisissez RDS – Add Role to Database (RDS – Ajoutez un rôle à la base de données).
11. Choisissez Suivant. Sur la page Ajouter des autorisations, recherchez la stratégie que vous avez créée à l'étape précédente et choisissez-la parmi celles répertoriées. Choisissez Next (Suivant).
12. Next: Review (Suivant : Vérification). Saisissez un nom pour le rôle IAM et une description.
13. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
14. Accédez à l'emplacement Région AWS où se trouve votre cluster de base de données Aurora PostgreSQL.
15. Dans le volet de navigation, choisissez Databases, puis choisissez le cluster de bases de données Aurora PostgreSQL que vous souhaitez utiliser avec l'IA. SageMaker
16. Choisissez l'onglet Connectivity & security (Connectivité et sécurité) et faites défiler la page pour accéder à la section Manage IAM roles (Gérer les rôles IAM). Dans le sélecteur Add IAM roles to this cluster (Ajouter des rôles IAM à ce cluster), choisissez le rôle que vous avez créé dans les étapes précédentes. Dans le sélecteur de fonctionnalités, choisissez SageMaker AI, puis choisissez Ajouter un rôle.

Le rôle (avec sa stratégie) est associé au cluster de bases de données Aurora PostgreSQL. Une fois le processus terminé, vous pouvez trouver le rôle dans la liste Rôles IAM actuels pour ce cluster.

La configuration IAM pour l' SageMaker IA est terminée. Continuez à configurer votre Aurora PostgreSQL pour qu'il fonctionne avec le machine learning Aurora en installant l'extension comme indiqué dans Installation de l'extension du machine learning Aurora.

Configuration d'Aurora PostgreSQL pour utiliser Amazon S3 SageMaker pour l'IA (version avancée)

Pour utiliser l' SageMaker IA avec vos propres modèles plutôt que d'utiliser les composants prédéfinis fournis par l' SageMaker IA, vous devez configurer un bucket Amazon Simple Storage Service (Amazon S3) que le cluster de bases de données Aurora PostgreSQL pourra utiliser. Il s'agit d'une rubrique avancée qui n'est pas entièrement documentée dans ce Guide de l'utilisateur Amazon

Aurora. Le processus général est le même que pour intégrer le support à l' SageMaker IA, comme suit.

1. Créez la politique et le rôle IAM pour Amazon S3.
2. Ajoutez le rôle IAM et l'importation ou l'exportation Amazon S3 en tant que fonction dans l'onglet Connectivité et sécurité de votre cluster de bases de données Aurora PostgreSQL.
3. Ajoutez l'ARN du rôle à votre groupe de paramètres de cluster de bases de données personnalisé pour chaque cluster de bases de données Aurora.

Pour plus d'informations de base, consultez Exportation de données vers Amazon S3 pour la formation de modèles d' SageMaker IA (niveau avancé).

Installation de l'extension du machine learning Aurora

Les extensions d'apprentissage automatique Aurora `aws_ml 1.0` fournissent deux fonctions que vous pouvez utiliser pour appeler Amazon Comprehend, des services d' SageMaker intelligence artificielle, ainsi `aws_ml 2.0` que deux fonctions supplémentaires que vous pouvez utiliser pour appeler les services Amazon Bedrock. L'installation de ces extensions sur votre cluster de bases de données Aurora PostgreSQL crée également un rôle administratif pour la fonction.

> **Note**
>
> L'utilisation de ces fonctions dépend de l'achèvement de la configuration IAM pour le service d'apprentissage automatique Aurora (Amazon Comprehend SageMaker , AI, Amazon Bedrock), comme indiqué dans. Configuration de votre cluster de bases de données Aurora PostgreSQL de façon à utiliser le machine learning Aurora

- aws_comprehend.detect_sentiment – Vous utilisez cette fonction pour appliquer une analyse des sentiments au texte stocké dans la base de données de votre cluster de bases de données Aurora PostgreSQL.
- aws_sagemaker.invoke_endpoint — Vous utilisez cette fonction dans votre code SQL pour communiquer avec le point de terminaison AI depuis votre cluster. SageMaker
- aws_bedrock.invoke_model : vous utilisez cette fonction dans votre code SQL pour communiquer avec les modèles Bedrock de votre cluster. La réponse de cette fonction sera exprimée au format TEXT. Ainsi, si un modèle renvoie une réponse au format corps JSON, la sortie de cette fonction sera transmise à l'utilisateur final sous forme de chaîne de caractères.

- aws_bedrock.invoke_model_get_embeddings : vous utilisez cette fonction dans votre code SQL pour invoquer des modèles Bedrock qui renvoient des vectorisations de sortie dans une réponse JSON. Cette fonctionnalité peut être utilisée pour extraire les vectorisations liés directement à la clé JSON, ce qui permet d'optimiser la réponse et de l'adapter à vos processus internes.

Installer l'extension du machine learning Aurora dans votre cluster de bases de données Aurora PostgreSQL

- Utilisez `psql` pour vous connecter à l'instance d'enregistreur de votre cluster de bases de données Aurora PostgreSQL. Connectez-vous à la base de données spécifique dans laquelle vous souhaitez installer l'extension `aws_ml`.

```
psql --host=cluster-instance-1.111122223333.aws-region.rds.amazonaws.com --
port=5432 --username=postgres --password --dbname=labdb
```

```
labdb=> CREATE EXTENSION IF NOT EXISTS aws_ml CASCADE;
NOTICE:  installing required extension "aws_commons"
CREATE EXTENSION
labdb=>
```

L'installation des extensions `aws_ml` crée également le rôle administratif `aws_ml` et trois nouveaux schémas, comme suit.

- `aws_comprehend` – Schéma du service Amazon Comprehend et source de la fonction `detect_sentiment` (`aws_comprehend.detect_sentiment`).
- `aws_sagemaker`— Schéma du service d' SageMaker IA et source de la `invoke_endpoint` fonction (`aws_sagemaker.invoke_endpoint`).
- `aws_bedrock` : schéma du service Amazon Bedrock et source des fonctions `invoke_model(aws_bedrock.invoke_model)` et `invoke_model_get_embeddings(aws_bedrock.invoke_model_get_embeddings)`.

Le rôle `rds_superuser` se voit attribuer le rôle administratif `aws_ml` et devient `OWNER` de ces trois schémas de machine learning Aurora. Pour permettre aux autres utilisateurs de la base de données d'accéder aux fonctions de machine learning Aurora, `rds_superuser` doit accorder des privilèges `EXECUTE` sur les fonctions de machine learning Aurora. Par défaut, les privilèges `EXECUTE` sont révoqués de `PUBLIC` sur les fonctions des deux schémas de machine learning Aurora.

Dans une configuration de base de données à locataires multiples, vous pouvez empêcher les locataires d'accéder aux fonctions de machine learning d'Aurora en utilisant `REVOKE USAGE` sur le schéma de machine learning Aurora spécifique que vous souhaitez protéger.

Utilisations d'Amazon Bedrock avec votre cluster de bases de données Aurora PostgreSQL

Pour Aurora PostgreSQL, le machine learning Aurora fournit la fonction Amazon Bedrock suivante pour travailler avec vos données texte. Cette fonction n'est disponible qu'après avoir installé l'extension `aws_ml` 2.0 et effectué toutes les procédures de configuration. Pour plus d'informations, consultez Configuration de votre cluster de bases de données Aurora PostgreSQL de façon à utiliser le machine learning Aurora.

aws_bedrock.invoke_model

Cette fonction prend en entrée un texte au format JSON, le traite pour différents modèles hébergés sur Amazon Bedrock, puis renvoie la réponse du modèle sous forme de texte JSON. Cette réponse peut contenir du texte, une image ou des vectorisations. Voici un résumé de la documentation de la fonction.

```
aws_bedrock.invoke_model(
   IN model_id      varchar,
   IN content_type  text,
   IN accept_type   text,
   IN model_input   text,
   OUT model_output varchar)
```

Les entrées et sorties de cette fonction sont les suivantes.

- `model_id` : identifiant du modèle.
- `content_type` : le type de demande adressée au modèle Bedrock.
- `accept_type` : le type de réponse à attendre du modèle Bedrock. Généralement application/JSON pour la plupart des modèles.
- `model_input` : invites ; ensemble spécifique d'entrées pour le modèle au format spécifié par content_type. Pour plus d'informations sur la demande acceptée format/structure par le modèle, voir Paramètres d'inférence pour les modèles de base.
- `model_output` : la sortie du modèle Bedrock sous forme de texte.

L’exemple suivant montre comment invoquer un modèle Anthropic Claude 2 pour Bedrock en utilisant invoke_model.

Example Exemple : une requête simple utilisant les fonctions Amazon Bedrock

```
SELECT aws_bedrock.invoke_model (
    model_id    := 'anthropic.claude-v2',
    content_type:= 'application/json',
    accept_type := 'application/json',
    model_input := '{"prompt": "\n\nHuman: You are a helpful assistant that answers
 questions directly and only using the information provided in the context below.
\nDescribe the answer
    in detail.\n\nContext: %s \n\nQuestion: %s \n
\nAssistant:","max_tokens_to_sample":4096,"temperature":0.5,"top_k":250,"top_p":0.5,"stop_seque
[]}'
);
```

aws_bedrock.invoke_model_get_embeddings

La sortie du modèle peut indiquer des vectorisations vectorielles dans certains cas. Étant donné que la réponse varie selon le modèle, il est possible d’utiliser une autre fonction, invoke_model_get_embeddings, qui fonctionne exactement comme invoke_model, mais renvoie les vectorisations en spécifiant la clé JSON appropriée.

```
aws_bedrock.invoke_model_get_embeddings(
   IN model_id      varchar,
   IN content_type  text,
   IN json_key      text,
   IN model_input   text,
   OUT model_output float8[])
```

Les entrées et sorties de cette fonction sont les suivantes.

- `model_id` : identifiant du modèle.
- `content_type` : le type de demande adressée au modèle Bedrock. Ici, accept_type est défini sur la valeur par défaut `application/json`.
- `model_input` : invites ; ensemble spécifique d’entrées pour le modèle au format spécifié par content_type. Pour plus d'informations sur la demande acceptée format/structure par le modèle, voir Paramètres d'inférence pour les modèles de base.

- `json_key` : référence au champ à partir duquel la vectorisation doit être extraite. Cela peut varier si le modèle de vectorisation change.
- `model_output` : la sortie du modèle Bedrock est un ensemble de vectorisations comportant des décimales de 16 bits.

L'exemple suivant montre comment générer une intégration à l'aide du modèle Titan Embeddings G1 — Text embedding pour l'expression PostgreSQL monitoring views. I/O

Example Exemple : une requête simple utilisant les fonctions Amazon Bedrock

```
SELECT aws_bedrock.invoke_model_get_embeddings(
   model_id      := 'amazon.titan-embed-text-v1',
   content_type  := 'application/json',
   json_key      := 'embedding',
   model_input   := '{ "inputText": "PostgreSQL I/O monitoring views"}') AS embedding;
```

Utiliser Amazon Comprehend avec votre cluster de bases de données Aurora PostgreSQL

Pour Aurora PostgreSQL, le machine learning Aurora fournit la fonction Amazon Comprehend suivante pour travailler avec vos données texte. Cette fonction n'est disponible qu'après avoir installé l'extension `aws_ml` et effectué toutes les procédures de configuration. Pour plus d'informations, consultez Configuration de votre cluster de bases de données Aurora PostgreSQL de façon à utiliser le machine learning Aurora.

aws_comprehend.detect_sentiment

Cette fonction prend du texte en entrée et évalue si le texte a une posture émotionnelle positive, négative, neutre ou mixte. Il produit ce sentiment ainsi qu'un niveau de confiance pour son évaluation. Voici un résumé de la documentation de la fonction.

```
aws_comprehend.detect_sentiment(
   IN input_text varchar,
   IN language_code varchar,
   IN max_rows_per_batch int,
   OUT sentiment varchar,
   OUT confidence real)
```

Les entrées et sorties de cette fonction sont les suivantes.

- `input_text` – Le texte pour évaluer et attribuer un sentiment (négatif, positif, neutre, mixte).
- `language_code` : la langue du `input_text` identifié à l'aide de l'identifiant à deux lettres ISO 639-1 avec une sous-étiquette régionale (selon les besoins) ou du code à trois lettres ISO 639-2, selon le cas. Par exemple, `en` est le code pour l'anglais, `zh` est le code pour le chinois simplifié. Pour plus d'informations, consultez Langues prises en charge dans le Guide du développeur Amazon Comprehend.
- `max_rows_per_batch` – Le nombre maximal de lignes par lot pour le traitement par lots. Pour plus d'informations, consultez Présentation du mode par lots et des fonctions de machine learning d'Aurora.
- `sentiment` : l'impression du texte de saisie, identifié comme étant POSITIVE, NEGATIVE, NEUTRAL ou MIXED.
- `confidence` – Le degré de fiabilité de la précision du `sentiment` spécifié. Les valeurs vont de 0,0 à 1,0.

Voici des exemples qui montrent comment utiliser cette fonction.

Example Exemple : une requête simple utilisant les fonctions Amazon Comprehend

Voici un exemple de requête simple qui fait appel à cette fonction pour évaluer la satisfaction des clients auprès de votre équipe d'assistance. Supposons que vous disposiez d'une table de base de données (`support`) qui enregistre les commentaires des clients après chaque demande d'aide. Cet exemple de requête applique la fonction `aws_comprehend.detect_sentiment` au texte de la colonne `feedback` du tableau et génère le sentiment et le niveau de confiance de ce sentiment. Cette requête génère également des résultats par ordre décroissant.

```
SELECT feedback, s.sentiment,s.confidence
    FROM support,aws_comprehend.detect_sentiment(feedback, 'en') s
    ORDER BY s.confidence DESC;
 feedback                          | sentiment | confidence
 ------------------------------------------------------------+-----------+------------
 Thank you for the excellent customer support!              | POSITIVE  |   0.999771
 The latest version of this product stinks!                 | NEGATIVE  |   0.999184
 Your support team is just awesome! I am blown away.        | POSITIVE  |   0.997774
 Your product is too complex, but your support is great.    | MIXED     |   0.957958
 Your support tech helped me in fifteen minutes.            | POSITIVE  |   0.949491
 My problem was never resolved!                             | NEGATIVE  |   0.920644
```

```
 When will the new version of this product be released?   | NEUTRAL   |   0.902706
 I cannot stand that chatbot.                             | NEGATIVE  |   0.895219
 Your support tech talked down to me.                     | NEGATIVE  |   0.868598
 It took me way too long to get a real person.            | NEGATIVE  |   0.481805

 (10 rows)
```

Pour éviter qu'une détection de sentiment vous soit facturée plusieurs fois par ligne de table, vous pouvez matérialiser les résultats. Faites-le sur les lignes qui vous intéressent. Par exemple, les notes du clinicien sont mises à jour afin que seules celles en français (`fr`) utilisent la fonction de détection des sentiments.

```
UPDATE clinician_notes
SET sentiment = (aws_comprehend.detect_sentiment (french_notes, 'fr')).sentiment,
    confidence = (aws_comprehend.detect_sentiment (french_notes, 'fr')).confidence
WHERE
    clinician_notes.french_notes IS NOT NULL AND
    LENGTH(TRIM(clinician_notes.french_notes)) > 0 AND
    clinician_notes.sentiment IS NULL;
```

Pour plus d'informations sur l'optimisation de vos appels de fonction, consultez Considérations sur les performances du machine learning Aurora avec Aurora PostgreSQL.

Utilisation de l' SageMaker IA avec votre cluster de base de données Aurora PostgreSQL

Après avoir configuré votre environnement d' SageMaker IA et intégré Aurora PostgreSQL comme indiqué Configuration d'Aurora PostgreSQL pour utiliser Amazon AI SageMaker dans la section, vous pouvez appeler des opérations à l'aide de la fonction. `aws_sagemaker.invoke_endpoint` La fonction `aws_sagemaker.invoke_endpoint` permet uniquement une connexion à un point de terminaison de modèle se trouvant dans la même Région AWS. Si votre instance de base de données comporte des répliques en plusieurs Régions AWS exemplaires, assurez-vous de configurer et de déployer chaque modèle d' SageMaker IA sur chaque Région AWS.

Les appels à `aws_sagemaker.invoke_endpoint` destination sont authentifiés à l'aide du rôle IAM que vous avez configuré pour associer votre cluster de base de données Aurora PostgreSQL au service SageMaker AI et au point de terminaison que vous avez fournis lors du processus de configuration. SageMaker Les points de terminaison du modèle d'IA sont limités à un compte

individuel et ne sont pas publics. L'`endpoint_name`URL ne contient pas l'identifiant du compte. SageMaker L'IA détermine l'ID du compte à partir du jeton d'authentification fourni par le rôle SageMaker AI IAM de l'instance de base de données.

aws_sagemaker.invoke_endpoint

Cette fonction prend le point de terminaison SageMaker AI en entrée et le nombre de lignes à traiter par lots. Il prend également en entrée les différents paramètres attendus par le point de terminaison du modèle d' SageMaker IA. La documentation de référence de cette fonction est la suivante.

```
aws_sagemaker.invoke_endpoint(
  IN endpoint_name varchar,
  IN max_rows_per_batch int,
  VARIADIC model_input "any",
  OUT model_output varchar
  )
```

Les entrées et sorties de cette fonction sont les suivantes.

- `endpoint_name`— Une URL de point de terminaison Région AWS indépendante.
- `max_rows_per_batch` – Le nombre maximal de lignes par lot pour le traitement par lots. Pour plus d'informations, consultez Présentation du mode par lots et des fonctions de machine learning d'Aurora.
- `model_input` : un ou plusieurs paramètres d'entrée pour le modèle. Il peut s'agir de n'importe quel type de données requis par le modèle d' SageMaker IA. PostgreSQL vous permet de spécifier jusqu'à 100 paramètres d'entrée pour une fonction. Les types de données des tableaux doivent être unidimensionnels, mais peuvent contenir autant d'éléments que prévu par le modèle d' SageMaker IA. Le nombre d'entrées d'un modèle d' SageMaker IA est limité uniquement par la limite de taille des messages SageMaker AI de 6 Mo.
- `model_output`— La sortie du modèle d' SageMaker IA sous forme de texte.

Création d'une fonction définie par l'utilisateur pour invoquer un modèle d' SageMaker IA

Créez une fonction distincte définie par l'utilisateur à appeler `aws_sagemaker.invoke_endpoint` pour chacun de vos modèles d' SageMaker IA. Votre fonction définie par l'utilisateur

représente le point de terminaison SageMaker AI hébergeant le modèle. La fonction `aws_sagemaker.invoke_endpoint` s'exécute dans la fonction définie par l'utilisateur. Les fonctions définies par l'utilisateur offrent de nombreux avantages :

- Vous pouvez donner son propre nom à votre modèle d' SageMaker IA au lieu de vous contenter de faire appel `aws_sagemaker.invoke_endpoint` à tous vos modèles d' SageMaker IA.
- Vous pouvez spécifier l'URL du point de terminaison du modèle en un seul endroit dans votre code d'application SQL.
- Vous pouvez contrôler les privilèges `EXECUTE` de chaque fonction de machine learning Aurora indépendamment.
- Vous pouvez déclarer les types d'entrée et de sortie du modèle à l'aide des types SQL. SQL impose le nombre et le type d'arguments transmis à votre modèle d' SageMaker IA et effectue une conversion de type si nécessaire. L'utilisation de types SQL se `SQL NULL` traduira également par la valeur par défaut appropriée attendue par votre modèle d' SageMaker IA.
- Vous pouvez réduire la taille maximale de lot si vous souhaitez retourner les premières lignes un peu plus rapidement.

Pour spécifier une fonction définie par l'utilisateur, utilisez l'instruction DDL (Data Definition Language) SQL `CREATE FUNCTION`. Lorsque vous créez cette fonction, vous spécifiez les éléments suivants :

- Paramètres d'entrée du modèle.
- Le point de terminaison d' SageMaker IA spécifique à invoquer.
- Type de retour.

La fonction définie par l'utilisateur renvoie l'inférence calculée par le point de terminaison SageMaker AI après avoir exécuté le modèle sur les paramètres d'entrée. L'exemple suivant crée une fonction définie par l'utilisateur pour un modèle d' SageMaker IA avec deux paramètres d'entrée.

```
CREATE FUNCTION classify_event (IN arg1 INT, IN arg2 DATE, OUT category INT)
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
        'sagemaker_model_endpoint_name', NULL,
        arg1, arg2                        -- model inputs are separate arguments
        )::INT                            -- cast the output to INT
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Notez ce qui suit :

- L'entrée de la fonction `aws_sagemaker.invoke_endpoint` peut être constituée par un ou plusieurs paramètres de n'importe quel type de données.
- Cet exemple utilise un type de sortie INT. Si vous faites passer la sortie d'un type `varchar` à un autre type, elle doit être convertie en un type scalaire intégré PostgreSQL tel que `INTEGER`, `REAL`, `FLOAT` ou `NUMERIC`. Pour plus d'informations sur ces types, consultez Date Types dans la documentation PostgreSQL.
- Spécifiez `PARALLEL SAFE` pour activer le traitement de requêtes parallèles. Pour plus d'informations, consultez Améliorer les temps de réponse grâce au traitement parallèle des requêtes.
- Spécifiez `COST 5000` pour estimer le coût d'exécution de la fonction. Utilisez un nombre positif donnant le coût d'exécution estimé de la fonction, en unités de `cpu_operator_cost`.

Transmission d'un tableau en entrée à un modèle d' SageMaker IA

La fonction `aws_sagemaker.invoke_endpoint` peut avoir jusqu'à 100 paramètres d'entrée, ce qui est la limite pour les fonctions PostgreSQL. Si le modèle d' SageMaker IA nécessite plus de 100 paramètres du même type, transmettez-les sous forme de tableau.

L'exemple suivant définit une fonction qui transmet un tableau en entrée au modèle de régression SageMaker AI. La sortie est convertie à une valeur `REAL`.

```
CREATE FUNCTION regression_model (params REAL[], OUT estimate REAL)
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
      'sagemaker_model_endpoint_name',
      NULL,
      params
      )::REAL
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Spécification de la taille du lot lors de l'appel d'un modèle d' SageMaker IA

L'exemple suivant crée une fonction définie par l'utilisateur pour un modèle d' SageMaker IA qui définit la taille du lot par défaut sur NULL. La fonction vous permet également de fournir une taille de lot différente lorsque vous l'invoquez.

```
CREATE FUNCTION classify_event (
```

```
    IN event_type INT, IN event_day DATE, IN amount REAL, -- model inputs
    max_rows_per_batch INT DEFAULT NULL,  -- optional batch size limit
    OUT category INT)                     -- model output
AS $$
    SELECT aws_sagemaker.invoke_endpoint (
        'sagemaker_model_endpoint_name', max_rows_per_batch,
        event_type, event_day, COALESCE(amount, 0.0)
        )::INT              -- casts output to type INT
$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Remarques :

- Utilisez le paramètre `max_rows_per_batch` facultatif pour contrôler le nombre de lignes pour une invocation de fonction en mode traitement par lots. Si vous utilisez une valeur NULL, l'optimiseur de requête choisit automatiquement la taille de lot maximale. Pour de plus amples informations, veuillez consulter Présentation du mode par lots et des fonctions de machine learning d'Aurora.
- Par défaut, la transmission de NULL en tant que valeur de paramètre est traduite en chaîne vide avant d'être transmise à SageMaker AI. Pour cet exemple, les entrées ont différents types.
- Si vous avez une entrée non textuelle ou une entrée de texte qui doit par défaut avoir une valeur autre qu'une chaîne vide, utilisez l'instruction `COALESCE`. Utilisez `COALESCE` pour traduire NULL en la valeur de remplacement nulle souhaitée dans l'appel à `aws_sagemaker.invoke_endpoint`. Pour le paramètre `amount` de cet exemple, une valeur NULL est convertie en 0.0.

Invoquer un modèle d' SageMaker IA doté de plusieurs sorties

L'exemple suivant crée une fonction définie par l'utilisateur pour un modèle d' SageMaker IA qui renvoie plusieurs sorties. Votre fonction doit convertir la sortie de la fonction `aws_sagemaker.invoke_endpoint` en un type de données correspondant. Par exemple, vous pouvez utiliser le type de point PostgreSQL intégré pour les paires (x, y) ou un type composite défini par l'utilisateur.

Cette fonction définie par l'utilisateur renvoie des valeurs à partir d'un modèle renvoyant plusieurs sorties à l'aide d'un type composite pour les sorties.

```
CREATE TYPE company_forecasts AS (
    six_month_estimated_return real,
    one_year_bankruptcy_probability float);
CREATE FUNCTION analyze_company (
    IN free_cash_flow NUMERIC(18, 6),
```

```
    IN debt NUMERIC(18,6),
    IN max_rows_per_batch INT DEFAULT NULL,
    OUT prediction company_forecasts)
AS $$
    SELECT (aws_sagemaker.invoke_endpoint('endpt_name',
       max_rows_per_batch,free_cash_flow, debt))::company_forecasts;


$$ LANGUAGE SQL PARALLEL SAFE COST 5000;
```

Pour le type composite, utilisez les champs dans le même ordre que celui dans lequel ils apparaissent dans la sortie du modèle et convertissez la sortie `aws_sagemaker.invoke_endpoint` en votre type composite. L'appelant peut extraire les champs individuels soit par leur nom, soit avec la notation PostgreSQL « .* ».

Exportation de données vers Amazon S3 pour la formation de modèles d' SageMaker IA (niveau avancé)

Nous vous recommandons de vous familiariser avec l'apprentissage automatique et l' SageMaker IA d'Aurora en utilisant les algorithmes et les exemples fournis plutôt que d'essayer de former vos propres modèles. Pour plus d'informations, consultez Get Started with Amazon SageMaker AI

Pour entraîner des modèles d' SageMaker IA, vous exportez des données vers un compartiment Amazon S3. Le compartiment Amazon S3 est utilisé par l' SageMaker IA pour entraîner votre modèle avant son déploiement. Vous pouvez interroger les données d'un cluster de bases de données Aurora PostgreSQL et les enregistrer directement dans des fichiers texte stockés dans un compartiment Amazon S3. L' SageMaker IA consomme ensuite les données du compartiment Amazon S3 à des fins d'entraînement. Pour en savoir plus sur la formation de modèles d' SageMaker IA, consultez Entraînez un modèle avec Amazon SageMaker AI.

> **Note**
>
> Lorsque vous créez un compartiment Amazon S3 pour l'entraînement des modèles d' SageMaker IA ou la notation par lots, `sagemaker` utilisez-le dans le nom du compartiment Amazon S3. Pour plus d'informations, consultez Spécifier un compartiment Amazon S3 pour télécharger des ensembles de données d'entraînement et stocker les données de sortie dans le manuel Amazon SageMaker AI Developer Guide.

Pour plus d'informations sur l'exportation de vos données, consultez Exportation de données à partir d'un cluster de bases de données Aurora PostgreSQL vers Amazon S3.

Considérations sur les performances du machine learning Aurora avec Aurora PostgreSQL

Les services Amazon Comprehend et SageMaker AI effectuent la majeure partie du travail lorsqu'ils sont invoqués par une fonction d'apprentissage automatique Aurora. Cela signifie que vous pouvez adapter ces ressources selon vos besoins, de manière indépendante. Pour votre cluster de bases de données Aurora PostgreSQL, vous pouvez rendre vos appels de fonctions aussi efficaces que possible. Vous trouverez ci-dessous quelques considérations relatives aux performances à prendre en compte lors de l'utilisation du machine learning Aurora depuis Aurora PostgreSQL.

Rubriques

Présentation du mode par lots et des fonctions de machine learning d'Aurora

Généralement, PostgreSQL exécute les fonctions une ligne à la fois. Le machine learning Aurora peut réduire cette surcharge en combinant en lots les appels au service de machine learning Aurora externe pour de nombreuses lignes avec une approche appelée exécution en mode traitement par lots. En mode traitement par lots, le machine learning Aurora reçoit les réponses d'un lot de lignes d'entrée, puis retransmet les réponses à la requête en cours d'exécution une ligne à la fois. Cette optimisation améliore le débit de vos requêtes Aurora sans limiter l'optimiseur de requêtes PostgreSQL.

Aurora utilise automatiquement le mode traitement par lots si la fonction est référencée à partir de la liste `SELECT`, d'une clause `WHERE` ou d'une clause `HAVING`. Notez que les expressions `CASE` simples de niveau supérieur sont éligibles à l'exécution en mode traitement par lots. Les expressions `CASE` recherchées de niveau supérieur sont également éligibles à l'exécution en mode traitement par lots à condition que la première clause `WHEN` soit un prédicat simple avec un appel de fonction en mode traitement par lots.

Votre fonction définie par l'utilisateur doit être une fonction `LANGUAGE SQL` et doit spécifier `PARALLEL SAFE` et `COST 5000`.

Migration de fonction de l'instruction SELECT vers la clause FROM

Habituellement, une fonction `aws_ml` éligible à l'exécution en mode traitement par lots est automatiquement migrée par Aurora vers la clause `FROM`.

La migration des fonctions en mode traitement par lots éligibles vers la clause FROM peut être examinée manuellement au niveau de chaque requête. Pour ce faire, vous utilisez les instructions EXPLAIN (et ANALYSE et VERBOSE) et vous recherchez les informations « Batch Processing (Traitement par lots) » sous chaque en mode traitement par lot `Function Scan`. Vous pouvez également utiliser EXPLAIN (avec VERBOSE) sans exécuter la requête. Vous observez ensuite si les appels à la fonction apparaissent sous la forme `Function Scan` sous une jointure de boucle imbriquée qui n'a pas été spécifiée dans l'instruction d'origine.

Dans l'exemple suivant, l'opérateur de jointure de boucle imbriquée dans le plan montre que Aurora a migré la fonction `anomaly_score`. Il a migré cette fonction de la liste SELECT vers la clause FROM, où elle est éligible à l'exécution en mode traitement par lots.

```
EXPLAIN (VERBOSE, COSTS false)
SELECT anomaly_score(ts.R.description) from ts.R;
                         QUERY PLAN
-------------------------------------------------------------
 Nested Loop
   Output: anomaly_score((r.description)::text)
   ->  Seq Scan on ts.r
         Output: r.id, r.description, r.score
   ->  Function Scan on public.anomaly_score
         Output: anomaly_score.anomaly_score
         Function Call: anomaly_score((r.description)::text)
```

Pour désactiver l'exécution en mode traitement par lots, définissez le paramètre `apg_enable_function_migration` sur `false`. Cela empêche la migration des fonctions `aws_ml` de la liste SELECT vers la clause `FROM`. L'exemple suivant indique comment procéder.

```
SET apg_enable_function_migration = false;
```

Le paramètre `apg_enable_function_migration` est un paramètre GUC (Grand Unified Configuration) reconnu par l'extension Aurora PostgreSQL `apg_plan_mgmt` pour la gestion des plans de requêtes. Pour désactiver la migration des fonctions dans une session, utilisez la gestion des plans de requêtes pour enregistrer le plan résultant en tant que plan `approved`. Lors

de l'exécution, la gestion des plans de requêtes applique le plan `approved` avec son paramètre `apg_enable_function_migration`. Cette application se produit indépendamment de la valeur du paramètre GUC `apg_enable_function_migration`. Pour plus d'informations, consultez Gestion des plans d'exécution de requêtes pour Aurora PostgreSQL.

Utilisation du paramètre max_rows_per_batch

Les fonctions `aws_comprehend.detect_sentiment` et `aws_sagemaker.invoke_endpoint` ont toutes deux un paramètre `max_rows_per_batch`. Ce paramètre indique le nombre de lignes qui peuvent être envoyées au service de machine learning Aurora. Plus le jeu de données traité par votre fonction est grand, plus vous pouvez augmenter la taille du lot.

Les fonctions en mode traitement par lots améliorent l'efficacité en créant des lots de lignes qui répartissent le coût des appels de fonction du machine learning Aurora sur un grand nombre de lignes. Toutefois, si une instruction `SELECT` se termine prématurément en raison d'une clause `LIMIT`, le lot peut être construit sur plus de lignes que la requête n'en utilise. Cette approche peut entraîner des frais supplémentaires sur votre AWS compte. Pour profiter des avantages de l'exécution en mode traitement par lots tout en évitant de créer des lots trop volumineux, utilisez une valeur plus petite pour le paramètre `max_rows_per_batch` dans vos appels de fonction.

Si vous effectuez une action `EXPLAIN` (`VERBOSE`, `ANALYZE`) sur une requête qui utilise l'exécution en mode traitement par lots, vous voyez un opérateur `FunctionScan` qui se trouve sous une jointure de boucle imbriquée. Le nombre de boucles rapporté par `EXPLAIN` équivaut au nombre de fois où une ligne a été extraite à partir de l'opérateur `FunctionScan`. Si une instruction utilise une clause LIMIT, le nombre d'extractions est cohérent. Pour optimiser la taille du lot, définissez le paramètre `max_rows_per_batch` sur cette valeur. Cependant, si la fonction de mode traitement par lots est référencée dans un prédicat dans la clause `WHERE` ou `HAVING`, vous ne pouvez probablement pas connaître le nombre d'extractions à l'avance. Dans ce cas, utilisez les boucles comme guide et testez l'élément `max_rows_per_batch` pour trouver un paramètre optimisant les performances.

Vérification de l'exécution en mode traitement par lots

Pour voir si une fonction a été exécutée en mode batch, utilisez `EXPLAIN ANALYZE`. Si l'exécution en mode traitement par lots a été utilisée, le plan de requêtes inclura les informations dans une section « Batch Processing (Traitement par lots) ».

```
EXPLAIN ANALYZE SELECT user-defined-function();
 Batch Processing: num batches=1 avg/min/max batch size=3333.000/3333.000/3333.000
                                 avg/min/max batch call time=146.273/146.273/146.273
```

Dans cet exemple, 1 lot contenait 3 333 lignes, dont le traitement a duré 146,273 ms. La section « Batch Processing (Traitement par lots) » contient les éléments suivants :

- Le nombre de lots pour cette opération d'analyse de fonction
- La taille moyenne, minimale et maximale du lot
- La durée moyenne, minimale et maximale d'exécution du lot

En général, le lot final est plus petit que le reste, ce qui entraîne souvent une taille de lot minimale bien inférieure à la taille moyenne.

Pour renvoyer les premières lignes plus rapidement, définissez le paramètre `max_rows_per_batch` sur une valeur plus petite.

Pour réduire le nombre d'appels en mode traitement par lots au service ML lorsque vous utilisez une clause `LIMIT` dans votre fonction définie par l'utilisateur, définissez le paramètre `max_rows_per_batch` sur une valeur plus petite.

Améliorer les temps de réponse grâce au traitement parallèle des requêtes

Pour obtenir des résultats le plus rapidement possible à partir d'un grand nombre de lignes, vous pouvez combiner le traitement des requêtes parallèles avec le traitement par lots. Vous pouvez utiliser le traitement des requêtes parallèles pour les instructions `SELECT`, `CREATE TABLE AS SELECT` et `CREATE MATERIALIZED VIEW`.

> **Note**
>
> PostgreSQL ne prend pas encore en charge les requêtes parallèles pour les instructions DML (Data Manipulation Language).

Le traitement des requêtes parallèles se produit à la fois dans la base de données et dans le service ML. Le nombre de cœurs dans la classe d'instance de la base de données limite le degré de parallélisme pouvant être utilisé lors de l'exécution d'une requête. Le serveur de base de données peut construire un plan d'exécution de requêtes parallèles qui partitionne la tâche entre un ensemble d'unités de travail parallèles. Ensuite, chacune de ces unités de travail peut créer des requêtes par lots contenant des dizaines de milliers de lignes (ou autant que ce qui est autorisé par chaque service).

Les demandes groupées provenant de tous les travailleurs parallèles sont envoyées au point de terminaison SageMaker AI. Le degré de parallélisme que le point de terminaison peut prendre en charge est limité par le nombre et le type d'instances qui le prennent en charge. Pour obtenir K degrés de parallélisme, vous avez besoin d'une classe d'instance de base de données ayant au moins K cœurs. Vous devez également configurer le point de terminaison SageMaker AI de votre modèle pour qu'il comporte K instances initiales d'une classe d'instance suffisamment performante.

Pour utiliser le traitement des requêtes parallèles, vous pouvez définir le paramètre de stockage `parallel_workers` de la table qui contient les données que vous prévoyez de transmettre. Vous définissez `parallel_workers` sur une fonction en mode traitement par lots telle que `aws_comprehend.detect_sentiment`. Si l'optimiseur choisit un plan de requête parallèle, les services AWS ML peuvent être appelés à la fois par lots et en parallèle.

Vous pouvez utiliser les paramètres suivants avec la fonction `aws_comprehend.detect_sentiment` pour obtenir un plan avec un parallélisme à quatre voies. Si vous modifiez l'un des deux paramètres suivants, vous devez redémarrer l'instance de base de données pour que les modifications soient effectives

```
-- SET max_worker_processes to 8;  -- default value is 8
-- SET max_parallel_workers to 8;  -- not greater than max_worker_processes
SET max_parallel_workers_per_gather to 4;  -- not greater than max_parallel_workers

-- You can set the parallel_workers storage parameter on the table that the data
-- for the Aurora machine learning function is coming from in order to manually
 override the degree of
-- parallelism that would otherwise be chosen by the query optimizer
--
ALTER TABLE yourTable SET (parallel_workers = 4);

-- Example query to exploit both batch-mode execution and parallel query
EXPLAIN (verbose, analyze, buffers, hashes)
SELECT aws_comprehend.detect_sentiment(description, 'en')).*
FROM yourTable
WHERE id < 100;
```

Pour plus d'informations sur le contrôle des requêtes parallèles, consultez Plans parallélisés dans la documentation PostgreSQL.

Utilisation des vues matérialisées et des colonnes matérialisées

Lorsque vous invoquez un AWS service tel qu' SageMaker AI ou Amazon Comprehend depuis votre base de données, votre compte est débité conformément à la politique tarifaire de ce service. Pour minimiser les frais sur votre compte, vous pouvez matérialiser le résultat de l'appel du AWS service dans une colonne matérialisée afin que le AWS service ne soit pas appelé plus d'une fois par ligne de saisie. Si vous le souhaitez, vous pouvez ajouter une colonne d'horodatage `materializedAt` pour enregistrer l'heure à laquelle les colonnes ont été matérialisées.

La latence d'une instruction `INSERT` ordinaire à une seule ligne est généralement beaucoup moins élevée que celle de l'appel d'une fonction en mode traitement par lots. Ainsi, vous risquez de ne pas être en mesure de répondre aux exigences de latence de votre application si vous appelez la fonction en mode traitement par lots pour chaque `INSERT` d'une seule ligne exécuté par votre application. Pour matérialiser le résultat de l'appel d'un AWS service dans une colonne matérialisée, les applications hautes performances doivent généralement remplir les colonnes matérialisées. Pour ce faire, elles émettent périodiquement une instruction `UPDATE` qui s'exécute sur un grand lot de lignes en même temps.

`UPDATE` applique un verrou au niveau de la ligne qui peut avoir un impact sur une application en cours d'exécution. Donc, vous pouvez avoir besoin d'utiliser `SELECT ... FOR UPDATE SKIP LOCKED` ou `MATERIALIZED VIEW`.

Les requêtes analytiques qui s'exécutent sur un grand nombre de lignes en temps réel peuvent combiner la matérialisation en mode traitement par lots et le traitement en temps réel. Pour ce faire, ces requêtes rassemblent sous la forme d'une opération `UNION ALL` les résultats prématérialisés avec une requête sur les lignes qui n'ont pas encore de résultats matérialisés. Dans certains cas, une telle opération `UNION ALL` est nécessaire à plusieurs endroits ou la requête peut être générée par une application tierce. Si c'est le cas, vous pouvez créer un élément `VIEW` pour encapsuler l'opération `UNION ALL` afin que ce détail ne soit pas exposé au reste de l'application SQL.

Vous pouvez utiliser une vue matérialisée pour matérialiser les résultats d'une instruction `SELECT` arbitraire à un moment dans le temps. Vous pouvez également l'utiliser pour actualiser la vue matérialisée à tout moment dans le futur. Actuellement, PostgreSQL ne prend pas en charge l'actualisation incrémentielle. Chaque fois que la vue matérialisée est actualisée, elle est entièrement recalculée.

Vous pouvez actualiser les vues matérialisées avec l'option `CONCURRENTLY`, qui met à jour le contenu de la vue matérialisée sans appliquer de verrou exclusif. Cela permet à une application SQL de lire la vue matérialisée pendant qu'elle est actualisée.

Surveillance du machine learning Aurora

Vous pouvez surveiller les fonctions `aws_ml` en définissant le paramètre `track_functions` de votre groupe de paramètres de cluster de bases de données personnalisé sur `all`. Par défaut, ce paramètre est défini sur `pl`, ce qui signifie que seules les fonctions du langage de procédure sont suivies. En le remplaçant par `all`, les fonctions `aws_ml` sont également suivies. Pour plus d'informations, consultez Statistiques d'exécution dans la documentation PostgreSQL.

Pour plus d'informations sur la surveillance des performances des opérations d' SageMaker IA appelées par les fonctions d'apprentissage automatique Aurora, consultez la section Monitor Amazon SageMaker AI dans le manuel Amazon SageMaker AI Developer Guide.

Avec `track_functions` défini sur `all`, vous pouvez interroger la vue `pg_stat_user_functions` pour obtenir des statistiques sur les fonctions que vous définissez et utilisez pour appeler les services de machine learning Aurora. Pour chaque fonction, la vue inclut le nombre de `calls`, `total_time` et `self_time`.

Pour consulter les statistiques des fonctions `aws_sagemaker.invoke_endpoint` et `aws_comprehend.detect_sentiment`, vous pouvez filtrer les résultats par nom de schéma à l'aide de la requête suivante.

```
SELECT * FROM pg_stat_user_functions
    WHERE schemaname
    LIKE 'aws_%';
```

Pour effacer les statistiques, procédez comme suit.

```
SELECT pg_stat_reset();
```

Vous pouvez obtenir les noms de vos fonctions SQL qui appellent la fonction `aws_sagemaker.invoke_endpoint` en interrogeant le catalogue du système `pg_proc` de PostgreSQL. Ce catalogue contient des informations sur les fonctions, les procédures et plus encore. Pour plus d'informations, consultez pg_proc dans la documentation PostgreSQL. Voici un exemple d'interrogation de la table pour obtenir les noms des fonctions (`proname`) dont la source (`prosrc`) inclut le texte invoke_endpoint.

```
SELECT proname FROM pg_proc WHERE prosrc LIKE '%invoke_endpoint%';
```

Exemples de code pour Aurora à l'aide d'Aurora AWS SDKs

Les exemples de code suivants montrent comment utiliser Aurora avec un kit de développement AWS logiciel (SDK).

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Mise en route

Bonjour Aurora

Les exemples de code suivants montrent comment bien démarrer avec Aurora.

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
```

```
using Microsoft.Extensions.Hosting;

namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
        {
            Console.WriteLine($"\tNo clusters found.");
        }
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Code pour le CMake fichier CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
```

```
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code pour le fichier source hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
```

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
```

```
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
```

```
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import boto3

# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
```

```
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Pour plus de détails sur l'API, voir AWSDescribe DBClusters in SDK for Python (Boto3) API Reference.

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)
```

```
# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
```

```
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Pour plus de détails sur l'API, consultez la section Description DBClusters dans le AWSSDK pour la référence de l'API Rust.

Exemples de code

- Exemples de base pour l'utilisation d'Aurora AWS SDKs
  - Bonjour Aurora
  - Découvrez les bases d'Aurora avec un AWS SDK
  - Actions pour Aurora utilisant AWS SDKs
    - Utilisation CreateDBCluster avec un AWS SDK
    - Utilisation CreateDBClusterParameterGroup avec un AWS SDK
    - Utilisation CreateDBClusterSnapshot avec un AWS SDK
    - Utilisation CreateDBInstance avec un AWS SDK
    - Utilisation DeleteDBCluster avec un AWS SDK

Exemples de base pour l'utilisation d'Aurora AWS SDKs

Les exemples de code suivants montrent comment utiliser les bases d'Amazon Aurora avec AWS SDKs.

Exemples

- Utilisation DescribeDBClusterParameters avec un AWS SDK
- Utilisation DescribeDBClusterSnapshots avec un AWS SDK
- Utilisation DescribeDBClusters avec un AWS SDK
- Utilisation DescribeDBEngineVersions avec un AWS SDK
- Utilisation DescribeDBInstances avec un AWS SDK
- Utilisation DescribeOrderableDBInstanceOptions avec un AWS SDK ou une CLI
- Utilisation ModifyDBClusterParameterGroup avec un AWS SDK

Bonjour Aurora

Les exemples de code suivants montrent comment bien démarrer avec Aurora.

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

namespace AuroraActions;

public static class HelloAurora
{
    static async Task Main(string[] args)
    {
        // Use the AWS .NET Core Setup package to set up dependency injection for
 the
        // Amazon Relational Database Service (Amazon RDS).
        // Use your AWS profile name, or leave it blank to use the default
 profile.
```

```
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
            ).Build();

        // Now the client is available for injection. Fetching it directly here
 for example purposes only.
        var rdsClient = host.Services.GetRequiredService<IAmazonRDS>();

        // You can use await and any of the async methods to get a response.
        var response = await rdsClient.DescribeDBClustersAsync(new
 DescribeDBClustersRequest { IncludeShared = true });
        Console.WriteLine($"Hello Amazon RDS Aurora! Let's list some clusters in
 this account:");
        if (response.DBClusters == null)
        {
            Console.WriteLine($"\tNo clusters found.");
        }
        else
        {
            foreach (var cluster in response.DBClusters)
            {
                Console.WriteLine(
                    $"\tCluster: database: {cluster.DatabaseName} identifier:
 {cluster.DBClusterIdentifier}.");
            }
        }
    }
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Code pour le CMake fichier CMake Lists.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Code pour le fichier source hello_aurora.cpp.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 *  A "Hello Aurora" starter application which initializes an Amazon Relational
 Database Service (Amazon RDS) client
 *  and describes the Amazon Aurora (Aurora) clusters.
 *
 *  main function
 *
 *  Usage: 'hello_aurora'
 *
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
        Aws::RDS::RDSClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                    rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
                result = 1;
                std::cerr << "Error with Aurora::GDescribeDBClusters. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!marker.empty());

        std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
        for (auto &clusterId: clusterIds) {
            std::cout << "  clusterId " << clusterId << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up
 to 20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 auroraClient := rds.NewFromConfig(sdkConfig)
 const maxClusters = 20
 fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
 output, err := auroraClient.DescribeDBClusters(
  ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
 if err != nil {
```

```
  fmt.Printf("Couldn't list DB clusters: %v\n", err)
  return
 }
 if len(output.DBClusters) == 0 {
  fmt.Println("No DB clusters found.")
 } else {
  for _, cluster := range output.DBClusters {
   fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
    *cluster.DatabaseName)
  }
 }
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.paginators.DescribeDBClustersIterable;

public class DescribeDbClusters {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        describeClusters(rdsClient);
        rdsClient.close();
```

```
    }

    public static void describeClusters(RdsClient rdsClient) {
        DescribeDBClustersIterable clustersIterable =
 rdsClient.describeDBClustersPaginator();
        clustersIterable.stream()
                .flatMap(r -> r.dbClusters().stream())
                .forEach(cluster -> System.out
                        .println("Database name: " + cluster.databaseName() + "
 Arn = " + cluster.dbClusterArn()));
    }
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import boto3

# Create an RDS client
rds = boto3.client("rds")

# Create a paginator for the describe_db_clusters operation
paginator = rds.get_paginator("describe_db_clusters")

# Use the paginator to get a list of DB clusters
response_iterator = paginator.paginate(
    PaginationConfig={
        "PageSize": 50,  # Adjust PageSize as needed
        "StartingToken": None,
    }
```

```
)

# Iterate through the pages of the response
clusters_found = False
for page in response_iterator:
    if "DBClusters" in page and page["DBClusters"]:
        clusters_found = True
        print("Here are your RDS Aurora clusters:")
        for cluster in page["DBClusters"]:
            print(
                f"Cluster ID: {cluster['DBClusterIdentifier']}, Engine:
 {cluster['Engine']}"
            )

if not clusters_found:
    print("No clusters found!")
```

- Pour plus de détails sur l'API, voir AWSDescribe DBClusters in SDK for Python (Boto3) API Reference.

Ruby

Kit SDK pour Ruby

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
require 'aws-sdk-rds'

# Creates an Amazon RDS client for the AWS Region
rds = Aws::RDS::Client.new

puts 'Listing clusters in this AWS account...'

# Calls the describe_db_clusters method to get information about clusters
resp = rds.describe_db_clusters(max_records: 20)
```

```
# Checks if any clusters are found and prints the appropriate message
if resp.db_clusters.empty?
  puts 'No clusters found!'
else
  # Loops through the array of cluster objects and prints the cluster identifier
  resp.db_clusters.each do |cluster|
    puts "Cluster identifier: #{cluster.db_cluster_identifier}"
  end
end
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
use aws_sdk_rds::Client;

#[derive(Debug)]
struct Error(String);
impl std::fmt::Display for Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
impl std::error::Error for Error {}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
```

```
    let client = Client::new(&sdk_config);

    let describe_db_clusters_output = client
        .describe_db_clusters()
        .send()
        .await
        .map_err(|e| Error(e.to_string()))?;
    println!(
        "Found {} clusters:",
        describe_db_clusters_output.db_clusters().len()
    );
    for cluster in describe_db_clusters_output.db_clusters() {
        let name = cluster.database_name().unwrap_or("Unknown");
        let engine = cluster.engine().unwrap_or("Unknown");
        let id = cluster.db_cluster_identifier().unwrap_or("Unknown");
        let class = cluster.db_cluster_instance_class().unwrap_or("Unknown");
        println!("\tDatabase: {name}",);
        println!("\t  Engine: {engine}",);
        println!("\t      ID: {id}",);
        println!("\tInstance: {class}",);
    }

    Ok(())
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le AWSSDK pour la référence de l'API Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Découvrez les bases d'Aurora avec un AWS SDK

Les exemples de code suivants montrent comment :

- Créez un groupe de paramètres pour le cluster de bases de données Aurora personnalisé et définissez des valeurs pour les paramètres.
- Créez un cluster de bases de données qui utilise le groupe de paramètres.
- Créez une instance de base de données qui contient une base de données.

- Prenez un instantané du cluster de bases de données, puis nettoyez les ressources.

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Exécutez un scénario interactif à une invite de commande.

```
using Amazon.RDS;
using Amazon.RDS.Model;
using AuroraActions;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace AuroraScenario;

/// <summary>
/// Scenario for Amazon Aurora examples.
/// </summary>
public class AuroraScenario
{
    /*
    Before running this .NET code example, set up your development environment,
 including your credentials.

    This .NET example performs the following tasks:
    1.  Return a list of the available DB engine families for Aurora MySql using
 the DescribeDBEngineVersionsAsync method.
    2.  Select an engine family and create a custom DB cluster parameter group
 using the CreateDBClusterParameterGroupAsync method.
    3.  Get the parameter group using the DescribeDBClusterParameterGroupsAsync
 method.
```

```
   4.  Get some parameters in the group using the
DescribeDBClusterParametersAsync method.
   5.  Parse and display some parameters in the group.
   6.  Modify the auto_increment_offset and auto_increment_increment parameters
       using the ModifyDBClusterParameterGroupAsync method.
   7.  Get and display the updated parameters using the
DescribeDBClusterParametersAsync method with a source of "user".
   8.  Get a list of allowed engine versions using the
DescribeDBEngineVersionsAsync method.
   9.  Create an Aurora DB cluster that contains a MySql database and uses the
parameter group.
       using the CreateDBClusterAsync method.
   10. Wait for the DB cluster to be ready using the DescribeDBClustersAsync
method.
   11. Display and select from a list of instance classes available for the
selected engine and version
       using the paginated DescribeOrderableDBInstanceOptions method.
   12. Create a database instance in the cluster using the CreateDBInstanceAsync
method.
   13. Wait for the DB instance to be ready using the DescribeDBInstances
method.
   14. Display the connection endpoint string for the new DB cluster.
   15. Create a snapshot of the DB cluster using the
CreateDBClusterSnapshotAsync method.
   16. Wait for DB snapshot to be ready using the
DescribeDBClusterSnapshotsAsync method.
   17. Delete the DB instance using the DeleteDBInstanceAsync method.
   18. Delete the DB cluster using the DeleteDBClusterAsync method.
   19. Wait for DB cluster to be deleted using the DescribeDBClustersAsync
methods.
   20. Delete the cluster parameter group using the
DeleteDBClusterParameterGroupAsync.
   */

   private static readonly string sepBar = new('-', 80);
   private static AuroraWrapper auroraWrapper = null!;
   private static ILogger logger = null!;
   private static readonly string engine = "aurora-mysql";
   static async Task Main(string[] args)
   {
       // Set up dependency injection for the Amazon Relational Database Service
(Amazon RDS).
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
```

```
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonRDS>()
                    .AddTransient<AuroraWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        }).CreateLogger<AuroraScenario>();

        auroraWrapper = host.Services.GetRequiredService<AuroraWrapper>();

        Console.WriteLine(sepBar);
        Console.WriteLine(
            "Welcome to the Amazon Aurora: get started with DB clusters
example.");
        Console.WriteLine(sepBar);

        DBClusterParameterGroup parameterGroup = null!;
        DBCluster? newCluster = null;
        DBInstance? newInstance = null;

        try
        {
            var parameterGroupFamily = await ChooseParameterGroupFamilyAsync();

            parameterGroup = await
CreateDBParameterGroupAsync(parameterGroupFamily);

            var parameters = await
DescribeParametersInGroupAsync(parameterGroup.DBClusterParameterGroupName,
                new List<string> { "auto_increment_offset",
"auto_increment_increment" });

            await
ModifyParametersAsync(parameterGroup.DBClusterParameterGroupName, parameters);
```

```
            await
DescribeUserSourceParameters(parameterGroup.DBClusterParameterGroupName);

            var engineVersionChoice = await
ChooseDBEngineVersionAsync(parameterGroupFamily);

            var newClusterIdentifier = "Example-Cluster-" + DateTime.Now.Ticks;

            newCluster = await CreateNewCluster
            (
                parameterGroup,
                engine,
                engineVersionChoice.EngineVersion,
                newClusterIdentifier
            );

            var instanceClassChoice = await ChooseDBInstanceClass(engine,
engineVersionChoice.EngineVersion);

            var newInstanceIdentifier = "Example-Instance-" + DateTime.Now.Ticks;

            newInstance = await CreateNewInstance(
                newClusterIdentifier,
                engine,
                engineVersionChoice.EngineVersion,
                instanceClassChoice.DBInstanceClass,
                newInstanceIdentifier
            );

            DisplayConnectionString(newCluster!);
            await CreateSnapshot(newCluster!);
            await CleanupResources(newInstance, newCluster, parameterGroup);

            Console.WriteLine("Scenario complete.");
            Console.WriteLine(sepBar);
        }
        catch (Exception ex)
        {
            await CleanupResources(newInstance, newCluster, parameterGroup);
            logger.LogError(ex, "There was a problem executing the scenario.");
        }
    }

    /// <summary>
```

```
   /// Choose the Aurora DB parameter group family from a list of available
options.
   /// </summary>
   /// <returns>The selected parameter group family.</returns>
   public static async Task<string> ChooseParameterGroupFamilyAsync()
   {
       Console.WriteLine(sepBar);
       // 1. Get a list of available engines.
       var engines = await
auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine);

       Console.WriteLine($"1. The following is a list of available DB parameter
group families for engine {engine}:");

       var parameterGroupFamilies =
           engines.GroupBy(e => e.DBParameterGroupFamily).ToList();
       for (var i = 1; i <= parameterGroupFamilies.Count; i++)
       {
           var parameterGroupFamily = parameterGroupFamilies[i - 1];
           // List the available parameter group families.
           Console.WriteLine(
               $"\t{i}. Family: {parameterGroupFamily.Key}");
       }

       var choiceNumber = 0;
       while (choiceNumber < 1 || choiceNumber > parameterGroupFamilies.Count)
       {
           Console.WriteLine("2. Select an available DB parameter group family
by entering a number from the preceding list:");
           var choice = Console.ReadLine();
           Int32.TryParse(choice, out choiceNumber);
       }
       var parameterGroupFamilyChoice = parameterGroupFamilies[choiceNumber -
1];
       Console.WriteLine(sepBar);
       return parameterGroupFamilyChoice.Key;
   }

   /// <summary>
   /// Create and get information on a DB parameter group.
   /// </summary>
   /// <param name="dbParameterGroupFamily">The DBParameterGroupFamily for the
new DB parameter group.</param>
   /// <returns>The new DBParameterGroup.</returns>
```

```
    public static async Task<DBClusterParameterGroup>
 CreateDBParameterGroupAsync(string dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"2. Create new DB parameter group with family
 {dbParameterGroupFamily}:");

        var parameterGroup = await
 auroraWrapper.CreateCustomClusterParameterGroupAsync(
            dbParameterGroupFamily,
            "ExampleParameterGroup-" + DateTime.Now.Ticks,
            "New example parameter group");

        var groupInfo =
            await
 auroraWrapper.DescribeCustomDBClusterParameterGroupAsync(parameterGroup.DBClusterParamet

        Console.WriteLine(
            $"3. New DB parameter group created: \n\t{groupInfo?.Description}, \n
\tARN {groupInfo?.DBClusterParameterGroupName}");
        Console.WriteLine(sepBar);
        return parameterGroup;
    }

    /// <summary>
    /// Get and describe parameters from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <param name="parameterNames">Optional specific names of parameters to
 describe.</param>
    /// <returns>The list of requested parameters.</returns>
    public static async Task<List<Parameter>>
 DescribeParametersInGroupAsync(string parameterGroupName, List<string>?
 parameterNames = null)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("4. Get some parameters from the group.");
        Console.WriteLine(sepBar);

        var parameters =
            await
 auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName);
```

```
        var matchingParameters =
            parameters.Where(p => parameterNames == null ||
 parameterNames.Contains(p.ParameterName)).ToList();

        Console.WriteLine("5. Parameter information:");
        matchingParameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);

        return matchingParameters;
    }

    /// <summary>
    /// Modify a parameter from a DBParameterGroup.
    /// </summary>
    /// <param name="parameterGroupName">Name of the DBParameterGroup.</param>
    /// <param name="parameters">The parameters to modify.</param>
    /// <returns>Async task.</returns>
    public static async Task ModifyParametersAsync(string parameterGroupName,
 List<Parameter> parameters)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("6. Modify some parameters in the group.");

        await
 auroraWrapper.ModifyIntegerParametersInGroupAsync(parameterGroupName,
 parameters);

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Describe the user source parameters in the group.
    /// </summary>
    /// <param name="parameterGroupName">The name of the DBParameterGroup.</
param>
    /// <returns>Async task.</returns>
    public static async Task DescribeUserSourceParameters(string
 parameterGroupName)
```

```
    {
        Console.WriteLine(sepBar);
        Console.WriteLine("7. Describe updated user source parameters in the
group.");

        var parameters =
            await
auroraWrapper.DescribeDBClusterParametersInGroupAsync(parameterGroupName,
"user");

        parameters.ForEach(p =>
            Console.WriteLine(
                $"\n\tParameter: {p.ParameterName}." +
                $"\n\tDescription: {p.Description}." +
                $"\n\tAllowed Values: {p.AllowedValues}." +
                $"\n\tValue: {p.ParameterValue}."));

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Choose a DB engine version.
    /// </summary>
    /// <param name="dbParameterGroupFamily">DB parameter group family for engine
choice.</param>
    /// <returns>The selected engine version.</returns>
    public static async Task<DBEngineVersion> ChooseDBEngineVersionAsync(string
dbParameterGroupFamily)
    {
        Console.WriteLine(sepBar);
        // Get a list of allowed engines.
        var allowedEngines =
            await auroraWrapper.DescribeDBEngineVersionsForEngineAsync(engine,
dbParameterGroupFamily);

        Console.WriteLine($"Available DB engine versions for parameter group
family {dbParameterGroupFamily}:");
        int i = 1;
        foreach (var version in allowedEngines)
        {
            Console.WriteLine(
                $"\t{i}. {version.DBEngineVersionDescription}.");
            i++;
        }
```

```
        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedEngines.Count)
        {
            Console.WriteLine("8. Select an available DB engine version by
 entering a number from the list above:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var engineChoice = allowedEngines[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return engineChoice;
    }

    /// <summary>
    /// Create a new RDS DB cluster.
    /// </summary>
    /// <param name="parameterGroup">Parameter group to use for the DB cluster.</
param>
    /// <param name="engineName">Engine to use for the DB cluster.</param>
    /// <param name="engineVersion">Engine version to use for the DB cluster.</
param>
    /// <param name="clusterIdentifier">Cluster identifier to use for the DB
 cluster.</param>
    /// <returns>The new DB cluster.</returns>
    public static async Task<DBCluster?> CreateNewCluster(DBClusterParameterGroup
 parameterGroup,
        string engineName, string engineVersion, string clusterIdentifier)
    {
        Console.WriteLine(sepBar);
        Console.WriteLine($"9. Create a new DB cluster with identifier
 {clusterIdentifier}.");

        DBCluster newCluster;
        var clusters = await auroraWrapper.DescribeDBClustersPagedAsync();
        var isClusterCreated = clusters.Any(i => i.DBClusterIdentifier ==
 clusterIdentifier);

        if (isClusterCreated)
        {
            Console.WriteLine("Cluster already created.");
            newCluster = clusters.First(i => i.DBClusterIdentifier ==
 clusterIdentifier);
```

```
        }
        else
        {
            Console.WriteLine("Enter an admin username:");
            var username = Console.ReadLine();

            Console.WriteLine("Enter an admin password:");
            var password = Console.ReadLine();

            newCluster = await auroraWrapper.CreateDBClusterWithAdminAsync(
                "ExampleDatabase",
                clusterIdentifier,
                parameterGroup.DBClusterParameterGroupName,
                engineName,
                engineVersion,
                username!,
                password!
            );

            Console.WriteLine("10. Waiting for DB cluster to be ready...");
            while (newCluster.Status != "available")
            {
                Console.Write(".");
                Thread.Sleep(5000);
                clusters = await
 auroraWrapper.DescribeDBClustersPagedAsync(clusterIdentifier);
                newCluster = clusters.First();
            }
        }

        Console.WriteLine(sepBar);
        return newCluster;
    }

    /// <summary>
    /// Choose a DB instance class for a particular engine and engine version.
    /// </summary>
    /// <param name="engine">DB engine for DB instance choice.</param>
    /// <param name="engineVersion">DB engine version for DB instance choice.</
param>
    /// <returns>The selected orderable DB instance option.</returns>
    public static async Task<OrderableDBInstanceOption>
 ChooseDBInstanceClass(string engine, string engineVersion)
    {
```

```
        Console.WriteLine(sepBar);
        // Get a list of allowed DB instance classes.
        var allowedInstances =
            await
 auroraWrapper.DescribeOrderableDBInstanceOptionsPagedAsync(engine,
 engineVersion);

        Console.WriteLine($"Available DB instance classes for engine {engine} and
 version {engineVersion}:");
        int i = 1;

        foreach (var instance in allowedInstances)
        {
            Console.WriteLine(
                $"\t{i}. Instance class: {instance.DBInstanceClass} (storage type
 {instance.StorageType})");
            i++;
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > allowedInstances.Count)
        {
            Console.WriteLine("11. Select an available DB instance class by
 entering a number from the preceding list:");
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        var instanceChoice = allowedInstances[choiceNumber - 1];
        Console.WriteLine(sepBar);
        return instanceChoice;
    }

    /// <summary>
    /// Create a new DB instance.
    /// </summary>
    /// <param name="engineName">Engine to use for the DB instance.</param>
    /// <param name="engineVersion">Engine version to use for the DB instance.</
param>
    /// <param name="instanceClass">Instance class to use for the DB instance.</
param>
    /// <param name="instanceIdentifier">Instance identifier to use for the DB
 instance.</param>
    /// <returns>The new DB instance.</returns>
```

```
   public static async Task<DBInstance?> CreateNewInstance(
       string clusterIdentifier,
       string engineName,
       string engineVersion,
       string instanceClass,
       string instanceIdentifier)
   {
       Console.WriteLine(sepBar);
       Console.WriteLine($"12. Create a new DB instance with identifier
{instanceIdentifier}.");
       bool isInstanceReady = false;
       DBInstance newInstance;
       var instances = await auroraWrapper.DescribeDBInstancesPagedAsync();
       isInstanceReady = instances.FirstOrDefault(i =>
           i.DBInstanceIdentifier == instanceIdentifier)?.DBInstanceStatus ==
"available";

       if (isInstanceReady)
       {
           Console.WriteLine("Instance already created.");
           newInstance = instances.First(i => i.DBInstanceIdentifier ==
instanceIdentifier);
       }
       else
       {
           newInstance = await auroraWrapper.CreateDBInstanceInClusterAsync(
               clusterIdentifier,
               instanceIdentifier,
               engineName,
               engineVersion,
               instanceClass
           );

           Console.WriteLine("13. Waiting for DB instance to be ready...");
           while (!isInstanceReady)
           {
               Console.Write(".");
               Thread.Sleep(5000);
               instances = await
auroraWrapper.DescribeDBInstancesPagedAsync(instanceIdentifier);
               isInstanceReady = instances.FirstOrDefault()?.DBInstanceStatus ==
"available";
               newInstance = instances.First();
           }
```

```
        }

        Console.WriteLine(sepBar);
        return newInstance;
    }

    /// <summary>
    /// Display a connection string for an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">The DB cluster to use to get a connection string.</
param>
    public static void DisplayConnectionString(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Display the connection string.
        Console.WriteLine("14. New DB cluster connection string: ");
        Console.WriteLine(
            $"\n{engine} -h {cluster.Endpoint} -P {cluster.Port} "
            + $"-u {cluster.MasterUsername} -p [YOUR PASSWORD]\n");

        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Create a snapshot from an Amazon RDS DB cluster.
    /// </summary>
    /// <param name="cluster">DB cluster to use when creating a snapshot.</param>
    /// <returns>The snapshot object.</returns>
    public static async Task<DBClusterSnapshot> CreateSnapshot(DBCluster cluster)
    {
        Console.WriteLine(sepBar);
        // Create a snapshot.
        Console.WriteLine($"15. Creating snapshot from DB cluster
 {cluster.DBClusterIdentifier}.");
        var snapshot = await
 auroraWrapper.CreateClusterSnapshotByIdentifierAsync(
            cluster.DBClusterIdentifier,
            "ExampleSnapshot-" + DateTime.Now.Ticks);

        // Wait for the snapshot to be available.
        bool isSnapshotReady = false;

        Console.WriteLine($"16. Waiting for snapshot to be ready...");
        while (!isSnapshotReady)
```

```
        {
            Console.Write(".");
            Thread.Sleep(5000);
            var snapshots =
                await
auroraWrapper.DescribeDBClusterSnapshotsByIdentifierAsync(cluster.DBClusterIdentifier);
            isSnapshotReady = snapshots.FirstOrDefault()?.Status == "available";
            snapshot = snapshots.First();
        }

        Console.WriteLine(
            $"Snapshot {snapshot.DBClusterSnapshotIdentifier} status is
{snapshot.Status}.");
        Console.WriteLine(sepBar);
        return snapshot;
    }

    /// <summary>
    /// Clean up resources from the scenario.
    /// </summary>
    /// <param name="newInstance">The instance to clean up.</param>
    /// <param name="newCluster">The cluster to clean up.</param>
    /// <param name="parameterGroup">The parameter group to clean up.</param>
    /// <returns>Async Task.</returns>
    private static async Task CleanupResources(
        DBInstance? newInstance,
        DBCluster? newCluster,
        DBClusterParameterGroup? parameterGroup)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        if (newInstance is not null && GetYesNoResponse($"\tClean up instance
{newInstance.DBInstanceIdentifier}? (y/n)"))
        {
            // Delete the DB instance.
            Console.WriteLine($"17. Deleting the DB instance
{newInstance.DBInstanceIdentifier}.");
            await
auroraWrapper.DeleteDBInstanceByIdentifierAsync(newInstance.DBInstanceIdentifier);
        }

        if (newCluster is not null && GetYesNoResponse($"\tClean up cluster
{newCluster.DBClusterIdentifier}? (y/n)"))
```

```
        {
            // Delete the DB cluster.
            Console.WriteLine($"18. Deleting the DB cluster
 {newCluster.DBClusterIdentifier}.");
            await
 auroraWrapper.DeleteDBClusterByIdentifierAsync(newCluster.DBClusterIdentifier);

            // Wait for the DB cluster to delete.
            Console.WriteLine($"19. Waiting for the DB cluster to delete...");
            bool isClusterDeleted = false;

            while (!isClusterDeleted)
            {
                Console.Write(".");
                Thread.Sleep(5000);
                var cluster = await auroraWrapper.DescribeDBClustersPagedAsync();
                isClusterDeleted = cluster.All(i => i.DBClusterIdentifier !=
 newCluster.DBClusterIdentifier);
            }

            Console.WriteLine("DB cluster deleted.");
        }

        if (parameterGroup is not null && GetYesNoResponse($"\tClean up parameter
 group? (y/n)"))
        {
            Console.WriteLine($"20. Deleting the DB parameter group
 {parameterGroup.DBClusterParameterGroupName}.");
            await
 auroraWrapper.DeleteClusterParameterGroupByNameAsync(parameterGroup.DBClusterParameterGr
            Console.WriteLine("Parameter group deleted.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
```

```
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null &&
                       ynResponse.Equals("y",
                           StringComparison.InvariantCultureIgnoreCase);
        return response;
    }
```

Méthodes d'encapsulation appelées par le scénario pour gérer les actions Aurora.

```
using Amazon.RDS;
using Amazon.RDS.Model;

namespace AuroraActions;

/// <summary>
/// Wrapper for the Amazon Aurora cluster client operations.
/// </summary>
public class AuroraWrapper
{
    private readonly IAmazonRDS _amazonRDS;
    public AuroraWrapper(IAmazonRDS amazonRDS)
    {
        _amazonRDS = amazonRDS;
    }

    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
 name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
 DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
```

```
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }

    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
    public async Task<DBClusterParameterGroup>
 CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
 _amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }

    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
 DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
```

```
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
 _amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }

    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }

    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
```

```
   /// <returns>The name of the group that was modified.</returns>
   public async Task<string> ModifyIntegerParametersInGroupAsync(string
groupName, List<Parameter> parameters, int newValue = 0)
   {
       foreach (var p in parameters)
       {
           if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
           {
               while (newValue == 0)
               {
                   Console.WriteLine(
                       $"Enter a new value for {p.ParameterName} from the
allowed values {p.AllowedValues} ");

                   var choice = Console.ReadLine();
                   int.TryParse(choice, out newValue);
               }

               p.ParameterValue = newValue.ToString();
           }
       }

       var request = new ModifyDBClusterParameterGroupRequest
       {
           Parameters = parameters,
           DBClusterParameterGroupName = groupName,
       };

       var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
       return result.DBClusterParameterGroupName;
   }


   /// <summary>
   /// Get a list of orderable DB instance options for a specific
   /// engine and engine version.
   /// </summary>
   /// <param name="engine">Name of the engine.</param>
   /// <param name="engineVersion">Version of the engine.</param>
   /// <returns>List of OrderableDBInstanceOptions.</returns>
   public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
```

```
    {
        // Use a paginator to get a list of DB instance options.
        var results = new List<OrderableDBInstanceOption>();
        var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
            new DescribeOrderableDBInstanceOptionsRequest()
            {
                Engine = engine,
                EngineVersion = engineVersion,
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }

    /// <summary>
    /// Delete a particular parameter group by name.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Create a new cluster and database.
    /// </summary>
    /// <param name="dbName">The name of the new database.</param>
    /// <param name="clusterIdentifier">The identifier of the cluster.</param>
    /// <param name="parameterGroupName">The name of the parameter group.</param>
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
```

```
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }

    /// <summary>
    /// Returns a list of DB instances.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
    /// <returns>List of DB instances.</returns>
    public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
    {
        var results = new List<DBInstance>();
        var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
            new DescribeDBInstancesRequest
            {
                DBInstanceIdentifier = dbInstanceIdentifier
            });
        // Get the entire list using the paginator.
```

```
        await foreach (var instances in instancesPaginator.DBInstances)
        {
            results.Add(instances);
        }
        return results;
    }

    /// <summary>
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
```

```
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }

    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });
```

```
        return response.DBClusterSnapshot;
    }

    /// <summary>
    /// Return a list of DB snapshots for a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>List of DB snapshots.</returns>
    public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
    {
        var results = new List<DBClusterSnapshot>();

        DescribeDBClusterSnapshotsResponse response;
        DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }

    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });
```

```
        return response.DBCluster;
    }

    /// <summary>
    /// Delete a particular DB instance.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
 dbInstanceIdentifier)
    {
        var response = await _amazonRDS.DeleteDBInstanceAsync(
            new DeleteDBInstanceRequest()
            {
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK pour .NET.
  - CréerDBCluster
  - CréerDBClusterParameterGroup
  - Créer un DBCluster instantané
  - CréerDBInstance
  - SuppressionDBCluster
  - SuppressionDBClusterParameterGroup
  - SuppressionDBInstance
  - DécrireDBClusterParameterGroups
  - Décrire DBCluster les paramètres
  - Décrire les DBCluster instantanés
  - DécrireDBClusters

- Décrire DBEngine les versions
- DécrireDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Routine which creates an Amazon Aurora DB cluster and demonstrates several
 operations
//! on that cluster.
/*!
 \sa gettingStartedWithDBClusters()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon
 Aurora)"
              << std::endl;
    std::cout << "get started with DB clusters demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB cluster parameter group named '" <<
              CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
```

```
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB cluster parameter group already exists.
        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
                 Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
 std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available parameter group families for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                                 engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available parameter group families for " <<
 DB_ENGINE
```

```
                  << "."
                  << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family <<
std::endl;
            }
        }

        int choice = askQuestionForIntRange("Which family do you want to use? ",
1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
    if (!parameterGroupFound) {
        // 3.  Create a DB cluster parameter group.
        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your cluster parameter
group."
```

```
              << std::endl;

   Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
   // 4.  Get the parameters in the DB cluster parameter group.
   if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME,
AUTO_INCREMENT_PREFIX,
                               NO_SOURCE,
                               autoIncrementParameters,
                               client)) {
       cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
       return false;
   }

   Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

   for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
       if (autoIncParameter.GetIsModifiable() &&
           (autoIncParameter.GetDataType() == "integer")) {
           std::cout << "The " << autoIncParameter.GetParameterName()
                     << " is described as: " <<
                     autoIncParameter.GetDescription() << "." << std::endl;
           if (autoIncParameter.ParameterValueHasBeenSet()) {
               std::cout << "The current value is "
                         << autoIncParameter.GetParameterValue()
                         << "." << std::endl;
           }
           std::vector<int> splitValues = splitToInts(
                   autoIncParameter.GetAllowedValues(), '-');
           if (splitValues.size() == 2) {
               int newValue = askQuestionForIntRange(
                       Aws::String("Enter a new value between ") +
                       autoIncParameter.GetAllowedValues() + ": ",
                       splitValues[0], splitValues[1]);
               autoIncParameter.SetParameterValue(std::to_string(newValue));
               updateParameters.push_back(autoIncParameter);

           }
           else {
               std::cerr << "Error parsing " <<
autoIncParameter.GetAllowedValues()
                         << std::endl;
           }
       }
   }
```

```
   {
       // 5.  Modify the auto increment parameters in the DB cluster parameter
group.
       Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
       request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
       request.SetParameters(updateParameters);

       Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
               client.ModifyDBClusterParameterGroup(request);

       if (outcome.IsSuccess()) {
           std::cout << "The DB cluster parameter group was successfully
modified."
                     << std::endl;
       }
       else {
           std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
       }
   }

   std::cout
           << "You can get a list of parameters you've set by specifying a
source of 'user'."
           << std::endl;

   Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
   // 6.  Display the modified parameters in the DB cluster parameter group.
   if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                               userParameters,
                               client)) {
       cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
       return false;
   }

   for (const auto &userParameter: userParameters) {
       std::cout << "  " << userParameter.GetParameterName() << ", " <<
                 userParameter.GetDescription() << ", parameter value - "
                 << userParameter.GetParameterValue() << std::endl;
   }
```

```
    printAsterisksLine();
    std::cout << "Checking for an existing DB Cluster." << std::endl;

    Aws::RDS::Model::DBCluster dbCluster;
    // 7.  Check if the DB cluster already exists.
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::String engineVersionName;
    Aws::String engineName;
    if (dbCluster.DBClusterIdentifierHasBeenSet()) {
        std::cout << "The DB cluster already exists." << std::endl;
        engineVersionName = dbCluster.GetEngineVersion();
        engineName = dbCluster.GetEngine();

    }
    else {
        std::cout << "Let's create a DB cluster." << std::endl;
        const Aws::String administratorName = askQuestion(
                "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
                "Enter a password for the administrator (at least 8 characters):
");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8.  Get a list of engine versions for the parameter group family.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily,
engineVersions,
                                 client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group family are:"
                  << std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion:
engineVersions) {
            std::cout << "   " << index << ": " <<
engineVersion.GetEngineVersion()
                      << std::endl;
```

```
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ",
1,

static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion =
engineVersions[choice -
                                                                              1];

        engineName = engineVersion.GetEngine();
        engineVersionName = engineVersion.GetEngineVersion();
        std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
                  << "' and database '" << DB_NAME << "'.\n"
                  << "The DB cluster is configured to use your custom cluster
parameter group '"
                  << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
                  << "selected engine version " <<
engineVersion.GetEngineVersion()
                  << ".\nThis typically takes several minutes." << std::endl;

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
    }
```

```
    std::cout << "Waiting for the DB cluster to become available." << std::endl;

    int counter = 0;
    // 11. Wait for the DB cluster to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for cluster to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        dbCluster = Aws::RDS::Model::DBCluster();
        if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, "", client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB cluster status is '"
                      << dbCluster.GetStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbCluster.GetStatus() != "available");

    if (dbCluster.GetStatus() == "available") {
        std::cout << "The DB cluster has been created." << std::endl;
    }

    printAsterisksLine();
    Aws::RDS::Model::DBInstance dbInstance;
    // 11.  Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
                         client);
        return false;
    }
```

```
    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;

        Aws::String dbInstanceClass;
        // 12.  Get a list of instance classes.
        if (!chooseDBInstanceClass(engineName,
                                   engineVersionName,
                                   dbInstanceClass,
                                   client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
                  << "' with selected DB instance class '" << dbInstanceClass
                  << "'.\nThis typically takes several minutes." << std::endl;

        // 13. Create a DB instance.
        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
```

```
        }
    }

    std::cout << "Waiting for the DB instance to become available." << std::endl;

    counter = 0;
    // 14. Wait for the DB instance to become available.
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 900) {
            std::cerr << "Wait for instance to become available timed out ofter "
                      << counter
                      << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        dbInstance = Aws::RDS::Model::DBInstance();
        if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                             DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus()
                      << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbInstance.GetDBInstanceStatus() != "available");

    if (dbInstance.GetDBInstanceStatus() == "available") {
        std::cout << "The DB instance has been created." << std::endl;
    }

    // 15. Display the connection string that can be used to connect a 'mysql'
shell to the database.
    displayConnection(dbCluster);

    printAsterisksLine();
```

```
   if (askYesNoQuestion(
           "Do you want to create a snapshot of your DB cluster (y/n)? ")) {
       Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
                              Aws::String(Aws::Utils::UUID::RandomUUID()));
       {
           std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
           std::cout << "This typically takes a few minutes." << std::endl;

           // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
           Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
           request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
           request.SetDBClusterSnapshotIdentifier(snapshotID);

           Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                   client.CreateDBClusterSnapshot(request);

           if (outcome.IsSuccess()) {
               std::cout << "Snapshot creation has started."
                         << std::endl;
           }
           else {
               std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                         << outcome.GetError().GetMessage()
                         << std::endl;
               cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
               return false;
           }
       }

       std::cout << "Waiting for the snapshot to become available." <<
std::endl;

       Aws::RDS::Model::DBClusterSnapshot snapshot;
       counter = 0;
       do {
           std::this_thread::sleep_for(std::chrono::seconds(1));
           ++counter;
           if (counter > 600) {
               std::cerr << "Wait for snapshot to be available timed out ofter "
                         << counter
```

```
                        << " seconds." << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            // 17. Wait for the snapshot to become available.
            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }

            if ((counter % 20) == 0) {
                std::cout << "Current snapshot status is '"
                          << snapshot.GetStatus()
                          << "' after " << counter << " seconds." << std::endl;
            }
        } while (snapshot.GetStatus() != "available");

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
```

```
            "Do you want to delete the DB cluster, DB instance, and parameter
 group (y/n)? ")) {
        result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                  DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                                  client);
    }

    return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }
```

```
    return result;

}


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBCLusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBCLusterParameters(const Aws::String
 &parameterGroupName,
                                            const Aws::String &namePrefix,
                                            const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
```

```
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
```

```
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
```

```
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
```

```
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
                client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
 &options =
                    outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
 options) {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                              instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
 "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine
 are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
```

```
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                      const Aws::String &dbClusterIdentifier,
                                      const Aws::String &dbInstanceIdentifier,
                                      const Aws::RDS::RDSClient &client) {
    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                    client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                        << "Waiting for DB instance to delete before deleting the
 parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    if (!dbClusterIdentifier.empty()) {
```

```
        {
            // 19. Delete the DB cluster.
            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }
    int counter = 0;

    while (clusterDeleting || instanceDeleting) {
        // 20. Wait for the DB cluster and instance to be deleted.
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 800) {
            std::cerr << "Wait for instance to delete timed out ofter " <<
counter
                      << " seconds." << std::endl;
            return false;
        }

        Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
        if (instanceDeleting) {
            if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
                return false;
```

```
            }
            instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
        }

        Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
        if (clusterDeleting) {
            if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
                return false;
            }

            clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
        }

        if ((counter % 20) == 0) {
            if (instanceDeleting) {
                std::cout << "Current DB instance status is '"
                          << dbInstance.GetDBInstanceStatus() << "." <<
std::endl;
            }

            if (clusterDeleting) {
                std::cout << "Current DB cluster status is '"
                          << dbCluster.GetStatus() << "." << std::endl;
            }
        }
    }

    if (!parameterGroupName.empty()) {
        // 21. Delete the DB cluster parameter group.
        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
```

```
        }
    }

    return result;
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour C++.
  - CréerDBCluster
  - CréerDBClusterParameterGroup
  - Créer un DBCluster instantané
  - CréerDBInstance
  - SuppressionDBCluster
  - SuppressionDBClusterParameterGroup
  - SuppressionDBInstance
  - DécrireDBClusterParameterGroups
  - Décrire DBCluster les paramètres
  - Décrire les DBCluster instantanés
  - DécrireDBClusters
  - Décrire DBEngine les versions
  - DécrireDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Exécutez un scénario interactif à une invite de commande.

```
import (
 "aurora/actions"
 "context"
 "fmt"
 "log"
 "slices"
 "sort"
 "strconv"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/google/uuid"
)

// GetStartedClusters is an interactive example that shows you how to use the AWS
 SDK for Go
// with Amazon Aurora to do the following:
//
// 1. Create a custom DB cluster parameter group and set parameter values.
// 2. Create an Aurora DB cluster that is configured to use the parameter group.
// 3. Create a DB instance in the DB cluster that contains a database.
// 4. Take a snapshot of the DB cluster.
// 5. Delete the DB instance, DB cluster, and parameter group.
type GetStartedClusters struct {
 sdkConfig  aws.Config
 dbClusters actions.DbClusters
 questioner demotools.IQuestioner
 helper     IScenarioHelper
 isTestRun  bool
}

// NewGetStartedClusters constructs a GetStartedClusters instance from a
 configuration.
// It uses the specified config to get an Amazon Relational Database Service
 (Amazon RDS)
// client and create wrappers for the actions used in the scenario.
```

```
func NewGetStartedClusters(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) GetStartedClusters {
 auroraClient := rds.NewFromConfig(sdkConfig)
 return GetStartedClusters{
  sdkConfig:  sdkConfig,
  dbClusters: actions.DbClusters{AuroraClient: auroraClient},
  questioner: questioner,
  helper:     helper,
 }
}

// Run runs the interactive scenario.
func (scenario GetStartedClusters) Run(ctx context.Context, dbEngine string,
 parameterGroupName string,
 clusterName string, dbName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the Amazon Aurora DB Cluster demo.")
 log.Println(strings.Repeat("-", 88))

 parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine,
 parameterGroupName)
 scenario.SetUserParameters(ctx, parameterGroupName)
 cluster := scenario.CreateCluster(ctx, clusterName, dbEngine, dbName,
 parameterGroup)
 scenario.helper.Pause(5)
 dbInstance := scenario.CreateInstance(ctx, cluster)
 scenario.DisplayConnection(cluster)
 scenario.CreateSnapshot(ctx, clusterName)
 scenario.Cleanup(ctx, dbInstance, cluster, parameterGroup)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a
 specified
```

```
// database engine and create a DB cluster parameter group that is compatible
 with a
// selected engine family.
func (scenario GetStartedClusters) CreateParameterGroup(ctx context.Context,
 dbEngine string,
 parameterGroupName string) *types.DBClusterParameterGroup {

 log.Printf("Checking for an existing DB cluster parameter group named %v.\n",
  parameterGroupName)
 parameterGroup, err := scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
 if err != nil {
  panic(err)
 }
 if parameterGroup == nil {
  log.Printf("Getting available database engine versions for %v.\n", dbEngine)
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine, "")
  if err != nil {
   panic(err)
  }

  familySet := map[string]struct{}{}
  for _, family := range engineVersions {
   familySet[*family.DBParameterGroupFamily] = struct{}{}
  }
  var families []string
  for family := range familySet {
   families = append(families, family)
  }
  sort.Strings(families)
  familyIndex := scenario.questioner.AskChoice("Which family do you want to use?
\n", families)
  log.Println("Creating a DB cluster parameter group.")
  _, err = scenario.dbClusters.CreateParameterGroup(
   ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
  if err != nil {
   panic(err)
  }
  parameterGroup, err = scenario.dbClusters.GetParameterGroup(ctx,
 parameterGroupName)
  if err != nil {
   panic(err)
  }
 }
```

```
 log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tName: %v\n", *parameterGroup.DBClusterParameterGroupName)
 log.Printf("\tARN: %v\n", *parameterGroup.DBClusterParameterGroupArn)
 log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
 log.Printf("\tDescription: %v\n", *parameterGroup.Description)
 log.Println(strings.Repeat("-", 88))
 return parameterGroup

}

// SetUserParameters shows how to get the parameters contained in a custom
 parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedClusters) SetUserParameters(ctx context.Context,
 parameterGroupName string) {
 log.Println("Let's set some parameter values in your parameter group.")
 dbParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName,
 "")
 if err != nil {
  panic(err)
 }
 var updateParams []types.Parameter
 for _, dbParam := range dbParameters {
  if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
   *dbParam.IsModifiable && *dbParam.DataType == "integer" {
   log.Printf("The %v parameter is described as:\n\t%v",
    *dbParam.ParameterName, *dbParam.Description)
   rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
   lower, _ := strconv.Atoi(rangeSplit[0])
   upper, _ := strconv.Atoi(rangeSplit[1])
   newValue := scenario.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
    demotools.InIntRange{Lower: lower, Upper: upper})
   dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
   updateParams = append(updateParams, dbParam)
  }
 }
 err = scenario.dbClusters.UpdateParameters(ctx, parameterGroupName,
 updateParams)
 if err != nil {
  panic(err)
 }
 log.Println("You can get a list of parameters you've set by specifying a source
 of 'user'.")
```

```
 userParameters, err := scenario.dbClusters.GetParameters(ctx,
 parameterGroupName, "user")
 if err != nil {
  panic(err)
 }
 log.Println("Here are the parameters you've set:")
 for _, param := range userParameters {
  log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
 }
 log.Println(strings.Repeat("-", 88))
}

// CreateCluster shows how to create an Aurora DB cluster that contains a
 database
// of a specified type. The database is also configured to use a custom DB
 cluster
// parameter group.
func (scenario GetStartedClusters) CreateCluster(ctx context.Context, clusterName
 string, dbEngine string,
 dbName string, parameterGroup *types.DBClusterParameterGroup) *types.DBCluster {

 log.Println("Checking for an existing DB cluster.")
 cluster, err := scenario.dbClusters.GetDbCluster(ctx, clusterName)
 if err != nil {
  panic(err)
 }
 if cluster == nil {
  adminUsername := scenario.questioner.Ask(
   "Enter an administrator user name for the database: ", demotools.NotEmpty{})
  adminPassword := scenario.questioner.Ask(
   "Enter a password for the administrator (at least 8 characters): ",
 demotools.NotEmpty{})
  engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine,
 *parameterGroup.DBParameterGroupFamily)
  if err != nil {
   panic(err)
  }
  var engineChoices []string
  for _, engine := range engineVersions {
   engineChoices = append(engineChoices, *engine.EngineVersion)
  }
  log.Println("The available engines for your parameter group are:")
  engineIndex := scenario.questioner.AskChoice("Which engine do you want to use?
\n", engineChoices)
```

```
  log.Printf("Creating DB cluster %v and database %v.\n", clusterName, dbName)
  log.Printf("The DB cluster is configured to use\nyour custom parameter group %v
\n",
   *parameterGroup.DBClusterParameterGroupName)
  log.Printf("and selected engine %v.\n", engineChoices[engineIndex])
  log.Println("This typically takes several minutes.")
  cluster, err = scenario.dbClusters.CreateDbCluster(
   ctx, clusterName, *parameterGroup.DBClusterParameterGroupName, dbName,
 dbEngine,
   engineChoices[engineIndex], adminUsername, adminPassword)
  if err != nil {
   panic(err)
  }
  for *cluster.Status != "available" {
   scenario.helper.Pause(30)
   cluster, err = scenario.dbClusters.GetDbCluster(ctx, clusterName)
   if err != nil {
    panic(err)
   }
   log.Println("Cluster created and available.")
  }
 }
 log.Println("Cluster data:")
 log.Printf("\tDBClusterIdentifier: %v\n", *cluster.DBClusterIdentifier)
 log.Printf("\tARN: %v\n", *cluster.DBClusterArn)
 log.Printf("\tStatus: %v\n", *cluster.Status)
 log.Printf("\tEngine: %v\n", *cluster.Engine)
 log.Printf("\tEngine version: %v\n", *cluster.EngineVersion)
 log.Printf("\tDBClusterParameterGroup: %v\n", *cluster.DBClusterParameterGroup)
 log.Printf("\tEngineMode: %v\n", *cluster.EngineMode)
 log.Println(strings.Repeat("-", 88))
 return cluster
}

// CreateInstance shows how to create a DB instance in an existing Aurora DB
 cluster.
// A new DB cluster contains no DB instances, so you must add one. The first DB
 instance
// that is added to a DB cluster defaults to a read-write DB instance.
func (scenario GetStartedClusters) CreateInstance(ctx context.Context, cluster
 *types.DBCluster) *types.DBInstance {
 log.Println("Checking for an existing database instance.")
 dbInstance, err := scenario.dbClusters.GetInstance(ctx,
 *cluster.DBClusterIdentifier)
```

```
if err != nil {
 panic(err)
}
if dbInstance == nil {
 log.Println("Let's create a database instance in your DB cluster.")
 log.Println("First, choose a DB instance type:")
 instOpts, err := scenario.dbClusters.GetOrderableInstances(
  ctx, *cluster.Engine, *cluster.EngineVersion)
 if err != nil {
  panic(err)
 }
 var instChoices []string
 for _, opt := range instOpts {
  instChoices = append(instChoices, *opt.DBInstanceClass)
 }
 slices.Sort(instChoices)
 instChoices = slices.Compact(instChoices)
 instIndex := scenario.questioner.AskChoice(
  "Which DB instance class do you want to use?\n", instChoices)
 log.Println("Creating a database instance. This typically takes several
minutes.")
 dbInstance, err = scenario.dbClusters.CreateInstanceInCluster(
  ctx, *cluster.DBClusterIdentifier, *cluster.DBClusterIdentifier,
*cluster.Engine,
  instChoices[instIndex])
 if err != nil {
  panic(err)
 }
 for *dbInstance.DBInstanceStatus != "available" {
  scenario.helper.Pause(30)
  dbInstance, err = scenario.dbClusters.GetInstance(ctx,
*cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
 }
}
log.Println("Instance data:")
log.Printf("\tDBInstanceIdentifier: %v\n", *dbInstance.DBInstanceIdentifier)
log.Printf("\tARN: %v\n", *dbInstance.DBInstanceArn)
log.Printf("\tStatus: %v\n", *dbInstance.DBInstanceStatus)
log.Printf("\tEngine: %v\n", *dbInstance.Engine)
log.Printf("\tEngine version: %v\n", *dbInstance.EngineVersion)
log.Println(strings.Repeat("-", 88))
```

```
 return dbInstance
}

// DisplayConnection displays connection information about an Aurora DB cluster
 and tips
// on how to connect to it.
func (scenario GetStartedClusters) DisplayConnection(cluster *types.DBCluster) {
 log.Println(
  "You can now connect to your database using your favorite MySql client.\n" +
   "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n"
 +
   "that is running in the same VPC as your database cluster. Pass the endpoint,
\n" +
   "port, and administrator user name to 'mysql' and enter your password\n" +
   "when prompted:")
 log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
  *cluster.Endpoint, *cluster.Port, *cluster.MasterUsername)
 log.Println("For more information, see the User Guide for Aurora:\n" +
  "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
 log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB cluster snapshot and wait until it's
 available.
func (scenario GetStartedClusters) CreateSnapshot(ctx context.Context,
 clusterName string) {
 if scenario.questioner.AskBool(
  "Do you want to create a snapshot of your DB cluster (y/n)? ", "y") {
  snapshotId := fmt.Sprintf("%v-%v", clusterName, scenario.helper.UniqueId())
  log.Printf("Creating a snapshot named %v. This typically takes a few minutes.
\n", snapshotId)
  snapshot, err := scenario.dbClusters.CreateClusterSnapshot(ctx, clusterName,
 snapshotId)
  if err != nil {
   panic(err)
  }
  for *snapshot.Status != "available" {
   scenario.helper.Pause(30)
   snapshot, err = scenario.dbClusters.GetClusterSnapshot(ctx, snapshotId)
   if err != nil {
    panic(err)
   }
  }
```

```
  log.Println("Snapshot data:")
  log.Printf("\tDBClusterSnapshotIdentifier: %v\n",
 *snapshot.DBClusterSnapshotIdentifier)
  log.Printf("\tARN: %v\n", *snapshot.DBClusterSnapshotArn)
  log.Printf("\tStatus: %v\n", *snapshot.Status)
  log.Printf("\tEngine: %v\n", *snapshot.Engine)
  log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
  log.Printf("\tDBClusterIdentifier: %v\n", *snapshot.DBClusterIdentifier)
  log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
  log.Println(strings.Repeat("-", 88))
 }
}

// Cleanup shows how to clean up a DB instance, DB cluster, and DB cluster
 parameter group.
// Before the DB cluster parameter group can be deleted, all associated DB
 instances and
// DB clusters must first be deleted.
func (scenario GetStartedClusters) Cleanup(ctx context.Context, dbInstance
 *types.DBInstance, cluster *types.DBCluster,
 parameterGroup *types.DBClusterParameterGroup) {

 if scenario.questioner.AskBool(
  "\nDo you want to delete the database instance, DB cluster, and parameter group
 (y/n)? ", "y") {
  log.Printf("Deleting database instance %v.\n",
 *dbInstance.DBInstanceIdentifier)
  err := scenario.dbClusters.DeleteInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleting database cluster %v.\n", *cluster.DBClusterIdentifier)
  err = scenario.dbClusters.DeleteDbCluster(ctx, *cluster.DBClusterIdentifier)
  if err != nil {
   panic(err)
  }
  log.Println(
   "Waiting for the DB instance and DB cluster to delete. This typically takes
 several minutes.")
  for dbInstance != nil || cluster != nil {
   scenario.helper.Pause(30)
   if dbInstance != nil {
```

```
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
 *dbInstance.DBInstanceIdentifier)
    if err != nil {
     panic(err)
    }
   }
   if cluster != nil {
    cluster, err = scenario.dbClusters.GetDbCluster(ctx,
 *cluster.DBClusterIdentifier)
    if err != nil {
     panic(err)
    }
   }
  }
  log.Printf("Deleting parameter group %v.",
 *parameterGroup.DBClusterParameterGroupName)
  err = scenario.dbClusters.DeleteParameterGroup(ctx,
 *parameterGroup.DBClusterParameterGroupName)
  if err != nil {
   panic(err)
  }
 }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
 Pause(secs int)
 UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
 return uuid.New().String()
}
```

Définissez des fonctions appelées par le scénario pour gérer des actions Aurora.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}


// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}


// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
```

```
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}



// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {
```

```
 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
```

```
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
```

```
 } else {
  return output.DBCluster, err
 }
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
  return nil
 }
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}



// GetClusterSnapshot gets a DB cluster snapshot.
```

```
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
```

```
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}



// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
```

```
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}



// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Pour plus d'informations sur l'API consultez les rubriques suivantes dans la référence de l'API AWS SDK pour Go.
  - CréerDBCluster
  - CréerDBClusterParameterGroup
  - Créer un DBCluster instantané
  - CréerDBInstance
  - SuppressionDBCluster
  - SuppressionDBClusterParameterGroup
  - SuppressionDBInstance
  - DécrireDBClusterParameterGroups
  - Décrire DBCluster les paramètres
  - Décrire les DBCluster instantanés
  - DécrireDBClusters
  - Décrire DBEngine les versions
  - DécrireDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Gets available engine families for Amazon Aurora MySQL-Compatible Edition
 * by calling the DescribeDbEngineVersions(Engine='aurora-mysql') method.
 * 2. Selects an engine family and creates a custom DB cluster parameter group
 * by invoking the describeDBClusterParameters method.
 * 3. Gets the parameter groups by invoking the describeDBClusterParameterGroups
 * method.
 * 4. Gets parameters in the group by invoking the describeDBClusterParameters
 * method.
 * 5. Modifies the auto_increment_offset parameter by invoking the
 * modifyDbClusterParameterGroupRequest method.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions by invoking the
 * describeDbEngineVersions method.
 * 8. Creates an Aurora DB cluster database cluster that contains a MySQL
 * database.
 * 9. Waits for DB instance to be ready.
 * 10. Gets a list of instance classes available for the selected engine.
 * 11. Creates a database instance in the cluster.
 * 12. Waits for DB instance to be ready.
 * 13. Creates a snapshot.
 * 14. Waits for DB snapshot to be ready.
 * 15. Deletes the DB cluster.
 * 16. Deletes the DB cluster group.
 */
public class AuroraScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
```

```
                "Usage:\n" +
                "    <dbClusterGroupName> <dbParameterGroupFamily>
<dbInstanceClusterIdentifier> <dbInstanceIdentifier> <dbName>
<dbSnapshotIdentifier><secretName>"
                +
                "Where:\n" +
                "    dbClusterGroupName - The name of the DB cluster parameter
group. \n" +
                "    dbParameterGroupFamily - The DB cluster parameter group
family name (for example, aurora-mysql5.7). \n"
                +
                "    dbInstanceClusterIdentifier - The instance cluster
identifier value.\n" +
                "    dbInstanceIdentifier - The database instance identifier.\n"
+
                "    dbName - The database name.\n" +
                "    dbSnapshotIdentifier - The snapshot identifier.\n" +
                "    secretName - The name of the AWS Secrets Manager secret that
contains the database credentials\"\n";
        ;

        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbClusterGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceClusterIdentifier = args[2];
        String dbInstanceIdentifier = args[3];
        String dbName = args[4];
        String dbSnapshotIdentifier = args[5];
        String secretName = args[6];

        // Retrieve the database credentials using AWS Secrets Manager.
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String username = user.getUsername();
        String userPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
```

```
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Aurora example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Return a list of the available DB engines");
        describeDBEngines(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a custom parameter group");
        createDBClusterParameterGroup(rdsClient, dbClusterGroupName,
dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get the parameter group");
        describeDbClusterParameterGroups(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the parameters in the group");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, 0);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Modify the auto_increment_offset parameter");
        modifyDBClusterParas(rdsClient, dbClusterGroupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Display the updated parameter value");
        describeDbClusterParameters(rdsClient, dbClusterGroupName, -1);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of allowed engine versions");
        getAllowedEngines(rdsClient, dbParameterGroupFamily);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an Aurora DB cluster database");
```

```
        String arnClusterVal = createDBCluster(rdsClient, dbClusterGroupName,
dbName, dbInstanceClusterIdentifier,
                username, userPassword);
        System.out.println("The ARN of the cluster is " + arnClusterVal);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of instance classes available for the
selected engine");
        String instanceClass = getListInstanceClasses(rdsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a database instance in the cluster.");
        String clusterDBARN = createDBInstanceCluster(rdsClient,
dbInstanceIdentifier, dbInstanceClusterIdentifier,
                instanceClass);
        System.out.println("The ARN of the database is " + clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB instance to be ready");
        waitDBInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Create a snapshot");
        createDBClusterSnapshot(rdsClient, dbInstanceClusterIdentifier,
dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbSnapshotIdentifier,
dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the DB instance");
```

```
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Delete the DB cluster");
        deleteCluster(rdsClient, dbInstanceClusterIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the DB cluster group");
        deleteDBClusterGroup(rdsClient, dbClusterGroupName, clusterDBARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);
        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
                .region(region)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();
    }

    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
```

```
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
                   .build();

           DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
           System.out.println("The status of the database is " +
response.dbInstance().dbInstanceStatus());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
```

```
            String snapshotReadyStr;
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
            String dbSnapshotIdentifier) {
        try {
            CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();
```

```
            CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
            System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void waitDBInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
    }

    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String getListInstanceClasses(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest optionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                    .builder()
                    .engine("aurora-mysql")
                    .maxRecords(20)
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(optionsRequest);
            List<OrderableDBInstanceOption> instanceOptions =
response.orderableDBInstanceOptions();
            String instanceClass = "";
            for (OrderableDBInstanceOption instanceOption : instanceOptions) {
```

```
                instanceClass = instanceOption.dbInstanceClass();
                System.out.println("The instance class is " +
instanceOption.dbInstanceClass());
                System.out.println("The engine version is " +
instanceOption.engineVersion());
            }
            return instanceClass;

        } catch (RdsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .build();

            while (!instanceReady) {
                DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
                List<DBCluster> clusterList = response.dbClusters();
                for (DBCluster cluster : clusterList) {
                    instanceReadyStr = cluster.status();
                    if (instanceReadyStr.contains("available")) {
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database cluster is available!");

        } catch (RdsException | InterruptedException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
            String dbClusterIdentifier, String userName, String password) {
        try {
            CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                    .databaseName(dbName)
                    .dbClusterIdentifier(dbClusterIdentifier)
                    .dbClusterParameterGroupName(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .masterUsername(userName)
                    .masterUserPassword(password)
                    .build();

            CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
            return response.dbCluster().dbClusterArn();

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("aurora-mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
```

```
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Modify the auto_increment_offset parameter.
    public static void modifyDBClusterParas(RdsClient rdsClient, String
dClusterGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbClusterParameterGroupRequest groupRequest =
ModifyDbClusterParameterGroupRequest.builder()
                    .dbClusterParameterGroupName(dClusterGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbClusterParameterGroupResponse response =
rdsClient.modifyDBClusterParameterGroup(groupRequest);
            System.out.println(
                    "The parameter group " +
response.dbClusterParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
```

```
        try {
            DescribeDbClusterParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                        .dbClusterParameterGroupName(dbCLusterGroupName)
                        .source("user")
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }

   public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
           String dbParameterGroupFamily) {
       try {
           CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
```

```
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
 DescribeDbEngineVersionsRequest.builder()
                    .engine("aurora-mysql")
                    .defaultOnly(true)
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
 rdsClient.describeDBEngineVersions(engineVersionsRequest);
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x.
  - CréerDBCluster
  - CréerDBClusterParameterGroup
  - Créer un DBCluster instantané
  - CréerDBInstance

- SuppressionDBCluster
- SuppressionDBClusterParameterGroup
- SuppressionDBInstance
- DécrireDBClusterParameterGroups
- Décrire DBCluster les paramètres
- Décrire les DBCluster instantanés
- DécrireDBClusters
- Décrire DBEngine les versions
- DécrireDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This example requires an AWS Secrets Manager secret that contains the database
 credentials. If you do not create a
secret, this example will not work. For more details, see:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
```

```
This Kotlin example performs the following tasks:

1. Returns a list of the available DB engines.
2. Creates a custom DB parameter group.
3. Gets the parameter groups.
4. Gets the parameters in the group.
5. Modifies the auto_increment_increment parameter.
6. Displays the updated parameter value.
7. Gets a list of allowed engine versions.
8. Creates an Aurora DB cluster database.
9. Waits for DB instance to be ready.
10. Gets a list of instance classes available for the selected engine.
11. Creates a database instance in the cluster.
12. Waits for the database instance in the cluster to be ready.
13. Creates a snapshot.
14. Waits for DB snapshot to be ready.
15. Deletes the DB instance.
16. Deletes the DB cluster.
17. Deletes the DB cluster group.
 */

var slTime: Long = 20

suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <dbClusterGroupName> <dbParameterGroupFamily>
 <dbInstanceClusterIdentifier> <dbName> <dbSnapshotIdentifier> <secretName>
        Where:
            dbClusterGroupName - The database group name.
            dbParameterGroupFamily - The database parameter group name.
            dbInstanceClusterIdentifier - The database instance identifier.
            dbName -  The database name.
            dbSnapshotIdentifier - The snapshot identifier.
            secretName - The name of the AWS Secrets Manager secret that contains
 the database credentials.
    """

    if (args.size != 7) {
        println(usage)
        exitProcess(1)
    }
```

```
    val dbClusterGroupName = args[0]
    val dbParameterGroupFamily = args[1]
    val dbInstanceClusterIdentifier = args[2]
    val dbInstanceIdentifier = args[3]
    val dbName = args[4]
    val dbSnapshotIdentifier = args[5]
    val secretName = args[6]

    val gson = Gson()
    val user = gson.fromJson(getSecretValues(secretName).toString(),
User::class.java)
    val username = user.username
    val userPassword = user.password

    println("1. Return a list of the available DB engines")
    describeAuroraDBEngines()

    println("2. Create a custom parameter group")
    createDBClusterParameterGroup(dbClusterGroupName, dbParameterGroupFamily)

    println("3. Get the parameter group")
    describeDbClusterParameterGroups(dbClusterGroupName)

    println("4. Get the parameters in the group")
    describeDbClusterParameters(dbClusterGroupName, 0)

    println("5. Modify the auto_increment_offset parameter")
    modifyDBClusterParas(dbClusterGroupName)

    println("6. Display the updated parameter value")
    describeDbClusterParameters(dbClusterGroupName, -1)

    println("7. Get a list of allowed engine versions")
    getAllowedClusterEngines(dbParameterGroupFamily)

    println("8. Create an Aurora DB cluster database")
    val arnClusterVal = createDBCluster(dbClusterGroupName, dbName,
dbInstanceClusterIdentifier, username, userPassword)
    println("The ARN of the cluster is $arnClusterVal")

    println("9. Wait for DB instance to be ready")
    waitForClusterInstanceReady(dbInstanceClusterIdentifier)
```

```
    println("10. Get a list of instance classes available for the selected
 engine")
    val instanceClass = getListInstanceClasses()

    println("11. Create a database instance in the cluster.")
    val clusterDBARN = createDBInstanceCluster(dbInstanceIdentifier,
 dbInstanceClusterIdentifier, instanceClass)
    println("The ARN of the database is $clusterDBARN")

    println("12. Wait for DB instance to be ready")
    waitDBAuroraInstanceReady(dbInstanceIdentifier)

    println("13. Create a snapshot")
    createDBClusterSnapshot(dbInstanceClusterIdentifier, dbSnapshotIdentifier)

    println("14. Wait for DB snapshot to be ready")
    waitSnapshotReady(dbSnapshotIdentifier, dbInstanceClusterIdentifier)

    println("15. Delete the DB instance")
    deleteDBInstance(dbInstanceIdentifier)

    println("16. Delete the DB cluster")
    deleteCluster(dbInstanceClusterIdentifier)

    println("17. Delete the DB cluster group")
    if (clusterDBARN != null) {
        deleteDBClusterGroup(dbClusterGroupName, clusterDBARN)
    }
    println("The Scenario has successfully completed.")
}

@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
```

```
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}

suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}

suspend fun waitSnapshotReady(
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
```

```
            }
        }
    }
    println("The Snapshot is available!")
}

suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}

suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
```

```
                }
            }
        }
    }
    println("Database instance is available! The connection endpoint is
 $endpoint")
}

suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}

suspend fun getListInstanceClasses(): String {
    val optionsRequest =
        DescribeOrderableDbInstanceOptionsRequest {
            engine = "aurora-mysql"
            maxRecords = 20
        }
    var instanceClass = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeOrderableDbInstanceOptions(optionsRequest)
        response.orderableDbInstanceOptions?.forEach { instanceOption ->
            instanceClass = instanceOption.dbInstanceClass.toString()
            println("The instance class is ${instanceOption.dbInstanceClass}")
            println("The engine version is ${instanceOption.engineVersion}")
        }
    }
```

```
    return instanceClass
}

// Waits until the database instance is available.
suspend fun waitForClusterInstanceReady(dbClusterIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")

    val instanceRequest =
        DescribeDbClustersRequest {
            dbClusterIdentifier = dbClusterIdentifierVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbClusters(instanceRequest)
            response.dbClusters?.forEach { cluster ->
                instanceReadyStr = cluster.status.toString()
                if (instanceReadyStr.contains("available")) {
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
    println("Database cluster is available!")
}

suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
```

```
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}

// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}

// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}

suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
```

```
    }
}

suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}

suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}

suspend fun describeAuroraDBEngines() {
    val engineVersionsRequest =
        DescribeDbEngineVersionsRequest {
            engine = "aurora-mysql"
            defaultOnly = true
            maxRecords = 20
        }
```

```
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(engineVersionsRequest)
        response.dbEngineVersions?.forEach { engineOb ->
            println("The name of the DB parameter group family for the database
 engine is ${engineOb.dbParameterGroupFamily}")
            println("The name of the database engine ${engineOb.engine}")
            println("The version number of the database engine
 ${engineOb.engineVersion}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour Kotlin.
  - CréerDBCluster
  - CréerDBClusterParameterGroup
  - Créer un DBCluster instantané
  - CréerDBInstance
  - SuppressionDBCluster
  - SuppressionDBClusterParameterGroup
  - SuppressionDBInstance
  - DécrireDBClusterParameterGroups
  - Décrire DBCluster les paramètres
  - Décrire les DBCluster instantanés
  - DécrireDBClusters
  - Décrire DBEngine les versions
  - DécrireDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Exécutez un scénario interactif à une invite de commande.

```
class AuroraClusterScenario:
    """Runs a scenario that shows how to get started using Aurora DB clusters."""

    def __init__(self, aurora_wrapper):
        """
        :param aurora_wrapper: An object that wraps Aurora DB cluster actions.
        """
        self.aurora_wrapper = aurora_wrapper

    def create_parameter_group(self, db_engine, parameter_group_name):
        """
        Shows how to get available engine versions for a specified database
 engine and
        create a DB cluster parameter group that is compatible with a selected
 engine family.

        :param db_engine: The database engine to use as a basis.
        :param parameter_group_name: The name given to the newly created
 parameter group.
        :return: The newly created parameter group.
        """
        print(
            f"Checking for an existing DB cluster parameter group named
 {parameter_group_name}."
        )
        parameter_group =
 self.aurora_wrapper.get_parameter_group(parameter_group_name)
        if parameter_group is None:
            print(f"Getting available database engine versions for {db_engine}.")
            engine_versions = self.aurora_wrapper.get_engine_versions(db_engine)
```

```
            families = list({ver["DBParameterGroupFamily"] for ver in
engine_versions})
            family_index = q.choose("Which family do you want to use? ",
families)
            print(f"Creating a DB cluster parameter group.")
            self.aurora_wrapper.create_parameter_group(
                parameter_group_name, families[family_index], "Example parameter
group."
            )
            parameter_group = self.aurora_wrapper.get_parameter_group(
                parameter_group_name
            )
        print(f"Parameter group
{parameter_group['DBClusterParameterGroupName']}:")
        pp(parameter_group)
        print("-" * 88)
        return parameter_group

    def set_user_parameters(self, parameter_group_name):
        """
        Shows how to get the parameters contained in a custom parameter group and
        update some of the parameter values in the group.

        :param parameter_group_name: The name of the parameter group to query and
modify.
        """
        print("Let's set some parameter values in your parameter group.")
        auto_inc_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, name_prefix="auto_increment"
        )
        update_params = []
        for auto_inc in auto_inc_parameters:
            if auto_inc["IsModifiable"] and auto_inc["DataType"] == "integer":
                print(f"The {auto_inc['ParameterName']} parameter is described
as:")
                print(f"\t{auto_inc['Description']}")
                param_range = auto_inc["AllowedValues"].split("-")
                auto_inc["ParameterValue"] = str(
                    q.ask(
                        f"Enter a value between {param_range[0]} and
{param_range[1]}: ",
                        q.is_int,
                        q.in_range(int(param_range[0]), int(param_range[1])),
                    )
```

```
                )
                update_params.append(auto_inc)
        self.aurora_wrapper.update_parameters(parameter_group_name,
update_params)
        print(
            "You can get a list of parameters you've set by specifying a source
of 'user'."
        )
        user_parameters = self.aurora_wrapper.get_parameters(
            parameter_group_name, source="user"
        )
        pp(user_parameters)
        print("-" * 88)

    def create_cluster(self, cluster_name, db_engine, db_name, parameter_group):
        """
        Shows how to create an Aurora DB cluster that contains a database of a
specified
        type. The database is also configured to use a custom DB cluster
parameter group.

        :param cluster_name: The name given to the newly created DB cluster.
        :param db_engine: The engine of the created database.
        :param db_name: The name given to the created database.
        :param parameter_group: The parameter group that is associated with the
DB cluster.
        :return: The newly created DB cluster.
        """
        print("Checking for an existing DB cluster.")
        cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
        if cluster is None:
            admin_username = q.ask(
                "Enter an administrator user name for the database: ",
q.non_empty
            )
            admin_password = q.ask(
                "Enter a password for the administrator (at least 8 characters):
",
                q.non_empty,
            )
            engine_versions = self.aurora_wrapper.get_engine_versions(
                db_engine, parameter_group["DBParameterGroupFamily"]
            )
            engine_choices = [
```

```
                ver["EngineVersionDescription"] for ver in engine_versions
            ]
            print("The available engines for your parameter group are:")
            engine_index = q.choose("Which engine do you want to use? ",
engine_choices)
            print(
                f"Creating DB cluster {cluster_name} and database {db_name}.\n"
                f"The DB cluster is configured to use\n"
                f"your custom parameter group
{parameter_group['DBClusterParameterGroupName']}\n"
                f"and selected engine {engine_choices[engine_index]}.\n"
                f"This typically takes several minutes."
            )
            cluster = self.aurora_wrapper.create_db_cluster(
                cluster_name,
                parameter_group["DBClusterParameterGroupName"],
                db_name,
                db_engine,
                engine_versions[engine_index]["EngineVersion"],
                admin_username,
                admin_password,
            )
            while cluster.get("Status") != "available":
                wait(30)
                cluster = self.aurora_wrapper.get_db_cluster(cluster_name)
            print("Cluster created and available.\n")
        print("Cluster data:")
        pp(cluster)
        print("-" * 88)
        return cluster

    def create_instance(self, cluster):
        """
        Shows how to create a DB instance in an existing Aurora DB cluster. A new
DB cluster
        contains no DB instances, so you must add one. The first DB instance that
is added
        to a DB cluster defaults to a read-write DB instance.

        :param cluster: The DB cluster where the DB instance is added.
        :return: The newly created DB instance.
        """
        print("Checking for an existing database instance.")
        cluster_name = cluster["DBClusterIdentifier"]
```

```
        db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        if db_inst is None:
            print("Let's create a database instance in your DB cluster.")
            print("First, choose a DB instance type:")
            inst_opts = self.aurora_wrapper.get_orderable_instances(
                cluster["Engine"], cluster["EngineVersion"]
            )
            inst_choices = list(
                {
                    opt["DBInstanceClass"] + ", storage type: " +
opt["StorageType"]
                    for opt in inst_opts
                }
            )
            inst_index = q.choose(
                "Which DB instance class do you want to use? ", inst_choices
            )
            print(
                f"Creating a database instance. This typically takes several
minutes."
            )
            db_inst = self.aurora_wrapper.create_instance_in_cluster(
                cluster_name,
                cluster_name,
                cluster["Engine"],
                inst_opts[inst_index]["DBInstanceClass"],
            )
            while db_inst.get("DBInstanceStatus") != "available":
                wait(30)
                db_inst = self.aurora_wrapper.get_db_instance(cluster_name)
        print("Instance data:")
        pp(db_inst)
        print("-" * 88)
        return db_inst

    @staticmethod
    def display_connection(cluster):
        """
        Displays connection information about an Aurora DB cluster and tips on
how to
        connect to it.

        :param cluster: The DB cluster to display.
        """
```

```
        print(
            "You can now connect to your database using your favorite MySql
 client.\n"
            "One way to connect is by using the 'mysql' shell on an Amazon EC2
 instance\n"
            "that is running in the same VPC as your database cluster. Pass the
 endpoint,\n"
            "port, and administrator user name to 'mysql' and enter your password
\n"
            "when prompted:\n"
        )
        print(
            f"\n\tmysql -h {cluster['Endpoint']} -P {cluster['Port']} -u
 {cluster['MasterUsername']} -p\n"
        )
        print(
            "For more information, see the User Guide for Aurora:\n"
            "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora
        )
        print("-" * 88)

    def create_snapshot(self, cluster_name):
        """
        Shows how to create a DB cluster snapshot and wait until it's available.

        :param cluster_name: The name of a DB cluster to snapshot.
        """
        if q.ask(
            "Do you want to create a snapshot of your DB cluster (y/n)? ",
 q.is_yesno
        ):
            snapshot_id = f"{cluster_name}-{uuid.uuid4()}"
            print(
                f"Creating a snapshot named {snapshot_id}. This typically takes a
 few minutes."
            )
            snapshot = self.aurora_wrapper.create_cluster_snapshot(
                snapshot_id, cluster_name
            )
            while snapshot.get("Status") != "available":
                wait(30)
                snapshot = self.aurora_wrapper.get_cluster_snapshot(snapshot_id)
            pp(snapshot)
```

```
            print("-" * 88)

    def cleanup(self, db_inst, cluster, parameter_group):
        """
        Shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
        Before the DB cluster parameter group can be deleted, all associated DB
instances and
        DB clusters must first be deleted.

        :param db_inst: The DB instance to delete.
        :param cluster: The DB cluster to delete.
        :param parameter_group: The DB cluster parameter group to delete.
        """
        cluster_name = cluster["DBClusterIdentifier"]
        parameter_group_name = parameter_group["DBClusterParameterGroupName"]
        if q.ask(
            "\nDo you want to delete the database instance, DB cluster, and
parameter "
            "group (y/n)? ",
            q.is_yesno,
        ):
            print(f"Deleting database instance
{db_inst['DBInstanceIdentifier']}.")

self.aurora_wrapper.delete_db_instance(db_inst["DBInstanceIdentifier"])
            print(f"Deleting database cluster {cluster_name}.")
            self.aurora_wrapper.delete_db_cluster(cluster_name)
            print(
                "Waiting for the DB instance and DB cluster to delete.\n"
                "This typically takes several minutes."
            )
            while db_inst is not None or cluster is not None:
                wait(30)
                if db_inst is not None:
                    db_inst = self.aurora_wrapper.get_db_instance(
                        db_inst["DBInstanceIdentifier"]
                    )
                if cluster is not None:
                    cluster = self.aurora_wrapper.get_db_cluster(
                        cluster["DBClusterIdentifier"]
                    )
            print(f"Deleting parameter group {parameter_group_name}.")
            self.aurora_wrapper.delete_parameter_group(parameter_group_name)
```

```
    def run_scenario(self, db_engine, parameter_group_name, cluster_name,
 db_name):
        print("-" * 88)
        print(
            "Welcome to the Amazon Relational Database Service (Amazon RDS) get
 started\n"
            "with Aurora DB clusters demo."
        )
        print("-" * 88)

        parameter_group = self.create_parameter_group(db_engine,
 parameter_group_name)
        self.set_user_parameters(parameter_group_name)
        cluster = self.create_cluster(cluster_name, db_engine, db_name,
 parameter_group)
        wait(5)
        db_inst = self.create_instance(cluster)
        self.display_connection(cluster)
        self.create_snapshot(cluster_name)
        self.cleanup(db_inst, cluster, parameter_group)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    try:
        scenario = AuroraClusterScenario(AuroraWrapper.from_client())
        scenario.run_scenario(
            "aurora-mysql",
            "doc-example-cluster-parameter-group",
            "doc-example-aurora",
            "docexampledb",
        )
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

Définissez des fonctions appelées par le scénario pour gérer des actions Aurora.

```
class AuroraWrapper:
```

```
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

```
    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
group.
        :param parameter_group_family: The family that is used as the basis of
the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
```

```
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
```

```
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
```

```
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
group.
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
```

```
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise
```

```
    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
DBClusterIdentifier=cluster_id
            )
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
created.
        :param db_engine: The database engine of a database to create in the DB
instance.
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
        else:
            return db_inst


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
list of
                                       engine versions to those that are
compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
instance.
        :param db_engine_version: The engine version that must be supported by
the DB instance.
```

```
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.

        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
```

```
                )
                raise
        else:
            return db_inst


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Python (Boto3) API Reference.
  - CréerDBCluster
  - CréerDBClusterParameterGroup
  - Créer un DBCluster instantané
  - CréerDBInstance

- SuppressionDBCluster
- SuppressionDBClusterParameterGroup
- SuppressionDBInstance
- DécrireDBClusterParameterGroups
- Décrire DBCluster les paramètres
- Décrire les DBCluster instantanés
- DécrireDBClusters
- Décrire DBEngine les versions
- DécrireDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

Une bibliothèque contenant les fonctions spécifiques au scénario Aurora.

```
use phf::{phf_set, Set};
use secrecy::SecretString;
use std::{collections::HashMap, fmt::Display, time::Duration};

use aws_sdk_rds::{
    error::ProvideErrorMetadata,

 operation::create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
    types::{DbCluster, DbClusterParameterGroup, DbClusterSnapshot, DbInstance,
 Parameter},
};
use sdk_examples_test_utils::waiter::Waiter;
```

```
use tracing::{info, trace, warn};

const DB_ENGINE: &str = "aurora-mysql";
const DB_CLUSTER_PARAMETER_GROUP_NAME: &str =
 "RustSDKCodeExamplesDBParameterGroup";
const DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION: &str =
    "Parameter Group created by Rust SDK Code Example";
const DB_CLUSTER_IDENTIFIER: &str = "RustSDKCodeExamplesDBCluster";
const DB_INSTANCE_IDENTIFIER: &str = "RustSDKCodeExamplesDBInstance";

static FILTER_PARAMETER_NAMES: Set<&'static str> = phf_set! {
    "auto_increment_offset",
    "auto_increment_increment",
};

#[derive(Debug, PartialEq, Eq)]
struct MetadataError {
    message: Option<String>,
    code: Option<String>,
}

impl MetadataError {
    fn from(err: &dyn ProvideErrorMetadata) -> Self {
        MetadataError {
            message: err.message().map(String::from),
            code: err.code().map(String::from),
        }
    }
}

impl Display for MetadataError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        let display = match (&self.message, &self.code) {
            (None, None) => "Unknown".to_string(),
            (None, Some(code)) => format!("({code})"),
            (Some(message), None) => message.to_string(),
            (Some(message), Some(code)) => format!("{message} ({code})"),
        };
        write!(f, "{display}")
    }
}

#[derive(Debug, PartialEq, Eq)]
pub struct ScenarioError {
```

```
    message: String,
    context: Option<MetadataError>,
}

impl ScenarioError {
    pub fn with(message: impl Into<String>) -> Self {
        ScenarioError {
            message: message.into(),
            context: None,
        }
    }

    pub fn new(message: impl Into<String>, err: &dyn ProvideErrorMetadata) ->
 Self {
        ScenarioError {
            message: message.into(),
            context: Some(MetadataError::from(err)),
        }
    }
}

impl std::error::Error for ScenarioError {}
impl Display for ScenarioError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.context {
            Some(c) => write!(f, "{}: {}", self.message, c),
            None => write!(f, "{}", self.message),
        }
    }
}

// Parse the ParameterName, Description, and AllowedValues values and display
 them.
#[derive(Debug)]
pub struct AuroraScenarioParameter {
    name: String,
    allowed_values: String,
    current_value: String,
}

impl Display for AuroraScenarioParameter {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(
            f,
```

```
            "{}: {} (allowed: {})",
            self.name, self.current_value, self.allowed_values
        )
    }
}

impl From<aws_sdk_rds::types::Parameter> for AuroraScenarioParameter {
    fn from(value: aws_sdk_rds::types::Parameter) -> Self {
        AuroraScenarioParameter {
            name: value.parameter_name.unwrap_or_default(),
            allowed_values: value.allowed_values.unwrap_or_default(),
            current_value: value.parameter_value.unwrap_or_default(),
        }
    }
}

pub struct AuroraScenario {
    rds: crate::rds::Rds,
    engine_family: Option<String>,
    engine_version: Option<String>,
    instance_class: Option<String>,
    db_cluster_parameter_group: Option<DbClusterParameterGroup>,
    db_cluster_identifier: Option<String>,
    db_instance_identifier: Option<String>,
    username: Option<String>,
    password: Option<SecretString>,
}

impl AuroraScenario {
    pub fn new(client: crate::rds::Rds) -> Self {
        AuroraScenario {
            rds: client,
            engine_family: None,
            engine_version: None,
            instance_class: None,
            db_cluster_parameter_group: None,
            db_cluster_identifier: None,
            db_instance_identifier: None,
            username: None,
            password: None,
        }
    }
```

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");

       if let Err(err) = describe_db_engine_versions {
           return Err(ScenarioError::new(
               "Failed to retrieve DB Engine Versions",
               &err,
           ));
       };

       let version_count = describe_db_engine_versions
           .as_ref()
           .map(|o| o.db_engine_versions().len())
           .unwrap_or_default();
       info!(version_count, "got list of versions");

       // Create a map of engine families to their available versions.
       let mut versions = HashMap::<String, Vec<String>>::new();
       describe_db_engine_versions
           .unwrap()
           .db_engine_versions()
           .iter()
           .filter_map(
               |v| match (&v.db_parameter_group_family, &v.engine_version) {
                   (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                   _ => None,
               },
           )
           .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

       Ok(versions)
   }

   pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
       let describe_orderable_db_instance_options_items = self
```

```
            .rds
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o|
o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Could not get available instance
classes", &err))
    }

    // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
        self.engine_family = Some(engine.to_string());
        self.engine_version = Some(version.to_string());
        let create_db_cluster_parameter_group = self
            .rds
            .create_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
                engine,
            )
            .await;

        match create_db_cluster_parameter_group {
            Ok(CreateDbClusterParameterGroupOutput {
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
```

```
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }

        Ok(())
    }

    pub fn set_instance_class(&mut self, instance_class: Option<String>) {
        self.instance_class = instance_class;
    }

    pub fn set_login(&mut self, username: Option<String>, password:
Option<SecretString>) {
        self.username = username;
        self.password = password;
    }

    pub async fn connection_string(&self) -> Result<String, ScenarioError> {
        let cluster = self.get_cluster().await?;
        let endpoint = cluster.endpoint().unwrap_or_default();
        let port = cluster.port().unwrap_or_default();
        let username = cluster.master_username().unwrap_or_default();
        Ok(format!("mysql -h {endpoint} -P {port} -u {username} -p"))
    }

    pub async fn get_cluster(&self) -> Result<DbCluster, ScenarioError> {
        let describe_db_clusters_output = self
            .rds
            .describe_db_clusters(
```

```
                self.db_cluster_identifier
                    .as_ref()
                    .expect("cluster identifier")
                    .as_str(),
            )
            .await;
        if let Err(err) = describe_db_clusters_output {
            return Err(ScenarioError::new("Failed to get cluster", &err));
        }

        let db_cluster = describe_db_clusters_output
            .unwrap()
            .db_clusters
            .and_then(|output| output.first().cloned());

        db_cluster.ok_or_else(|| ScenarioError::with("Did not find the cluster"))
    }

    // Get the parameter group. rds.DescribeDbClusterParameterGroups
    // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
    // Parse the ParameterName, Description, and AllowedValues values and display
them.
    pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
        let parameters_output = self
            .rds
            .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
            .await;

        if let Err(err) = parameters_output {
            return Err(ScenarioError::new(
                format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
                &err,
            ));
        }

        let parameters = parameters_output
            .unwrap()
            .into_iter()
            .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
```

```
            .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
            .map(AuroraScenarioParameter::from)
            .collect::<Vec<_>>();

        Ok(parameters)
    }

    // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
    pub async fn update_auto_increment(
        &self,
        offset: u8,
        increment: u8,
    ) -> Result<(), ScenarioError> {
        let modify_db_cluster_parameter_group = self
            .rds
            .modify_db_cluster_parameter_group(
                DB_CLUSTER_PARAMETER_GROUP_NAME,
                vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value(format!("{offset}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value(format!("{increment}"))
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ],
            )
            .await;

        if let Err(error) = modify_db_cluster_parameter_group {
            return Err(ScenarioError::new(
                "Failed to modify cluster parameter group",
                &error,
            ));
        }

        Ok(())
    }
```

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
Status == 'available'.
    pub async fn snapshot(&self, name: &str) -> Result<DbClusterSnapshot,
ScenarioError> {
        let id = self.db_cluster_identifier.as_deref().unwrap_or_default();
        let snapshot = self
            .rds
            .snapshot_cluster(id, format!("{id}_{name}").as_str())
            .await;
        match snapshot {
            Ok(output) => match output.db_cluster_snapshot {
                Some(snapshot) => Ok(snapshot),
                None => Err(ScenarioError::with("Missing Snapshot")),
            },
            Err(err) => Err(ScenarioError::new("Failed to create snapshot",
&err)),
        }
    }

    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
```

```
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
```

```
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
```

```
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
```

```
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }
}

#[cfg(test)]
pub mod tests;
```

Teste la bibliothèque à l'aide de simulations automatiques autour de l'encapsuleur du client RDS.

```
use crate::rds::MockRdsImpl;

use super::*;

use std::io::{Error, ErrorKind};

use assert_matches::assert_matches;
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::DeleteDbClusterOutput,
        delete_db_cluster_parameter_group::DeleteDbClusterParameterGroupOutput,
        delete_db_instance::DeleteDbInstanceOutput,
        describe_db_cluster_endpoints::DescribeDbClusterEndpointsOutput,
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
```

```
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{
        error::DbParameterGroupAlreadyExistsFault, DbClusterEndpoint,
 DbEngineVersion,
        OrderableDbInstanceOption,
    },
};
use aws_smithy_runtime_api::http::{Response, StatusCode};
use aws_smithy_types::body::SdkBody;
use mockall::predicate::eq;
use secrecy::ExposeSecret;

#[tokio::test]
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });
```

```
    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}

#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

.db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
```

```
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}

#[tokio::test]
async fn test_scenario_get_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
```

```
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert!(cluster.is_ok());
}

#[tokio::test]
async fn test_scenario_get_cluster_missing_cluster() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| Ok(DescribeDbClustersOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Did not find the cluster");
}

#[tokio::test]
async fn test_scenario_get_cluster_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
```

```
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_clusters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let cluster = scenario.get_cluster().await;

    assert_matches!(cluster, Err(ScenarioError { message, context: _ }) if
 message == "Failed to get cluster");
}

#[tokio::test]
async fn test_scenario_connection_string() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .endpoint("test_endpoint")
                        .port(3306)
                        .master_username("test_username")
                        .build(),
```

```
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let connection_string = scenario.connection_string().await;

    assert_eq!(
        connection_string,
        Ok("mysql -h test_endpoint -P 3306 -u test_username -p".into())
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
```

```
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
```

```
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);
```

```
    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
```

```
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
```

```
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
```

```
 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
```

```
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
```

```
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

 .db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

```
#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
```

```
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
```

```
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
```

```
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_snapshot() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Ok(CreateDbClusterSnapshotOutput::builder()
                .db_cluster_snapshot(
                    DbClusterSnapshot::builder()
                        .db_cluster_identifier("MockCluster")

 .db_cluster_snapshot_identifier("MockCluster_MockSnapshot")
                        .build(),
                )
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert!(create_snapshot.is_ok());
}

#[tokio::test]
async fn test_scenario_snapshot_error() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _| {
            Err(SdkError::service_error(
                CreateDBClusterSnapshotError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create snapshot error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Failed to create snapshot");
}

#[tokio::test]
async fn test_scenario_snapshot_invalid() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_snapshot_cluster()
        .with(eq("MockCluster"), eq("MockCluster_MockSnapshot"))
        .times(1)
        .return_once(|_, _|
 Ok(CreateDbClusterSnapshotOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("MockCluster".into());
    let create_snapshot = scenario.snapshot("MockSnapshot").await;
    assert_matches!(create_snapshot, Err(ScenarioError { message, context: _}) if
 message == "Missing Snapshot");
}
```

Un binaire pour exécuter le scénario de bout en bout, en utilisant Inquirer pour que l'utilisateur puisse prendre des décisions.

```
use std::fmt::Display;

use anyhow::anyhow;
use aurora_code_examples::{
    aurora_scenario::{AuroraScenario, ScenarioError},
    rds::Rds as RdsClient,
};
use aws_sdk_rds::Client;
use inquire::{validator::StringValidator, CustomUserError};
use secrecy::SecretString;
use tracing::warn;

#[derive(Default, Debug)]
struct Warnings(Vec<String>);

impl Warnings {
    fn new() -> Self {
        Warnings(Vec::with_capacity(5))
    }

    fn push(&mut self, warning: &str, error: ScenarioError) {
        let formatted = format!("{warning}: {error}");
        warn!("{formatted}");
        self.0.push(formatted);
    }

    fn is_empty(&self) -> bool {
        self.0.is_empty()
    }
}

impl Display for Warnings {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        writeln!(f, "Warnings:")?;
        for warning in &self.0 {
            writeln!(f, "{: >4}- {warning}", "")?;
        }
        Ok(())
    }
```

```
}

fn select(
    prompt: &str,
    choices: Vec<String>,
    error_message: &str,
) -> Result<String, anyhow::Error> {
    inquire::Select::new(prompt, choices)
        .prompt()
        .map_err(|error| anyhow!("{error_message}: {error}"))
}

// Prepare the Aurora Scenario. Prompt for several settings that are optional to
 the Scenario, but that the user should choose for the demo.
// This includes the engine, engine version, and instance class.
async fn prepare_scenario(rds: RdsClient) -> Result<AuroraScenario,
 anyhow::Error> {
    let mut scenario = AuroraScenario::new(rds);

    // Get available engine families for Aurora MySql.
 rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
 'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
    let available_engines = scenario.get_engines().await;
    if let Err(error) = available_engines {
        return Err(anyhow!("Failed to get available engines: {}", error));
    }
    let available_engines = available_engines.unwrap();

    // Select an engine family and create a custom DB cluster parameter group.
 rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
    let engine = select(
        "Select an Aurora engine family",
        available_engines.keys().cloned().collect::<Vec<String>>(),
        "Invalid engine selection",
    )?;

    let version = select(
        format!("Select an Aurora engine version for {engine}").as_str(),
        available_engines.get(&engine).cloned().unwrap_or_default(),
        "Invalid engine version selection",
    )?;

    let set_engine = scenario.set_engine(engine.as_str(),
 version.as_str()).await;
```

```
    if let Err(error) = set_engine {
        return Err(anyhow!("Could not set engine: {}", error));
    }

    let instance_classes = scenario.get_instance_classes().await;
    match instance_classes {
        Ok(classes) => {
            let instance_class = select(
                format!("Select an Aurora instance class for {engine}").as_str(),
                classes,
                "Invalid instance class selection",
            )?;
            scenario.set_instance_class(Some(instance_class))
        }
        Err(err) => return Err(anyhow!("Failed to get instance classes for
 engine: {err}")),
    }

    Ok(scenario)
}

// Prepare the cluster, creating a custom parameter group overriding some group
 parameters based on user input.
async fn prepare_cluster(scenario: &mut AuroraScenario, warnings: &mut Warnings)
 -> Result<(), ()> {
    show_parameters(scenario, warnings).await;

    let offset = prompt_number_or_default(warnings, "auto_increment_offset", 5);
    let increment = prompt_number_or_default(warnings,
 "auto_increment_increment", 3);

    // Modify both the auto_increment_offset and auto_increment_increment
 parameters in one call in the custom parameter group. Set their ParameterValue
 fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
    let update_auto_increment = scenario.update_auto_increment(offset,
 increment).await;

    if let Err(error) = update_auto_increment {
        warnings.push("Failed to update auto increment", error);
        return Err(());
    }

    // Get and display the updated parameters. Specify Source of 'user' to get
 just the modified parameters. rds.DescribeDbClusterParameters(Source='user')
```

```
    show_parameters(scenario, warnings).await;

    let username = inquire::Text::new("Username for the database (default
'testuser')")
        .with_default("testuser")
        .with_initial_value("testuser")
        .prompt();

    if let Err(error) = username {
        warnings.push(
            "Failed to get username, using default",
            ScenarioError::with(format!("Error from inquirer: {error}")),
        );
        return Err(());
    }
    let username = username.unwrap();

    let password = inquire::Text::new("Password for the database (minimum 8
characters)")
        .with_validator(|i: &str| {
            if i.len() >= 8 {
                Ok(inquire::validator::Validation::Valid)
            } else {
                Ok(inquire::validator::Validation::Invalid(
                    "Password must be at least 8 characters".into(),
                ))
            }
        })
        .prompt();

    let password: Option<SecretString> = match password {
        Ok(password) => Some(SecretString::from(password)),
        Err(error) => {
            warnings.push(
                "Failed to get password, using none (and not starting a DB)",
                ScenarioError::with(format!("Error from inquirer: {error}")),
            );
            return Err(());
        }
    };

    scenario.set_login(Some(username), password);

    Ok(())
```

```
}

// Start a single instance in the cluster,
async fn run_instance(scenario: &mut AuroraScenario) -> Result<(), ScenarioError>
 {
    // Create an Aurora DB cluster database cluster that contains a MySql
 database and uses the parameter group you created.
    // Create a database instance in the cluster.
    // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
 for DBInstanceStatus == 'available'.
    scenario.start_cluster_and_instance().await?;

    let connection_string = scenario.connection_string().await?;

    println!("Database ready: {connection_string}",);

    let _ = inquire::Text::new("Use the database with the connection string. When
 you're finished, press enter key to continue.").prompt();

    // Create a snapshot of the DB cluster. rds.CreateDbClusterSnapshot.
    // Wait for the snapshot to create. rds.DescribeDbClusterSnapshots until
 Status == 'available'.
    let snapshot_name = inquire::Text::new("Provide a name for the snapshot")
        .prompt()
        .unwrap_or(String::from("ScenarioRun"));
    let snapshot = scenario.snapshot(snapshot_name.as_str()).await?;
    println!(
        "Snapshot is available: {}",
        snapshot.db_cluster_snapshot_arn().unwrap_or("Missing ARN")
    );

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), anyhow::Error> {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::from_env().load().await;
    let client = Client::new(&sdk_config);
    let rds = RdsClient::new(client);
    let mut scenario = prepare_scenario(rds).await?;

    // At this point, the scenario has things in AWS and needs to get cleaned up.
    let mut warnings = Warnings::new();
```

```
    if prepare_cluster(&mut scenario, &mut warnings).await.is_ok() {
        println!("Configured database cluster, starting an instance.");
        if let Err(err) = run_instance(&mut scenario).await {
            warnings.push("Problem running instance", err);
        }
    }

    // Clean up the instance, cluster, and parameter group, waiting for the
 instance and cluster to delete before moving on.
    let clean_up = scenario.clean_up().await;
    if let Err(errors) = clean_up {
        for error in errors {
            warnings.push("Problem cleaning up scenario", error);
        }
    }

    if warnings.is_empty() {
        Ok(())
    } else {
        println!("There were problems running the scenario:");
        println!("{warnings}");
        Err(anyhow!("There were problems running the scenario"))
    }
}

#[derive(Clone)]
struct U8Validator {}
impl StringValidator for U8Validator {
    fn validate(&self, input: &str) -> Result<inquire::validator::Validation,
 CustomUserError> {
        if input.parse::<u8>().is_err() {
            Ok(inquire::validator::Validation::Invalid(
                "Can't parse input as number".into(),
            ))
        } else {
            Ok(inquire::validator::Validation::Valid)
        }
    }
}

async fn show_parameters(scenario: &AuroraScenario, warnings: &mut Warnings) {
    let parameters = scenario.cluster_parameters().await;
```

```
    match parameters {
        Ok(parameters) => {
            println!("Current parameters");
            for parameter in parameters {
                println!("\t{parameter}");
            }
        }
        Err(error) => warnings.push("Could not find cluster parameters", error),
    }
}

fn prompt_number_or_default(warnings: &mut Warnings, name: &str, default: u8) ->
 u8 {
    let input = inquire::Text::new(format!("Updated {name}:").as_str())
        .with_validator(U8Validator {})
        .prompt();

    match input {
        Ok(increment) => match increment.parse::<u8>() {
            Ok(increment) => increment,
            Err(error) => {
                warnings.push(
                    format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                    ScenarioError::with(format!("{error}")),
                );
                default
            }
        },
        Err(error) => {
            warnings.push(
                format!("Invalid updated {name} (using {default}
 instead)").as_str(),
                ScenarioError::with(format!("{error}")),
            );
            default
        }
    }
}
```

Un encapsuleur autour du service Amazon RDS qui permet d’autosimuler les tests.

```
use aws_sdk_rds::{
    error::SdkError,
    operation::{
        create_db_cluster::{CreateDBClusterError, CreateDbClusterOutput},
        create_db_cluster_parameter_group::CreateDBClusterParameterGroupError,
        create_db_cluster_parameter_group::CreateDbClusterParameterGroupOutput,
        create_db_cluster_snapshot::{CreateDBClusterSnapshotError,
 CreateDbClusterSnapshotOutput},
        create_db_instance::{CreateDBInstanceError, CreateDbInstanceOutput},
        delete_db_cluster::{DeleteDBClusterError, DeleteDbClusterOutput},
        delete_db_cluster_parameter_group::{
            DeleteDBClusterParameterGroupError,
 DeleteDbClusterParameterGroupOutput,
        },
        delete_db_instance::{DeleteDBInstanceError, DeleteDbInstanceOutput},
        describe_db_cluster_endpoints::{
            DescribeDBClusterEndpointsError, DescribeDbClusterEndpointsOutput,
        },
        describe_db_cluster_parameters::{
            DescribeDBClusterParametersError, DescribeDbClusterParametersOutput,
        },
        describe_db_clusters::{DescribeDBClustersError,
 DescribeDbClustersOutput},
        describe_db_engine_versions::{
            DescribeDBEngineVersionsError, DescribeDbEngineVersionsOutput,
        },
        describe_db_instances::{DescribeDBInstancesError,
 DescribeDbInstancesOutput},

 describe_orderable_db_instance_options::DescribeOrderableDBInstanceOptionsError,
        modify_db_cluster_parameter_group::{
            ModifyDBClusterParameterGroupError,
 ModifyDbClusterParameterGroupOutput,
        },
    },
    types::{OrderableDbInstanceOption, Parameter},
    Client as RdsClient,
};
use secrecy::{ExposeSecret, SecretString};

#[cfg(test)]
use mockall::automock;
```

```
#[cfg(test)]
pub use MockRdsImpl as Rds;
#[cfg(not(test))]
pub use RdsImpl as Rds;

pub struct RdsImpl {
    pub inner: RdsClient,
}

#[cfg_attr(test, automock)]
impl RdsImpl {
    pub fn new(inner: RdsClient) -> Self {
        RdsImpl { inner }
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
 SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
 SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
```

```
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }
```

```
    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
```

```
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

    pub async fn describe_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner
            .describe_db_instances()
            .db_instance_identifier(instance_identifier)
            .send()
            .await
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

    pub async fn describe_db_cluster_endpoints(
        &self,
        cluster_identifier: &str,
```

```
    ) -> Result<DescribeDbClusterEndpointsOutput,
SdkError<DescribeDBClusterEndpointsError>> {
        self.inner
            .describe_db_cluster_endpoints()
            .db_cluster_identifier(cluster_identifier)
            .send()
            .await
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
```

```
    }
}
```

Le fichier Cargo.toml avec les dépendances utilisées dans ce scénario.

```
[package]
name = "aurora-code-examples"
authors = [
  "David Souther <dpsouth@amazon.com>",
]
edition = "2021"
version = "0.1.0"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/
reference/manifest.html

[dependencies]
anyhow = "1.0.75"
assert_matches = "1.5.0"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime-api = { version = "1.0.1" }
aws-sdk-rds = { version = "1.3.0" }
inquire = "0.6.2"
mockall = "0.11.4"
phf = { version = "0.11.2", features = ["std", "macros"] }
sdk-examples-test-utils = { path = "../../test-utils" }
secrecy = "0.8.0"
tokio = { version = "1.20.1", features = ["full", "test-util"] }
tracing = "0.1.37"
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour Rust.
  - CréerDBCluster
  - CréerDBClusterParameterGroup
  - Créer un DBCluster instantané
  - CréerDBInstance
  - SuppressionDBCluster

- SuppressionDBClusterParameterGroup
- SuppressionDBInstance
- DécrireDBClusterParameterGroups
- Décrire DBCluster les paramètres
- Décrire les DBCluster instantanés
- DécrireDBClusters
- Décrire DBEngine les versions
- DécrireDBInstances
- DescribeOrderableDBInstanceOptions
- ModifyDBClusterParameterGroup

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Actions pour Aurora utilisant AWS SDKs

Les exemples de code suivants montrent comment effectuer des actions Aurora individuelles avec AWSSDKs. Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Ces extraits appellent l'API Aurora et sont des extraits de code de programmes plus volumineux qui doivent être exécutés en contexte. Vous pouvez voir les actions dans leur contexte dans Scénarios d'utilisation d'Aurora AWS SDKs .

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour obtenir la liste complète, consultez la Référence des API Amazon Aurora.

Exemples

- Utilisation CreateDBCluster avec un AWS SDK
- Utilisation CreateDBClusterParameterGroup avec un AWS SDK
- Utilisation CreateDBClusterSnapshot avec un AWS SDK
- Utilisation CreateDBInstance avec un AWS SDK
- Utilisation DeleteDBCluster avec un AWS SDK
- Utilisation DeleteDBClusterParameterGroup avec un AWS SDK

- Utilisation DeleteDBInstance avec un AWS SDK
- Utilisation DescribeDBClusterParameterGroups avec un AWS SDK
- Utilisation DescribeDBClusterParameters avec un AWS SDK
- Utilisation DescribeDBClusterSnapshots avec un AWS SDK
- Utilisation DescribeDBClusters avec un AWS SDK
- Utilisation DescribeDBEngineVersions avec un AWS SDK
- Utilisation DescribeDBInstances avec un AWS SDK
- Utilisation DescribeOrderableDBInstanceOptions avec un AWS SDK ou une CLI
- Utilisation ModifyDBClusterParameterGroup avec un AWS SDK

Utilisation **`CreateDBCluster`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `CreateDBCluster`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
/// <summary>
/// Create a new cluster and database.
/// </summary>
/// <param name="dbName">The name of the new database.</param>
/// <param name="clusterIdentifier">The identifier of the cluster.</param>
/// <param name="parameterGroupName">The name of the parameter group.</param>
```

```
    /// <param name="dbEngine">The engine to use for the new cluster.</param>
    /// <param name="dbEngineVersion">The version of the engine to use.</param>
    /// <param name="adminName">The admin username.</param>
    /// <param name="adminPassword">The primary admin password.</param>
    /// <returns>The cluster object.</returns>
    public async Task<DBCluster> CreateDBClusterWithAdminAsync(
        string dbName,
        string clusterIdentifier,
        string parameterGroupName,
        string dbEngine,
        string dbEngineVersion,
        string adminName,
        string adminPassword)
    {
        var request = new CreateDBClusterRequest
        {
            DatabaseName = dbName,
            DBClusterIdentifier = clusterIdentifier,
            DBClusterParameterGroupName = parameterGroupName,
            Engine = dbEngine,
            EngineVersion = dbEngineVersion,
            MasterUsername = adminName,
            MasterUserPassword = adminPassword,
        };

        var response = await _amazonRDS.CreateDBClusterAsync(request);
        return response.DBCluster;
    }
```

- Pour plus de détails sur l'API, consultez Create DBCluster in AWS SDK pour .NETAPI Reference.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
            return false;
        }
```

- Pour plus de détails sur l'API, consultez Create DBCluster in AWS SDK pour C++API Reference.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateDbCluster creates a DB cluster that is configured to use the specified
 parameter group.
// The newly created DB cluster contains a database that uses the specified
 engine and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName
 string, parameterGroupName string,
 dbName string, dbEngine string, dbEngineVersion string, adminName string,
 adminPassword string) (
 *types.DBCluster, error) {

 output, err := clusters.AuroraClient.CreateDBCluster(ctx,
 &rds.CreateDBClusterInput{
  DBClusterIdentifier:         aws.String(clusterName),
  Engine:                      aws.String(dbEngine),
  DBClusterParameterGroupName: aws.String(parameterGroupName),
  DatabaseName:                aws.String(dbName),
  EngineVersion:               aws.String(dbEngineVersion),
  MasterUserPassword:          aws.String(adminPassword),
  MasterUsername:              aws.String(adminName),
 })
 if err != nil {
  log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
  return nil, err
 } else {
  return output.DBCluster, err
```

```
 }
}
```

- Pour plus de détails sur l'API, consultez Create DBCluster in AWS SDK pour GoAPI Reference.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static String createDBCluster(RdsClient rdsClient, String
dbParameterGroupFamily, String dbName,
           String dbClusterIdentifier, String userName, String password) {
       try {
           CreateDbClusterRequest clusterRequest =
CreateDbClusterRequest.builder()
                   .databaseName(dbName)
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .dbClusterParameterGroupName(dbParameterGroupFamily)
                   .engine("aurora-mysql")
                   .masterUsername(userName)
                   .masterUserPassword(password)
                   .build();

           CreateDbClusterResponse response =
rdsClient.createDBCluster(clusterRequest);
           return response.dbCluster().dbClusterArn();

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
       return "";
```

```
    }
```

- Pour plus de détails sur l'API, consultez Create DBCluster in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun createDBCluster(
    dbParameterGroupFamilyVal: String?,
    dbName: String?,
    dbClusterIdentifierVal: String?,
    userName: String?,
    password: String?,
): String? {
    val clusterRequest =
        CreateDbClusterRequest {
            databaseName = dbName
            dbClusterIdentifier = dbClusterIdentifierVal
            dbClusterParameterGroupName = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
            masterUsername = userName
            masterUserPassword = password
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbCluster(clusterRequest)
        return response.dbCluster?.dbClusterArn
    }
}
```

- Pour plus de détails sur l'API, voir Create DBCluster in AWSSDK for Kotlin API reference.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_db_cluster(
        self,
        cluster_name,
        parameter_group_name,
        db_name,
        db_engine,
        db_engine_version,
        admin_name,
        admin_password,
    ):
        """
        Creates a DB cluster that is configured to use the specified parameter
 group.
```

```
        The newly created DB cluster contains a database that uses the specified
engine and
        engine version.

        :param cluster_name: The name of the DB cluster to create.
        :param parameter_group_name: The name of the parameter group to associate
with
                                     the DB cluster.
        :param db_name: The name of the database to create.
        :param db_engine: The database engine of the database that is created,
such as MySql.
        :param db_engine_version: The version of the database engine.
        :param admin_name: The user name of the database administrator.
        :param admin_password: The password of the database administrator.
        :return: The newly created DB cluster.
        """
        try:
            response = self.rds_client.create_db_cluster(
                DatabaseName=db_name,
                DBClusterIdentifier=cluster_name,
                DBClusterParameterGroupName=parameter_group_name,
                Engine=db_engine,
                EngineVersion=db_engine_version,
                MasterUsername=admin_name,
                MasterUserPassword=admin_password,
            )
            cluster = response["DBCluster"]
        except ClientError as err:
            logger.error(
                "Couldn't create database %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return cluster
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Create DBCluster in AWSSDK for Python (Boto3).

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
```

```
            )
            .await;
        if let Err(err) = create_db_cluster {
            return Err(ScenarioError::new(
                "Failed to create DB Cluster with cluster group",
                &err,
            ));
        }

        self.db_cluster_identifier = create_db_cluster
            .unwrap()
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
```

```
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));
```

```
            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_cluster(
        &self,
        name: &str,
        parameter_group: &str,
        engine: &str,
        version: &str,
        username: &str,
        password: SecretString,
    ) -> Result<CreateDbClusterOutput, SdkError<CreateDBClusterError>> {
        self.inner
            .create_db_cluster()
            .db_cluster_identifier(name)
            .db_cluster_parameter_group_name(parameter_group)
            .engine(engine)
```

```
            .engine_version(version)
            .master_username(username)
            .master_user_password(password.expose_secret())
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
```

```
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));
```

```
    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
```

```
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
```

```
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
```

```
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
```

```
    let _ = assertions.await;
}
```

- Pour plus de détails sur l'API, voir Create DBCluster in AWSSDK for Rust API reference.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`CreateDBClusterParameterGroup`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `CreateDBClusterParameterGroup`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /// <summary>
    /// Create a custom cluster parameter group.
    /// </summary>
    /// <param name="parameterGroupFamily">The family of the parameter group.</
param>
    /// <param name="groupName">The name for the new parameter group.</param>
    /// <param name="description">A description for the new parameter group.</
param>
    /// <returns>The new parameter group object.</returns>
```

```
    public async Task<DBClusterParameterGroup>
CreateCustomClusterParameterGroupAsync(
        string parameterGroupFamily,
        string groupName,
        string description)
    {
        var request = new CreateDBClusterParameterGroupRequest
        {
            DBParameterGroupFamily = parameterGroupFamily,
            DBClusterParameterGroupName = groupName,
            Description = description,
        };

        var response = await
_amazonRDS.CreateDBClusterParameterGroupAsync(request);
        return response.DBClusterParameterGroup;
    }
```

- Pour plus de détails sur l'API, consultez Create DBCluster ParameterGroup in AWS SDK pour .NETAPI Reference.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
```

```
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
                client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
 created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Pour plus de détails sur l'API, consultez Create DBCluster ParameterGroup in AWS SDK pour C++API Reference.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)
```

```
type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateParameterGroup creates a DB cluster parameter group that is based on the
 specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
 ctx context.Context, parameterGroupName string, parameterGroupFamily string,
 description string) (
 *types.DBClusterParameterGroup, error) {

 output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
  &rds.CreateDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   DBParameterGroupFamily:      aws.String(parameterGroupFamily),
   Description:                 aws.String(description),
  })
 if err != nil {
  log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
  return nil, err
 } else {
  return output.DBClusterParameterGroup, err
 }
}
```

- Pour plus de détails sur l'API, consultez Create DBCluster ParameterGroup in AWS SDK pour GoAPI Reference.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void createDBClusterParameterGroup(RdsClient rdsClient, String
dbClusterGroupName,
           String dbParameterGroupFamily) {
       try {
           CreateDbClusterParameterGroupRequest groupRequest =
CreateDbClusterParameterGroupRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .dbParameterGroupFamily(dbParameterGroupFamily)
                   .description("Created by using the AWS SDK for Java")
                   .build();

           CreateDbClusterParameterGroupResponse response =
rdsClient.createDBClusterParameterGroup(groupRequest);
           System.out.println("The group name is " +
response.dbClusterParameterGroup().dbClusterParameterGroupName());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Pour plus de détails sur l'API, consultez Create DBCluster ParameterGroup in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun createDBClusterParameterGroup(
    dbClusterGroupNameVal: String?,
    dbParameterGroupFamilyVal: String?,
) {
    val groupRequest =
        CreateDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dbClusterGroupNameVal
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            description = "Created by using the AWS SDK for Kotlin"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterParameterGroup(groupRequest)
        println("The group name is
 ${response.dbClusterParameterGroup?.dbClusterParameterGroupName}")
    }
}
```

- Pour plus de détails sur l'API, voir Create DBCluster ParameterGroup in AWSSDK for Kotlin API reference.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_parameter_group(
        self, parameter_group_name, parameter_group_family, description
    ):
        """
        Creates a DB cluster parameter group that is based on the specified
 parameter group
        family.

        :param parameter_group_name: The name of the newly created parameter
 group.
        :param parameter_group_family: The family that is used as the basis of
 the new
                                       parameter group.
        :param description: A description given to the parameter group.
        :return: Data about the newly created parameter group.
        """
        try:
            response = self.rds_client.create_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                DBParameterGroupFamily=parameter_group_family,
                Description=description,
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't create parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Create DBCluster ParameterGroup in AWSSDK for Python (Boto3).

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   // Select an engine family and create a custom DB cluster parameter group.
rds.CreateDbClusterParameterGroup(DBParameterGroupFamily='aurora-mysql8.0')
   pub async fn set_engine(&mut self, engine: &str, version: &str) -> Result<(),
ScenarioError> {
       self.engine_family = Some(engine.to_string());
       self.engine_version = Some(version.to_string());
       let create_db_cluster_parameter_group = self
           .rds
           .create_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_CLUSTER_PARAMETER_GROUP_DESCRIPTION,
               engine,
           )
           .await;

       match create_db_cluster_parameter_group {
           Ok(CreateDbClusterParameterGroupOutput {
```

```
                db_cluster_parameter_group: None,
                ..
            }) => {
                return Err(ScenarioError::with(
                    "CreateDBClusterParameterGroup had empty response",
                ));
            }
            Err(error) => {
                if error.code() == Some("DBParameterGroupAlreadyExists") {
                    info!("Cluster Parameter Group already exists, nothing to
 do");
                } else {
                    return Err(ScenarioError::new(
                        "Could not create Cluster Parameter Group",
                        &error,
                    ));
                }
            }
            _ => {
                info!("Created Cluster Parameter Group");
            }
        }

        Ok(())
    }

    pub async fn create_db_cluster_parameter_group(
        &self,
        name: &str,
        description: &str,
        family: &str,
    ) -> Result<CreateDbClusterParameterGroupOutput,
 SdkError<CreateDBClusterParameterGroupError>>
    {
        self.inner
            .create_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .description(description)
            .db_parameter_group_family(family)
            .send()
            .await
    }

#[tokio::test]
```

```
async fn test_scenario_set_engine() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

 .db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert_eq!(set_engine, Ok(()));
    assert_eq!(Some("aurora-mysql"), scenario.engine_family.as_deref());
    assert_eq!(Some("aurora-mysql8.0"), scenario.engine_version.as_deref());
}

#[tokio::test]
async fn test_scenario_set_engine_not_create() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .with(
            eq("RustSDKCodeExamplesDBParameterGroup"),
            eq("Parameter Group created by Rust SDK Code Example"),
            eq("aurora-mysql"),
        )
        .return_once(|_, _, _|
 Ok(CreateDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}

#[tokio::test]
async fn test_scenario_set_engine_param_group_exists() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster_parameter_group()
        .withf(|_, _, _| true)
        .return_once(|_, _, _| {
            Err(SdkError::service_error(

 CreateDBClusterParameterGroupError::DbParameterGroupAlreadyExistsFault(
                    DbParameterGroupAlreadyExistsFault::builder().build(),
                ),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);

    let set_engine = scenario.set_engine("aurora-mysql", "aurora-
mysql8.0").await;

    assert!(set_engine.is_err());
}
```

- Pour plus de détails sur l'API, voir Create DBCluster ParameterGroup in AWSSDK for Rust API reference.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`CreateDBClusterSnapshot`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `CreateDBClusterSnapshot`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /// <summary>
    /// Create a snapshot of a cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="snapshotIdentifier">Identifier for the snapshot.</param>
    /// <returns>DB snapshot object.</returns>
    public async Task<DBClusterSnapshot>
CreateClusterSnapshotByIdentifierAsync(string dbClusterIdentifier, string
snapshotIdentifier)
    {
        var response = await _amazonRDS.CreateDBClusterSnapshotAsync(
            new CreateDBClusterSnapshotRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBClusterSnapshotIdentifier = snapshotIdentifier,
            });

        return response.DBClusterSnapshot;
    }
```

- Pour plus de détails sur l'API, consultez la section Créer un DBCluster instantané dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
            request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
                    client.CreateDBClusterSnapshot(request);

            if (outcome.IsSuccess()) {
                std::cout << "Snapshot creation has started."
                          << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
                return false;
            }
```

- Pour plus de détails sur l'API, consultez la section Créer un DBCluster instantané dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context,
 clusterName string, snapshotName string) (
 *types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
 &rds.CreateDBClusterSnapshotInput{
  DBClusterIdentifier:         aws.String(clusterName),
  DBClusterSnapshotIdentifier: aws.String(snapshotName),
 })
 if err != nil {
  log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
```

```
  return nil, err
 } else {
  return output.DBClusterSnapshot, nil
 }
}
```

- Pour plus de détails sur l'API, consultez la section Créer un DBCluster instantané dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void createDBClusterSnapshot(RdsClient rdsClient, String
dbInstanceClusterIdentifier,
           String dbSnapshotIdentifier) {
       try {
           CreateDbClusterSnapshotRequest snapshotRequest =
CreateDbClusterSnapshotRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .build();

           CreateDbClusterSnapshotResponse response =
rdsClient.createDBClusterSnapshot(snapshotRequest);
           System.out.println("The Snapshot ARN is " +
response.dbClusterSnapshot().dbClusterSnapshotArn());

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Pour plus de détails sur l'API, consultez la section Créer un DBCluster instantané dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun createDBClusterSnapshot(
    dbInstanceClusterIdentifier: String?,
    dbSnapshotIdentifier: String?,
) {
    val snapshotRequest =
        CreateDbClusterSnapshotRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbClusterSnapshot(snapshotRequest)
        println("The Snapshot ARN is
 ${response.dbClusterSnapshot?.dbClusterSnapshotArn}")
    }
}
```

- Pour plus de détails sur l'API, voir Créer un DBCluster instantané dans le AWSSDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_cluster_snapshot(self, snapshot_id, cluster_id):
        """
        Creates a snapshot of a DB cluster.

        :param snapshot_id: The ID to give the created snapshot.
        :param cluster_id: The DB cluster to snapshot.
        :return: Data about the newly created snapshot.
        """
        try:
            response = self.rds_client.create_db_cluster_snapshot(
                DBClusterSnapshotIdentifier=snapshot_id,
 DBClusterIdentifier=cluster_id
            )
```

```
            snapshot = response["DBClusterSnapshot"]
        except ClientError as err:
            logger.error(
                "Couldn't create snapshot of %s. Here's why: %s: %s",
                cluster_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Create DBCluster Snapshot in AWSSDK for Python (Boto3).

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
```

```
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
           .db_cluster
           .and_then(|c| c.db_cluster_identifier);

       if self.db_cluster_identifier.is_none() {
           return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
       }

       info!(
           "Started a db cluster: {}",
           self.db_cluster_identifier
               .as_deref()
               .unwrap_or("Missing ARN")
       );
```

```
        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
```

```
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }
```

```
        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn snapshot_cluster(
        &self,
        db_cluster_identifier: &str,
        snapshot_name: &str,
    ) -> Result<CreateDbClusterSnapshotOutput,
 SdkError<CreateDBClusterSnapshotError>> {
        self.inner
            .create_db_cluster_snapshot()
            .db_cluster_identifier(db_cluster_identifier)
            .db_cluster_snapshot_identifier(snapshot_name)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
```

```
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
```

```
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
```

```
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
```

```
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
```

```
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
```

```
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Pour plus de détails sur l'API, voir Create DBCluster Snapshot in AWSSDK for Rust API reference.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`CreateDBInstance`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `CreateDBInstance`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /// <summary>
    /// Create an Amazon Relational Database Service (Amazon RDS) DB instance
    /// with a particular set of properties. Use the action
DescribeDBInstancesAsync
    /// to determine when the DB instance is ready to use.
    /// </summary>
    /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <param name="dbEngine">The engine for the DB instance.</param>
    /// <param name="dbEngineVersion">Version for the DB instance.</param>
    /// <param name="instanceClass">Class for the DB instance.</param>
    /// <returns>DB instance object.</returns>
    public async Task<DBInstance> CreateDBInstanceInClusterAsync(
        string dbClusterIdentifier,
        string dbInstanceIdentifier,
        string dbEngine,
        string dbEngineVersion,
        string instanceClass)
    {
        // When creating the instance within a cluster, do not specify the name
or size.
        var response = await _amazonRDS.CreateDBInstanceAsync(
            new CreateDBInstanceRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                DBInstanceIdentifier = dbInstanceIdentifier,
                Engine = dbEngine,
                EngineVersion = dbEngineVersion,
                DBInstanceClass = instanceClass
            });

        return response.DBInstance;
    }
```

- Pour plus de détails sur l'API, consultez Create DBInstance in AWS SDK pour .NETAPI Reference.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::CreateDBInstanceRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetEngine(engineName);
        request.SetDBInstanceClass(dbInstanceClass);

        Aws::RDS::Model::CreateDBInstanceOutcome outcome =
                client.CreateDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB instance creation has started."
                      << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBInstance. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                             client);
            return false;
        }
```

- Pour plus de détails sur l'API, consultez Create DBInstance in AWS SDK pour C++API Reference.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// CreateInstanceInCluster creates a database instance in an existing DB cluster.
 The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context,
 clusterName string, instanceName string,
 dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
 output, err := clusters.AuroraClient.CreateDBInstance(ctx,
 &rds.CreateDBInstanceInput{
  DBInstanceIdentifier: aws.String(instanceName),
  DBClusterIdentifier:  aws.String(clusterName),
  Engine:               aws.String(dbEngine),
  DBInstanceClass:      aws.String(dbInstanceClass),
 })
 if err != nil {
  log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
```

```
  return nil, err
 } else {
  return output.DBInstance, nil
 }
}
```

- Pour plus de détails sur l'API, consultez Create DBInstance in AWS SDK pour GoAPI Reference.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    public static String createDBInstanceCluster(RdsClient rdsClient,
            String dbInstanceIdentifier,
            String dbInstanceClusterIdentifier,
            String instanceClass) {
        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbClusterIdentifier(dbInstanceClusterIdentifier)
                    .engine("aurora-mysql")
                    .dbInstanceClass(instanceClass)
                    .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
            return response.dbInstance().dbInstanceArn();

        } catch (RdsException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- Pour plus de détails sur l'API, consultez Create DBInstance in AWS SDK for Java 2.xAPI Reference.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun createDBInstanceCluster(
    dbInstanceIdentifierVal: String?,
    dbInstanceClusterIdentifierVal: String?,
    instanceClassVal: String?,
): String? {
    val instanceRequest =
        CreateDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            dbClusterIdentifier = dbInstanceClusterIdentifierVal
            engine = "aurora-mysql"
            dbInstanceClass = instanceClassVal
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.createDbInstance(instanceRequest)
        print("The status is ${response.dbInstance?.dbInstanceStatus}")
        return response.dbInstance?.dbInstanceArn
    }
}
```

- Pour plus de détails sur l'API, voir Create DBInstance in AWSSDK for Kotlin API reference.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def create_instance_in_cluster(
        self, instance_id, cluster_id, db_engine, instance_class
    ):
        """
        Creates a database instance in an existing DB cluster. The first database
 that is
        created defaults to a read-write DB instance.

        :param instance_id: The ID to give the newly created DB instance.
        :param cluster_id: The ID of the DB cluster where the DB instance is
 created.
```

```
        :param db_engine: The database engine of a database to create in the DB
instance.
                          This must be compatible with the configured parameter
group
                          of the DB cluster.
        :param instance_class: The DB instance class for the newly created DB
instance.
        :return: Data about the newly created DB instance.
        """
        try:
            response = self.rds_client.create_db_instance(
                DBInstanceIdentifier=instance_id,
                DBClusterIdentifier=cluster_id,
                Engine=db_engine,
                DBInstanceClass=instance_class,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't create DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Create DBInstance in AWSSDK for Python (Boto3).

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn create_db_instance(
        &self,
        cluster_name: &str,
        instance_name: &str,
        instance_class: &str,
        engine: &str,
    ) -> Result<CreateDbInstanceOutput, SdkError<CreateDBInstanceError>> {
        self.inner
            .create_db_instance()
            .db_cluster_identifier(cluster_name)
            .db_instance_identifier(instance_name)
            .db_instance_class(instance_class)
            .engine(engine)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
```

```
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });
```

```
    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

.db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
```

```
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
```

```
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
```

```
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
```

```
                    .build(),
            )
            .build())
    });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Pour plus de détails sur l'API, voir Create DBInstance in AWSSDK for Rust API reference.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DeleteDBCluster`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `DeleteDBCluster`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /// <summary>
    /// Delete a particular DB cluster.
    /// </summary>
    /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
    /// <returns>DB cluster object.</returns>
    public async Task<DBCluster> DeleteDBClusterByIdentifierAsync(string
dbClusterIdentifier)
    {
        var response = await _amazonRDS.DeleteDBClusterAsync(
            new DeleteDBClusterRequest()
            {
                DBClusterIdentifier = dbClusterIdentifier,
                SkipFinalSnapshot = true
            });

        return response.DBCluster;
    }
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DeleteDBClusterRequest request;
            request.SetDBClusterIdentifier(dbClusterIdentifier);
            request.SetSkipFinalSnapshot(true);

            Aws::RDS::Model::DeleteDBClusterOutcome outcome =
                    client.DeleteDBCluster(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB cluster deletion has started."
                          << std::endl;
                clusterDeleting = true;
                std::cout
                        << "Waiting for DB cluster to delete before deleting the
parameter group."
                        << std::endl;
                std::cout << "This may take a while." << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBCluster. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
  DBClusterIdentifier: aws.String(clusterName),
  SkipFinalSnapshot:   aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
  return err
 } else {
```

```
  return nil
 }
}
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void deleteCluster(RdsClient rdsClient, String
dbInstanceClusterIdentifier) {
       try {
           DeleteDbClusterRequest deleteDbClusterRequest =
DeleteDbClusterRequest.builder()
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
                   .skipFinalSnapshot(true)
                   .build();

           rdsClient.deleteDBCluster(deleteDbClusterRequest);
           System.out.println(dbInstanceClusterIdentifier + " was deleted!");

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun deleteCluster(dbInstanceClusterIdentifier: String) {
    val deleteDbClusterRequest =
        DeleteDbClusterRequest {
            dbClusterIdentifier = dbInstanceClusterIdentifier
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        rdsClient.deleteDbCluster(deleteDbClusterRequest)
        println("$dbInstanceClusterIdentifier was deleted!")
    }
}
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster dans le AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""
```

```
    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_cluster(self, cluster_name):
        """
        Deletes a DB cluster.

        :param cluster_name: The name of the DB cluster to delete.
        """
        try:
            self.rds_client.delete_db_cluster(
                DBClusterIdentifier=cluster_name, SkipFinalSnapshot=True
            )
            logger.info("Deleted DB cluster %s.", cluster_name)
        except ClientError:
            logger.exception("Couldn't delete DB cluster %s.", cluster_name)
            raise
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Delete DBCluster in AWSSDK for Python (Boto3).

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
```

```
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
```

```
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
```

```
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
 DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster(
        &self,
        cluster_identifier: &str,
    ) -> Result<DeleteDbClusterOutput, SdkError<DeleteDBClusterError>> {
        self.inner
            .delete_db_cluster()
            .db_cluster_identifier(cluster_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
```

```
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));
```

```
    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
```

```
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
```

```
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
Describe Instances
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster dans le AWS SDK pour la référence de l'API Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DeleteDBClusterParameterGroup`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `DeleteDBClusterParameterGroup`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /// <summary>
    /// Delete a particular parameter group by name.
```

```
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteClusterParameterGroupByNameAsync(string
groupName)
    {
        var request = new DeleteDBClusterParameterGroupRequest
        {
            DBClusterParameterGroupName = groupName,
        };

        var response = await
_amazonRDS.DeleteDBClusterParameterGroupAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster ParameterGroup dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(parameterGroupName);

        Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
                client.DeleteDBClusterParameterGroup(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "The DB parameter group was successfully deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster ParameterGroup dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
 parameterGroupName string) error {
 _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
  &rds.DeleteDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster ParameterGroup dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void deleteDBClusterGroup(RdsClient rdsClient, String
dbClusterGroupName, String clusterDBARN)
           throws InterruptedException {
       try {
           boolean isDataDel = false;
           boolean didFind;
           String instanceARN;

           // Make sure that the database has been deleted.
```

```
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        System.out.println(clusterDBARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }

            DeleteDbClusterParameterGroupRequest clusterParameterGroupRequest =
DeleteDbClusterParameterGroupRequest
                    .builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .build();


rdsClient.deleteDBClusterParameterGroup(clusterParameterGroupRequest);
            System.out.println(dbClusterGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster ParameterGroup dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
@Throws(InterruptedException::class)
suspend fun deleteDBClusterGroup(
    dbClusterGroupName: String,
    clusterDBARN: String,
) {
    var isDataDel = false
    var didFind: Boolean
    var instanceARN: String

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        // Make sure that the database has been deleted.
        while (!isDataDel) {
            val response = rdsClient.describeDbInstances()
            val instanceList = response.dbInstances
            val listSize = instanceList?.size
            isDataDel = false
            didFind = false
            var index = 1
            if (instanceList != null) {
                for (instance in instanceList) {
                    instanceARN = instance.dbInstanceArn.toString()
                    if (instanceARN.compareTo(clusterDBARN) == 0) {
                        println("$clusterDBARN still exists")
                        didFind = true
                    }
                    if (index == listSize && !didFind) {
                        // Went through the entire list and did not find the
 database ARN.
                        isDataDel = true
                    }
                    delay(slTime * 1000)
                    index++
```

```
                }
            }
        }
        val clusterParameterGroupRequest =
            DeleteDbClusterParameterGroupRequest {
                dbClusterParameterGroupName = dbClusterGroupName
            }

        rdsClient.deleteDbClusterParameterGroup(clusterParameterGroupRequest)
        println("$dbClusterGroupName was deleted.")
    }
}
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster ParameterGroup dans le AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
```

```
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_parameter_group(self, parameter_group_name):
        """
        Deletes a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to delete.
        :return: Data about the parameter group.
        """
        try:
            response = self.rds_client.delete_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete parameter group %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Delete DBCluster ParameterGroup in AWSSDK for Python (Boto3).

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l’exemple complet et découvrez comment le configurer et l’exécuter dans le référentiel d’exemples de code AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
```

```
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
```

```
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
```

```
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_cluster_parameter_group(
        &self,
        name: &str,
    ) -> Result<DeleteDbClusterParameterGroupOutput,
 SdkError<DeleteDBClusterParameterGroupError>>
    {
        self.inner
            .delete_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
```

```
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
```

```
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
```

```
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
 Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Pour plus de détails sur l'API, voir Supprimer DBCluster ParameterGroup dans le AWS SDK pour la référence de l'API Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DeleteDBInstance`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `DeleteDBInstance`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   /// <summary>
   /// Delete a particular DB instance.
   /// </summary>
   /// <param name="dbInstanceIdentifier">DB instance identifier.</param>
   /// <returns>DB instance object.</returns>
   public async Task<DBInstance> DeleteDBInstanceByIdentifierAsync(string
dbInstanceIdentifier)
   {
       var response = await _amazonRDS.DeleteDBInstanceAsync(
           new DeleteDBInstanceRequest()
           {
```

```
                DBInstanceIdentifier = dbInstanceIdentifier,
                SkipFinalSnapshot = true,
                DeleteAutomatedBackups = true
            });

        return response.DBInstance;
    }
```

- Pour plus de détails sur l'API, voir Supprimer DBInstance dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DeleteDBInstanceRequest request;
        request.SetDBInstanceIdentifier(dbInstanceIdentifier);
        request.SetSkipFinalSnapshot(true);
        request.SetDeleteAutomatedBackups(true);

        Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                client.DeleteDBInstance(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB instance deletion has started."
                      << std::endl;
            instanceDeleting = true;
            std::cout
```

```
                        << "Waiting for DB instance to delete before deleting the
 parameter group."
                        << std::endl;
            }
            else {
                std::cerr << "Error with Aurora::DeleteDBInstance. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
```

- Pour plus de détails sur l'API, voir Supprimer DBInstance dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName
 string) error {
 _, err := clusters.AuroraClient.DeleteDBInstance(ctx,
 &rds.DeleteDBInstanceInput{
  DBInstanceIdentifier:   aws.String(instanceName),
  SkipFinalSnapshot:      aws.Bool(true),
  DeleteAutomatedBackups: aws.Bool(true),
 })
 if err != nil {
  log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
  return err
 } else {
  return nil
 }
}
```

- Pour plus de détails sur l'API, voir Supprimer DBInstance dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
       try {
           DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                   .dbInstanceIdentifier(dbInstanceIdentifier)
                   .deleteAutomatedBackups(true)
                   .skipFinalSnapshot(true)
```

```
                    .build();

            DeleteDbInstanceResponse response =
 rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.println("The status of the database is " +
 response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Pour plus de détails sur l'API, voir Supprimer DBInstance dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun deleteDBInstance(dbInstanceIdentifierVal: String) {
    val deleteDbInstanceRequest =
        DeleteDbInstanceRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
            deleteAutomatedBackups = true
            skipFinalSnapshot = true
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.deleteDbInstance(deleteDbInstanceRequest)
        print("The status of the database is
 ${response.dbInstance?.dbInstanceStatus}")
    }
}
```

- Pour plus de détails sur l'API, voir Supprimer DBInstance dans le AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def delete_db_instance(self, instance_id):
        """
        Deletes a DB instance.

        :param instance_id: The ID of the DB instance to delete.
        :return: Data about the deleted DB instance.
        """
```

```
        try:
            response = self.rds_client.delete_db_instance(
                DBInstanceIdentifier=instance_id,
                SkipFinalSnapshot=True,
                DeleteAutomatedBackups=True,
            )
            db_inst = response["DBInstance"]
        except ClientError as err:
            logger.error(
                "Couldn't delete DB instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return db_inst
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Delete DBInstance in AWSSDK for Python (Boto3).

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
```

```
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
```

```
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
```

```
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
                &error,
            ))
        }
```

```
        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn delete_db_instance(
        &self,
        instance_identifier: &str,
    ) -> Result<DeleteDbInstanceOutput, SdkError<DeleteDBInstanceError>> {
        self.inner
            .delete_db_instance()
            .db_instance_identifier(instance_identifier)
            .skip_final_snapshot(true)
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
```

```
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
```

```
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
```

```
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));
```

```
    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Pour plus de détails sur l'API, voir Supprimer DBInstance dans le AWS SDK pour la référence de l'API Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DescribeDBClusterParameterGroups`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `DescribeDBClusterParameterGroups`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /// <summary>
    /// Get the description of a DB cluster parameter group by name.
    /// </summary>
    /// <param name="name">The name of the DB parameter group to describe.</
param>
    /// <returns>The parameter group description.</returns>
    public async Task<DBClusterParameterGroup?>
 DescribeCustomDBClusterParameterGroupAsync(string name)
    {
        var response = await _amazonRDS.DescribeDBClusterParameterGroupsAsync(
            new DescribeDBClusterParameterGroupsRequest()
            {
                DBClusterParameterGroupName = name
            });
        return response.DBClusterParameterGroups.FirstOrDefault();
    }
```

- Pour plus de détails sur l'API, voir Description DBCluster ParameterGroups dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                      CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
 std::endl;
            dbParameterGroupFamily =
 outcome.GetResult().GetDBClusterParameterGroups()
[0].GetDBParameterGroupFamily();
        }

        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- Pour plus de détails sur l'API, voir Description DBCluster ParameterGroups dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
 parameterGroupName string) (
 *types.DBClusterParameterGroup, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
  ctx, &rds.DescribeDBClusterParameterGroupsInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
  })
 if err != nil {
  var notFoundError *types.DBParameterGroupNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
   err = nil
  } else {
   log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
```

```
  }
  return nil, err
 } else {
  return &output.DBClusterParameterGroups[0], err
 }
}
```

- Pour plus de détails sur l'API, voir Description DBCluster ParameterGroups dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
           DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                   .dbClusterParameterGroupName(dbClusterGroupName)
                   .maxRecords(20)
                   .build();

           List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                   .dbClusterParameterGroups();
           for (DBClusterParameterGroup group : groups) {
               System.out.println("The group name is " +
group.dbClusterParameterGroupName());
               System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
           }
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Pour plus de détails sur l'API, voir Description DBCluster ParameterGroups dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun describeDbClusterParameterGroups(dbClusterGroupName: String?) {
    val groupsRequest =
        DescribeDbClusterParameterGroupsRequest {
            dbClusterParameterGroupName = dbClusterGroupName
            maxRecords = 20
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbClusterParameterGroups(groupsRequest)
        response.dbClusterParameterGroups?.forEach { group ->
            println("The group name is ${group.dbClusterParameterGroupName}")
            println("The group ARN is ${group.dbClusterParameterGroupArn}")
        }
    }
}
```

- Pour plus de détails sur l'API, voir Description DBCluster ParameterGroups dans le AWSSDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_parameter_group(self, parameter_group_name):
        """
        Gets a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to retrieve.
        :return: The requested parameter group.
        """
        try:
            response = self.rds_client.describe_db_cluster_parameter_groups(
                DBClusterParameterGroupName=parameter_group_name
            )
            parameter_group = response["DBClusterParameterGroups"][0]
        except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "DBParameterGroupNotFound":
                logger.info("Parameter group %s does not exist.",
parameter_group_name)
            else:
                logger.error(
                    "Couldn't get parameter group %s. Here's why: %s: %s",
                    parameter_group_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return parameter_group
```

- Pour plus de détails sur l'API, voir AWSDescribe DBCluster ParameterGroups in SDK for Python (Boto3) API Reference.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DescribeDBClusterParameters`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `DescribeDBClusterParameters`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /// <summary>
    /// Describe the cluster parameters in a parameter group.
    /// </summary>
    /// <param name="groupName">The name of the parameter group.</param>
    /// <param name="source">The optional name of the source filter.</param>
    /// <returns>The collection of parameters.</returns>
    public async Task<List<Parameter>>
DescribeDBClusterParametersInGroupAsync(string groupName, string? source = null)
    {
        var paramList = new List<Parameter>();

        DescribeDBClusterParametersResponse response;
        var request = new DescribeDBClusterParametersRequest
        {
            DBClusterParameterGroupName = groupName,
            Source = source,
        };

        // Get the full list if there are multiple pages.
        do
        {
            response = await
_amazonRDS.DescribeDBClusterParametersAsync(request);
            paramList.AddRange(response.Parameters);

            request.Marker = response.Marker;
        }
        while (response.Marker is not null);

        return paramList;
    }
```

- Pour plus de détails sur l'API, consultez la section Décrire DBCluster les paramètres dans le AWS SDK pour .NETmanuel de référence des API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBClusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBClusterParameters(const Aws::String
 &parameterGroupName,
                                            const Aws::String &namePrefix,
                                            const Aws::String &source,

 Aws::Vector<Aws::RDS::Model::Parameter> &parametersResult,
                                            const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
```

```
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
                client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                    outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}
```

- Pour plus de détails sur l'API, consultez la section Décrire DBCluster les paramètres dans le AWS SDK pour C++manuel de référence des API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetParameters gets the parameters that are contained in a DB cluster parameter
 group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
 string, source string) (
 []types.Parameter, error) {

 var output *rds.DescribeDBClusterParametersOutput
 var params []types.Parameter
 var err error
 parameterPaginator :=
 rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
  &rds.DescribeDBClusterParametersInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Source:                      aws.String(source),
  })
```

```
 for parameterPaginator.HasMorePages() {
  output, err = parameterPaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Pour plus de détails sur l'API, consultez la section Décrire DBCluster les paramètres dans le AWS SDK pour Gomanuel de référence des API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
```

```
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Pour plus de détails sur l'API, consultez la section Décrire DBCluster les paramètres dans le AWS SDK for Java 2.xmanuel de référence des API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Pour plus de détails sur l'API, voir Description DBCluster des paramètres dans le AWSSDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_parameters(self, parameter_group_name, name_prefix="", source=None):
        """
        Gets the parameters that are contained in a DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to query.
        :param name_prefix: When specified, the retrieved list of parameters is
filtered
                            to contain only parameters that start with this
prefix.
        :param source: When specified, only parameters from this source are
retrieved.
                       For example, a source of 'user' retrieves only parameters
that
                       were set by a user.
        :return: The list of requested parameters.
        """
        try:
            kwargs = {"DBClusterParameterGroupName": parameter_group_name}
            if source is not None:
                kwargs["Source"] = source
            parameters = []
            paginator =
self.rds_client.get_paginator("describe_db_cluster_parameters")
            for page in paginator.paginate(**kwargs):
                parameters += [
                    p
                    for p in page["Parameters"]
                    if p["ParameterName"].startswith(name_prefix)
                ]
        except ClientError as err:
            logger.error(
                "Couldn't get parameters for %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return parameters
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Describe DBCluster Parameters in AWSSDK for Python (Boto3).

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   // Get the parameter group. rds.DescribeDbClusterParameterGroups
   // Get parameters in the group. This is a long list so you will have to
paginate. Find the auto_increment_offset and auto_increment_increment parameters
(by ParameterName). rds.DescribeDbClusterParameters
   // Parse the ParameterName, Description, and AllowedValues values and display
them.
   pub async fn cluster_parameters(&self) ->
Result<Vec<AuroraScenarioParameter>, ScenarioError> {
       let parameters_output = self
           .rds
           .describe_db_cluster_parameters(DB_CLUSTER_PARAMETER_GROUP_NAME)
           .await;

       if let Err(err) = parameters_output {
           return Err(ScenarioError::new(
               format!("Failed to retrieve parameters for
{DB_CLUSTER_PARAMETER_GROUP_NAME}"),
               &err,
           ));
       }

       let parameters = parameters_output
           .unwrap()
           .into_iter()
           .flat_map(|p| p.parameters.unwrap_or_default().into_iter())
           .filter(|p|
FILTER_PARAMETER_NAMES.contains(p.parameter_name().unwrap_or_default()))
           .map(AuroraScenarioParameter::from)
```

```
            .collect::<Vec<_>>();

        Ok(parameters)
    }

    pub async fn describe_db_cluster_parameters(
        &self,
        name: &str,
    ) -> Result<Vec<DescribeDbClusterParametersOutput>,
 SdkError<DescribeDBClusterParametersError>>
    {
        self.inner
            .describe_db_cluster_parameters()
            .db_cluster_parameter_group_name(name)
            .into_paginator()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_cluster_parameters() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Ok(vec![DescribeDbClusterParametersOutput::builder()
                .parameters(Parameter::builder().parameter_name("a").build())
                .parameters(Parameter::builder().parameter_name("b").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("c").build())
                .parameters(
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .build(),
                )
                .parameters(Parameter::builder().parameter_name("d").build())
                .build()])
```

```
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());

    let params = scenario.cluster_parameters().await.expect("cluster params");
    let names: Vec<String> = params.into_iter().map(|p| p.name).collect();
    assert_eq!(
        names,
        vec!["auto_increment_offset", "auto_increment_increment"]
    );
}

#[tokio::test]
async fn test_scenario_cluster_parameters_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_cluster_parameters()
        .with(eq("RustSDKCodeExamplesDBParameterGroup"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBClusterParametersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_cluster_parameters_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some("RustSDKCodeExamplesDBCluster".into());
    let params = scenario.cluster_parameters().await;
    assert_matches!(params, Err(ScenarioError { message, context: _ }) if message
 == "Failed to retrieve parameters for RustSDKCodeExamplesDBParameterGroup");
}
```

- Pour plus de détails sur l'API, voir Description DBCluster des paramètres dans le AWSSDK pour la référence de l'API Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DescribeDBClusterSnapshots`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `DescribeDBClusterSnapshots`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   /// <summary>
   /// Return a list of DB snapshots for a particular DB cluster.
   /// </summary>
   /// <param name="dbClusterIdentifier">DB cluster identifier.</param>
   /// <returns>List of DB snapshots.</returns>
   public async Task<List<DBClusterSnapshot>>
DescribeDBClusterSnapshotsByIdentifierAsync(string dbClusterIdentifier)
   {
       var results = new List<DBClusterSnapshot>();

       DescribeDBClusterSnapshotsResponse response;
       DescribeDBClusterSnapshotsRequest request = new
DescribeDBClusterSnapshotsRequest
       {
           DBClusterIdentifier = dbClusterIdentifier
       };
```

```
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClusterSnapshotsAsync(request);
            results.AddRange(response.DBClusterSnapshots);
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Pour plus de détails sur l'API, consultez la section Décrire les DBCluster instantanés dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

            Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
            request.SetDBClusterSnapshotIdentifier(snapshotID);

            Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
                    client.DescribeDBClusterSnapshots(request);

            if (outcome.IsSuccess()) {
                snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
            }
            else {
```

```
                std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                 DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
 client);
                return false;
            }
```

- Pour plus de détails sur l'API, consultez la section Décrire les DBCluster instantanés dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetClusterSnapshot gets a DB cluster snapshot.
```

```
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
 string) (*types.DBClusterSnapshot, error) {
 output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
  &rds.DescribeDBClusterSnapshotsInput{
   DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
 if err != nil {
  log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
  return nil, err
 } else {
  return &output.DBClusterSnapshots[0], nil
 }
}
```

- Pour plus de détails sur l'API, consultez la section Décrire les DBCluster instantanés dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void waitForSnapshotReady(RdsClient rdsClient, String
dbSnapshotIdentifier,
           String dbInstanceClusterIdentifier) {
       try {
           boolean snapshotReady = false;
           String snapshotReadyStr;
           System.out.println("Waiting for the snapshot to become available.");

           DescribeDbClusterSnapshotsRequest snapshotsRequest =
DescribeDbClusterSnapshotsRequest.builder()
                   .dbClusterSnapshotIdentifier(dbSnapshotIdentifier)
                   .dbClusterIdentifier(dbInstanceClusterIdentifier)
```

```
                    .build();

            while (!snapshotReady) {
                DescribeDbClusterSnapshotsResponse response =
rdsClient.describeDBClusterSnapshots(snapshotsRequest);
                List<DBClusterSnapshot> snapshotList =
response.dbClusterSnapshots();
                for (DBClusterSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.println(".");
                        Thread.sleep(sleepTime * 5000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");

        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Pour plus de détails sur l'API, consultez la section Décrire les DBCluster instantanés dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun waitSnapshotReady(
```

```
    dbSnapshotIdentifier: String?,
    dbInstanceClusterIdentifier: String?,
) {
    var snapshotReady = false
    var snapshotReadyStr: String
    println("Waiting for the snapshot to become available.")

    val snapshotsRequest =
        DescribeDbClusterSnapshotsRequest {
            dbClusterSnapshotIdentifier = dbSnapshotIdentifier
            dbClusterIdentifier = dbInstanceClusterIdentifier
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!snapshotReady) {
            val response = rdsClient.describeDbClusterSnapshots(snapshotsRequest)
            val snapshotList = response.dbClusterSnapshots
            if (snapshotList != null) {
                for (snapshot in snapshotList) {
                    snapshotReadyStr = snapshot.status.toString()
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true
                    } else {
                        println(".")
                        delay(slTime * 5000)
                    }
                }
            }
        }
    }
    println("The Snapshot is available!")
}
```

- Pour plus de détails sur l'API, consultez la section Description DBCluster des instantanés dans le AWSSDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_cluster_snapshot(self, snapshot_id):
        """
        Gets a DB cluster snapshot.

        :param snapshot_id: The ID of the snapshot to retrieve.
        :return: The retrieved snapshot.
        """
        try:
            response = self.rds_client.describe_db_cluster_snapshots(
                DBClusterSnapshotIdentifier=snapshot_id
            )
            snapshot = response["DBClusterSnapshots"][0]
        except ClientError as err:
```

```
            logger.error(
                "Couldn't get DB cluster snapshot %s. Here's why: %s: %s",
                snapshot_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return snapshot
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Describe DBCluster Snapshots in AWSSDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DescribeDBClusters`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `DescribeDBClusters`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /// <summary>
```

```
    /// Returns a list of DB clusters.
    /// </summary>
    /// <param name="dbInstanceIdentifier">Optional name of a specific DB
cluster.</param>
    /// <returns>List of DB clusters.</returns>
    public async Task<List<DBCluster>> DescribeDBClustersPagedAsync(string?
dbClusterIdentifier = null)
    {
        var results = new List<DBCluster>();

        DescribeDBClustersResponse response;
        DescribeDBClustersRequest request = new DescribeDBClustersRequest
        {
            DBClusterIdentifier = dbClusterIdentifier
        };
        // Get the full list if there are multiple pages.
        do
        {
            response = await _amazonRDS.DescribeDBClustersAsync(request);
            if (response.DBClusters != null)
            {
                results.AddRange(response.DBClusters);
            }
            request.Marker = response.Marker;
        }
        while (response.Marker is not null);
        return results;
    }
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
            client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB cluster does not exist.
        // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;

}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
 (*types.DBCluster, error) {
 output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
  &rds.DescribeDBClustersInput{
   DBClusterIdentifier: aws.String(clusterName),
  })
 if err != nil {
  var notFoundError *types.DBClusterNotFoundFault
```

```
  if errors.As(err, &notFoundError) {
   log.Printf("DB cluster %v does not exist.\n", clusterName)
   err = nil
  } else {
   log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
  }
  return nil, err
 } else {
  return &output.DBClusters[0], err
 }
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void describeDbClusterParameters(RdsClient rdsClient, String
dbCLusterGroupName, int flag) {
       try {
           DescribeDbClusterParametersRequest dbParameterGroupsRequest;
           if (flag == 0) {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .build();
           } else {
               dbParameterGroupsRequest =
DescribeDbClusterParametersRequest.builder()
                       .dbClusterParameterGroupName(dbCLusterGroupName)
                       .source("user")
```

```
                        .build();
            }

            DescribeDbClusterParametersResponse response = rdsClient
                    .describeDBClusterParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset
or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") ==
0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun describeDbClusterParameters(
    dbCLusterGroupName: String?,
    flag: Int,
) {
    val dbParameterGroupsRequest: DescribeDbClusterParametersRequest
    dbParameterGroupsRequest =
        if (flag == 0) {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
            }
        } else {
            DescribeDbClusterParametersRequest {
                dbClusterParameterGroupName = dbCLusterGroupName
                source = "user"
            }
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response =
 rdsClient.describeDbClusterParameters(dbParameterGroupsRequest)
        response.parameters?.forEach { para ->
            // Only print out information about either auto_increment_offset or
 auto_increment_increment.
            val paraName = para.parameterName
            if (paraName != null) {
                if (paraName.compareTo("auto_increment_offset") == 0 ||
 paraName.compareTo("auto_increment_increment ") == 0) {
                    println("*** The parameter name is  $paraName")
                    println("*** The parameter value is  ${para.parameterValue}")
                    println("*** The parameter data type is ${para.dataType}")
                    println("*** The parameter description is
 ${para.description}")
```

```
                    println("*** The parameter allowed values  is
 ${para.allowedValues}")
                }
            }
        }
    }
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le AWSSDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)
```

```
    def get_db_cluster(self, cluster_name):
        """
        Gets data about an Aurora DB cluster.

        :param cluster_name: The name of the DB cluster to retrieve.
        :return: The retrieved DB cluster.
        """
        try:
            response = self.rds_client.describe_db_clusters(
                DBClusterIdentifier=cluster_name
            )
            cluster = response["DBClusters"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBClusterNotFoundFault":
                logger.info("Cluster %s does not exist.", cluster_name)
            else:
                logger.error(
                    "Couldn't verify the existence of DB cluster %s. Here's why:
%s: %s",
                    cluster_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return cluster
```

- Pour plus de détails sur l'API, voir AWSDescribe DBClusters in SDK for Python (Boto3) API Reference.

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   // Get a list of allowed engine versions.
rds.DescribeDbEngineVersions(Engine='aurora-mysql', DBParameterGroupFamily=<the
family used to create your parameter group in step 2>)
   // Create an Aurora DB cluster database cluster that contains a MySql
database and uses the parameter group you created.
   // Wait for DB cluster to be ready. Call rds.DescribeDBClusters and check for
Status == 'available'.
   // Get a list of instance classes available for the selected engine
and engine version. rds.DescribeOrderableDbInstanceOptions(Engine='mysql',
EngineVersion=).

   // Create a database instance in the cluster.
   // Wait for DB instance to be ready. Call rds.DescribeDbInstances and check
for DBInstanceStatus == 'available'.
   pub async fn start_cluster_and_instance(&mut self) -> Result<(),
ScenarioError> {
       if self.password.is_none() {
           return Err(ScenarioError::with(
               "Must set Secret Password before starting a cluster",
           ));
       }
       let create_db_cluster = self
           .rds
           .create_db_cluster(
               DB_CLUSTER_IDENTIFIER,
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               DB_ENGINE,
               self.engine_version.as_deref().expect("engine version"),
               self.username.as_deref().expect("username"),
               self.password
                   .replace(SecretString::new("".to_string()))
                   .expect("password"),
           )
           .await;
       if let Err(err) = create_db_cluster {
           return Err(ScenarioError::new(
               "Failed to create DB Cluster with cluster group",
               &err,
           ));
       }

       self.db_cluster_identifier = create_db_cluster
           .unwrap()
```

```
            .db_cluster
            .and_then(|c| c.db_cluster_identifier);

        if self.db_cluster_identifier.is_none() {
            return Err(ScenarioError::with("Created DB Cluster missing
Identifier"));
        }

        info!(
            "Started a db cluster: {}",
            self.db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing ARN")
        );

        let create_db_instance = self
            .rds
            .create_db_instance(
                self.db_cluster_identifier.as_deref().expect("cluster name"),
                DB_INSTANCE_IDENTIFIER,
                self.instance_class.as_deref().expect("instance class"),
                DB_ENGINE,
            )
            .await;
        if let Err(err) = create_db_instance {
            return Err(ScenarioError::new(
                "Failed to create Instance in DB Cluster",
                &err,
            ));
        }

        self.db_instance_identifier = create_db_instance
            .unwrap()
            .db_instance
            .and_then(|i| i.db_instance_identifier);

        // Cluster creation can take up to 20 minutes to become available
        let cluster_max_wait = Duration::from_secs(20 * 60);
        let waiter = Waiter::builder().max(cluster_max_wait).build();
        while waiter.sleep().await.is_ok() {
            let cluster = self
                .rds
                .describe_db_clusters(
                    self.db_cluster_identifier
```

```
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = cluster {
                warn!(?err, "Failed to describe cluster while waiting for
ready");
                continue;
            }

            let instance = self
                .rds
                .describe_db_instance(
                    self.db_instance_identifier
                        .as_deref()
                        .expect("instance identifier"),
                )
                .await;
            if let Err(err) = instance {
                return Err(ScenarioError::new(
                    "Failed to find instance for cluster",
                    &err,
                ));
            }

            let instances_available = instance
                .unwrap()
                .db_instances()
                .iter()
                .all(|instance| instance.db_instance_status() ==
Some("Available"));

            let endpoints = self
                .rds
                .describe_db_cluster_endpoints(
                    self.db_cluster_identifier
                        .as_deref()
                        .expect("cluster identifier"),
                )
                .await;

            if let Err(err) = endpoints {
                return Err(ScenarioError::new(
```

```
                    "Failed to find endpoint for cluster",
                    &err,
                ));
            }

            let endpoints_available = endpoints
                .unwrap()
                .db_cluster_endpoints()
                .iter()
                .all(|endpoint| endpoint.status() == Some("available"));

            if instances_available && endpoints_available {
                return Ok(());
            }
        }

        Err(ScenarioError::with("timed out waiting for cluster"))
    }

    pub async fn describe_db_clusters(
        &self,
        id: &str,
    ) -> Result<DescribeDbClustersOutput, SdkError<DescribeDBClustersError>> {
        self.inner
            .describe_db_clusters()
            .db_cluster_identifier(id)
            .send()
            .await
    }

#[tokio::test]
async fn test_start_cluster_and_instance() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
```

```
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

.db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_describe_db_instance()
        .with(eq("RustSDKCodeExamplesDBInstance"))
        .return_once(|name| {
            Ok(DescribeDbInstancesOutput::builder()
```

```
                .db_instances(
                    DbInstance::builder()
                        .db_instance_identifier(name)
                        .db_instance_status("Available")
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_cluster_endpoints()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
        assert!(scenario
            .password
            .replace(SecretString::new("BAD SECRET".into()))
            .unwrap()
            .expose_secret()
            .is_empty());
        assert_eq!(
            scenario.db_cluster_identifier,
            Some("RustSDKCodeExamplesDBCluster".into())
        );
    });
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

```
#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Err(SdkError::service_error(
                CreateDBClusterError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _}) if message
 == "Failed to create DB Cluster with cluster group")
}

#[tokio::test]
async fn test_start_cluster_and_instance_cluster_create_missing_id() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .return_once(|_, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()
                .db_cluster(DbCluster::builder().build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
```

```
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context:_ }) if message
 == "Created DB Cluster missing Identifier");
}

#[tokio::test]
async fn test_start_cluster_and_instance_instance_create_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .return_once(|_, _, _, _| {
            Err(SdkError::service_error(
                CreateDBInstanceError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "create db instance error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    let create = scenario.start_cluster_and_instance().await;
    assert_matches!(create, Err(ScenarioError { message, context: _ }) if message
 == "Failed to create Instance in DB Cluster")
}

#[tokio::test]
async fn test_start_cluster_and_instance_wait_hiccup() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_create_db_cluster()
        .withf(|id, params, engine, version, username, password| {
            assert_eq!(id, "RustSDKCodeExamplesDBCluster");
            assert_eq!(params, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(engine, "aurora-mysql");
            assert_eq!(version, "aurora-mysql8.0");
            assert_eq!(username, "test username");
            assert_eq!(password.expose_secret(), "test password");
            true
        })
        .return_once(|id, _, _, _, _, _| {
            Ok(CreateDbClusterOutput::builder()

 .db_cluster(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds
        .expect_create_db_instance()
        .withf(|cluster, name, class, engine| {
            assert_eq!(cluster, "RustSDKCodeExamplesDBCluster");
            assert_eq!(name, "RustSDKCodeExamplesDBInstance");
            assert_eq!(class, "m5.large");
            assert_eq!(engine, "aurora-mysql");
            true
        })
        .return_once(|cluster, name, class, _| {
            Ok(CreateDbInstanceOutput::builder()
                .db_instance(
```

```
                    DbInstance::builder()
                        .db_cluster_identifier(cluster)
                        .db_instance_identifier(name)
                        .db_instance_class(class)
                        .build(),
                )
                .build())
        });

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe cluster error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        })
        .with(eq("RustSDKCodeExamplesDBCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()

.db_clusters(DbCluster::builder().db_cluster_identifier(id).build())
                .build())
        });

    mock_rds.expect_describe_db_instance().return_once(|name| {
        Ok(DescribeDbInstancesOutput::builder()
            .db_instances(
                DbInstance::builder()
                    .db_instance_identifier(name)
                    .db_instance_status("Available")
                    .build(),
            )
            .build())
    });

    mock_rds
```

```
        .expect_describe_db_cluster_endpoints()
        .return_once(|_| {
            Ok(DescribeDbClusterEndpointsOutput::builder()

 .db_cluster_endpoints(DbClusterEndpoint::builder().status("available").build())
                .build())
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_version = Some("aurora-mysql8.0".into());
    scenario.instance_class = Some("m5.large".into());
    scenario.username = Some("test username".into());
    scenario.password = Some(SecretString::new("test password".into()));

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let create = scenario.start_cluster_and_instance().await;
        assert!(create.is_ok());
    });

    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::advance(Duration::from_secs(1)).await;
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Pour plus de détails sur l'API, voir Description DBClusters dans le AWSSDK pour la référence de l'API Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DescribeDBEngineVersions`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `DescribeDBEngineVersions`.

Les exemples d’actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l’exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /// <summary>
    /// Get a list of DB engine versions for a particular DB engine.
    /// </summary>
    /// <param name="engine">The name of the engine.</param>
    /// <param name="parameterGroupFamily">Optional parameter group family
name.</param>
    /// <returns>A list of DBEngineVersions.</returns>
    public async Task<List<DBEngineVersion>>
DescribeDBEngineVersionsForEngineAsync(string engine,
        string? parameterGroupFamily = null)
    {
        var response = await _amazonRDS.DescribeDBEngineVersionsAsync(
            new DescribeDBEngineVersionsRequest()
            {
                Engine = engine,
                DBParameterGroupFamily = parameterGroupFamily
            });
        return response.DBEngineVersions;
    }
```

- Pour plus de détails sur l'API, consultez la section Description DBEngine des versions dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

 Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
```

```
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
                client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                    outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                        engineVersions.begin(),
 engineVersions.end());
            marker  = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}
```

- Pour plus de détails sur l'API, consultez la section Description DBEngine des versions dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}



// GetEngineVersions gets database engine versions that are available for the
 specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
 parameterGroupFamily string) (
 []types.DBEngineVersion, error) {
 output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
  &rds.DescribeDBEngineVersionsInput{
   Engine:                 aws.String(engine),
   DBParameterGroupFamily: aws.String(parameterGroupFamily),
  })
 if err != nil {
  log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
  return nil, err
 } else {
  return output.DBEngineVersions, nil
 }
}
```

- Pour plus de détails sur l'API, consultez la section Description DBEngine des versions dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Pour plus de détails sur l'API, consultez la section Description DBEngine des versions dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
// Get a list of allowed engine versions.
suspend fun getAllowedClusterEngines(dbParameterGroupFamilyVal: String?) {
    val versionsRequest =
        DescribeDbEngineVersionsRequest {
            dbParameterGroupFamily = dbParameterGroupFamilyVal
            engine = "aurora-mysql"
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.describeDbEngineVersions(versionsRequest)
        response.dbEngineVersions?.forEach { dbEngine ->
            println("The engine version is ${dbEngine.engineVersion}")
            println("The engine description is ${dbEngine.dbEngineDescription}")
        }
    }
}
```

- Pour plus de détails sur l'API, voir Description des DBEngine versions dans le AWSSDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_engine_versions(self, engine, parameter_group_family=None):
        """
        Gets database engine versions that are available for the specified engine
        and parameter group family.

        :param engine: The database engine to look up.
        :param parameter_group_family: When specified, restricts the returned
 list of
                                       engine versions to those that are
 compatible with
                                       this parameter group family.
        :return: The list of database engine versions.
        """
```

```
        try:
            kwargs = {"Engine": engine}
            if parameter_group_family is not None:
                kwargs["DBParameterGroupFamily"] = parameter_group_family
            response = self.rds_client.describe_db_engine_versions(**kwargs)
            versions = response["DBEngineVersions"]
        except ClientError as err:
            logger.error(
                "Couldn't get engine versions for %s. Here's why: %s: %s",
                engine,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return versions
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Describe DBEngine Versions in AWSSDK for Python (Boto3).

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   // Get available engine families for Aurora MySql.
rds.DescribeDbEngineVersions(Engine='aurora-mysql') and build a set of the
'DBParameterGroupFamily' field values. I get {aurora-mysql8.0, aurora-mysql5.7}.
   pub async fn get_engines(&self) -> Result<HashMap<String, Vec<String>>,
ScenarioError> {
       let describe_db_engine_versions =
self.rds.describe_db_engine_versions(DB_ENGINE).await;
       trace!(versions=?describe_db_engine_versions, "full list of versions");
```

```
        if let Err(err) = describe_db_engine_versions {
            return Err(ScenarioError::new(
                "Failed to retrieve DB Engine Versions",
                &err,
            ));
        };

        let version_count = describe_db_engine_versions
            .as_ref()
            .map(|o| o.db_engine_versions().len())
            .unwrap_or_default();
        info!(version_count, "got list of versions");

        // Create a map of engine families to their available versions.
        let mut versions = HashMap::<String, Vec<String>>::new();
        describe_db_engine_versions
            .unwrap()
            .db_engine_versions()
            .iter()
            .filter_map(
                |v| match (&v.db_parameter_group_family, &v.engine_version) {
                    (Some(family), Some(version)) => Some((family.clone(),
version.clone())),
                    _ => None,
                },
            )
            .for_each(|(family, version)|
versions.entry(family).or_default().push(version));

        Ok(versions)
    }

    pub async fn describe_db_engine_versions(
        &self,
        engine: &str,
    ) -> Result<DescribeDbEngineVersionsOutput,
SdkError<DescribeDBEngineVersionsError>> {
        self.inner
            .describe_db_engine_versions()
            .engine(engine)
            .send()
            .await
    }
```

```
#[tokio::test]
async fn test_scenario_get_engines() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Ok(DescribeDbEngineVersionsOutput::builder()
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1a")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f1")
                        .engine_version("f1b")
                        .build(),
                )
                .db_engine_versions(
                    DbEngineVersion::builder()
                        .db_parameter_group_family("f2")
                        .engine_version("f2a")
                        .build(),
                )
                .db_engine_versions(DbEngineVersion::builder().build())
                .build())
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;

    assert_eq!(
        versions_map,
        Ok(HashMap::from([
            ("f1".into(), vec!["f1a".into(), "f1b".into()]),
            ("f2".into(), vec!["f2a".into()])
        ]))
    );
}
```

```
#[tokio::test]
async fn test_scenario_get_engines_failed() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_db_engine_versions()
        .with(eq("aurora-mysql"))
        .return_once(|_| {
            Err(SdkError::service_error(
                DescribeDBEngineVersionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_db_engine_versions error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let versions_map = scenario.get_engines().await;
    assert_matches!(
        versions_map,
        Err(ScenarioError { message, context: _ }) if message == "Failed to
 retrieve DB Engine Versions"
    );
}
```

- Pour plus de détails sur l'API, consultez la section Description DBEngine des versions dans le AWSSDK pour la référence de l'API Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DescribeDBInstances`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `DescribeDBInstances`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   /// <summary>
   /// Returns a list of DB instances.
   /// </summary>
   /// <param name="dbInstanceIdentifier">Optional name of a specific DB
instance.</param>
   /// <returns>List of DB instances.</returns>
   public async Task<List<DBInstance>> DescribeDBInstancesPagedAsync(string?
dbInstanceIdentifier = null)
   {
       var results = new List<DBInstance>();
       var instancesPaginator = _amazonRDS.Paginators.DescribeDBInstances(
           new DescribeDBInstancesRequest
           {
               DBInstanceIdentifier = dbInstanceIdentifier
           });
       // Get the entire list using the paginator.
       await foreach (var instances in instancesPaginator.DBInstances)
       {
           results.Add(instances);
       }
       return results;
   }
```

- Pour plus de détails sur l'API, voir Description DBInstances dans le manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets a DB instance description.
/*!
 \sa describeDBCluster()
 \param dbInstanceIdentifier: A DB instance identifier.
 \param instanceResult: The 'DBInstance' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                        Aws::RDS::Model::DBInstance
 &instanceResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
```

```
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
        // This example does not log an error if the DB instance does not exist.
        // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}
```

- Pour plus de détails sur l'API, voir Description DBInstances dans le manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)
```

```
type DbClusters struct {
 AuroraClient *rds.Client
}



// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string)
 (
 *types.DBInstance, error) {
 output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
  &rds.DescribeDBInstancesInput{
   DBInstanceIdentifier: aws.String(instanceName),
  })
 if err != nil {
  var notFoundError *types.DBInstanceNotFoundFault
  if errors.As(err, &notFoundError) {
   log.Printf("DB instance %v does not exist.\n", instanceName)
   err = nil
  } else {
   log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
  }
  return nil, err
 } else {
  return &output.DBInstances[0], nil
 }
}
```

- Pour plus de détails sur l'API, voir Description DBInstances dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   // Waits until the database instance is available.
   public static void waitForInstanceReady(RdsClient rdsClient, String
dbClusterIdentifier) {
       boolean instanceReady = false;
       String instanceReadyStr;
       System.out.println("Waiting for instance to become available.");
       try {
           DescribeDbClustersRequest instanceRequest =
DescribeDbClustersRequest.builder()
                   .dbClusterIdentifier(dbClusterIdentifier)
                   .build();

           while (!instanceReady) {
               DescribeDbClustersResponse response =
rdsClient.describeDBClusters(instanceRequest);
               List<DBCluster> clusterList = response.dbClusters();
               for (DBCluster cluster : clusterList) {
                   instanceReadyStr = cluster.status();
                   if (instanceReadyStr.contains("available")) {
                       instanceReady = true;
                   } else {
                       System.out.print(".");
                       Thread.sleep(sleepTime * 1000);
                   }
               }
           }
           System.out.println("Database cluster is available!");

       } catch (RdsException | InterruptedException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
    }
```

- Pour plus de détails sur l'API, voir Description DBInstances dans le manuel de référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
suspend fun waitDBAuroraInstanceReady(dbInstanceIdentifierVal: String?) {
    var instanceReady = false
    var instanceReadyStr: String
    println("Waiting for instance to become available.")
    val instanceRequest =
        DescribeDbInstancesRequest {
            dbInstanceIdentifier = dbInstanceIdentifierVal
        }

    var endpoint = ""
    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        while (!instanceReady) {
            val response = rdsClient.describeDbInstances(instanceRequest)
            response.dbInstances?.forEach { instance ->
                instanceReadyStr = instance.dbInstanceStatus.toString()
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint?.address.toString()
                    instanceReady = true
                } else {
                    print(".")
                    delay(sleepTime * 1000)
                }
            }
        }
    }
```

```
    println("Database instance is available! The connection endpoint is
 $endpoint")
}
```

- Pour plus de détails sur l'API, voir Description DBInstances dans le AWSSDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_db_instance(self, instance_id):
        """
        Gets data about a DB instance.
```

```
        :param instance_id: The ID of the DB instance to retrieve.
        :return: The retrieved DB instance.
        """
        try:
            response = self.rds_client.describe_db_instances(
                DBInstanceIdentifier=instance_id
            )
            db_inst = response["DBInstances"][0]
        except ClientError as err:
            if err.response["Error"]["Code"] == "DBInstanceNotFound":
                logger.info("Instance %s does not exist.", instance_id)
            else:
                logger.error(
                    "Couldn't get DB instance %s. Here's why: %s: %s",
                    instance_id,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return db_inst
```

- Pour plus de détails sur l'API, voir AWSDescribe DBInstances in SDK for Python (Boto3) API Reference.

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    pub async fn clean_up(self) -> Result<(), Vec<ScenarioError>> {
        let mut clean_up_errors: Vec<ScenarioError> = vec![];

        // Delete the instance. rds.DeleteDbInstance.
```

```
        let delete_db_instance = self
            .rds
            .delete_db_instance(
                self.db_instance_identifier
                    .as_deref()
                    .expect("instance identifier"),
            )
            .await;
        if let Err(err) = delete_db_instance {
            let identifier = self
                .db_instance_identifier
                .as_deref()
                .unwrap_or("Missing Instance Identifier");
            let message = format!("failed to delete db instance {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance to delete
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_instances =
self.rds.describe_db_instances().await;
                if let Err(err) = describe_db_instances {
                    clean_up_errors.push(ScenarioError::new(
                        "Failed to check instance state during deletion",
                        &err,
                    ));
                    break;
                }
                let db_instances = describe_db_instances
                    .unwrap()
                    .db_instances()
                    .iter()
                    .filter(|instance| instance.db_cluster_identifier ==
self.db_cluster_identifier)
                    .cloned()
                    .collect::<Vec<DbInstance>>();

                if db_instances.is_empty() {
                    trace!("Delete Instance waited and no instances were found");
                    break;
                }
                match db_instances.first().unwrap().db_instance_status() {
                    Some("Deleting") => continue,
                    Some(status) => {
```

```
                        info!("Attempting to delete but instances is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB instance");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster. rds.DeleteDbCluster.
        let delete_db_cluster = self
            .rds
            .delete_db_cluster(
                self.db_cluster_identifier
                    .as_deref()
                    .expect("cluster identifier"),
            )
            .await;

        if let Err(err) = delete_db_cluster {
            let identifier = self
                .db_cluster_identifier
                .as_deref()
                .unwrap_or("Missing DB Cluster Identifier");
            let message = format!("failed to delete db cluster {identifier}");
            clean_up_errors.push(ScenarioError::new(message, &err));
        } else {
            // Wait for the instance and cluster to fully delete.
rds.DescribeDbInstances and rds.DescribeDbClusters until both are not found.
            let waiter = Waiter::default();
            while waiter.sleep().await.is_ok() {
                let describe_db_clusters = self
                    .rds
                    .describe_db_clusters(
                        self.db_cluster_identifier
                            .as_deref()
                            .expect("cluster identifier"),
                    )
                    .await;
                if let Err(err) = describe_db_clusters {
                    clean_up_errors.push(ScenarioError::new(
```

```
                        "Failed to check cluster state during deletion",
                        &err,
                    ));
                    break;
                }
                let describe_db_clusters = describe_db_clusters.unwrap();
                let db_clusters = describe_db_clusters.db_clusters();
                if db_clusters.is_empty() {
                    trace!("Delete cluster waited and no clusters were found");
                    break;
                }
                match db_clusters.first().unwrap().status() {
                    Some("Deleting") => continue,
                    Some(status) => {
                        info!("Attempting to delete but clusters is in
{status}");
                        continue;
                    }
                    None => {
                        warn!("No status for DB cluster");
                        break;
                    }
                }
            }
        }

        // Delete the DB cluster parameter group.
rds.DeleteDbClusterParameterGroup.
        let delete_db_cluster_parameter_group = self
            .rds
            .delete_db_cluster_parameter_group(
                self.db_cluster_parameter_group
                    .map(|g| {
                        g.db_cluster_parameter_group_name
                            .unwrap_or_else(||
DB_CLUSTER_PARAMETER_GROUP_NAME.to_string())
                    })
                    .as_deref()
                    .expect("cluster parameter group name"),
            )
            .await;
        if let Err(error) = delete_db_cluster_parameter_group {
            clean_up_errors.push(ScenarioError::new(
                "Failed to delete the db cluster parameter group",
```

```
                &error,
            ))
        }

        if clean_up_errors.is_empty() {
            Ok(())
        } else {
            Err(clean_up_errors)
        }
    }

    pub async fn describe_db_instances(
        &self,
    ) -> Result<DescribeDbInstancesOutput, SdkError<DescribeDBInstancesError>> {
        self.inner.describe_db_instances().send().await
    }

#[tokio::test]
async fn test_scenario_clean_up() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| Ok(DescribeDbInstancesOutput::builder().build()));
```

```
    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| Ok(DescribeDbClustersOutput::builder().build()));

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_ok());
    });
```

```
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}

#[tokio::test]
async fn test_scenario_clean_up_errors() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_delete_db_instance()
        .with(eq("MockInstance"))
        .return_once(|_| Ok(DeleteDbInstanceOutput::builder().build()));

    mock_rds
        .expect_describe_db_instances()
        .with()
        .times(1)
        .returning(|| {
            Ok(DescribeDbInstancesOutput::builder()
                .db_instances(
                    DbInstance::builder()
                        .db_cluster_identifier("MockCluster")
                        .db_instance_status("Deleting")
                        .build(),
                )
                .build())
        })
        .with()
        .times(1)
        .returning(|| {
            Err(SdkError::service_error(
                DescribeDBInstancesError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db instances error",
                ))),
```

```
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster()
        .with(eq("MockCluster"))
        .return_once(|_| Ok(DeleteDbClusterOutput::builder().build()));

    mock_rds
        .expect_describe_db_clusters()
        .with(eq("MockCluster"))
        .times(1)
        .returning(|id| {
            Ok(DescribeDbClustersOutput::builder()
                .db_clusters(
                    DbCluster::builder()
                        .db_cluster_identifier(id)
                        .status("Deleting")
                        .build(),
                )
                .build())
        })
        .with(eq("MockCluster"))
        .times(1)
        .returning(|_| {
            Err(SdkError::service_error(
                DescribeDBClustersError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe db clusters error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
SdkBody::empty()),
            ))
        });

    mock_rds
        .expect_delete_db_cluster_parameter_group()
        .with(eq("MockParamGroup"))
        .return_once(|_|
Ok(DeleteDbClusterParameterGroupOutput::builder().build()));

    let mut scenario = AuroraScenario::new(mock_rds);
```

```
    scenario.db_cluster_identifier = Some(String::from("MockCluster"));
    scenario.db_instance_identifier = Some(String::from("MockInstance"));
    scenario.db_cluster_parameter_group = Some(
        DbClusterParameterGroup::builder()
            .db_cluster_parameter_group_name("MockParamGroup")
            .build(),
    );

    tokio::time::pause();
    let assertions = tokio::spawn(async move {
        let clean_up = scenario.clean_up().await;
        assert!(clean_up.is_err());
        let errs = clean_up.unwrap_err();
        assert_eq!(errs.len(), 2);
        assert_matches!(errs.first(), Some(ScenarioError {message, context: _})
 if message == "Failed to check instance state during deletion");
        assert_matches!(errs.get(1), Some(ScenarioError {message, context: _}) if
 message == "Failed to check cluster state during deletion");
    });

    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Instances
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for first
 Describe Cluster
    tokio::time::advance(Duration::from_secs(1)).await; // Wait for second
 Describe Cluster
    tokio::time::resume();
    let _ = assertions.await;
}
```

- Pour plus de détails sur l'API, voir Description DBInstances dans le AWSSDK pour la référence de l'API Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`DescribeOrderableDBInstanceOptions`** avec un AWS SDK ou une CLI

Les exemples de code suivants illustrent comment utiliser `DescribeOrderableDBInstanceOptions`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   /// <summary>
   /// Get a list of orderable DB instance options for a specific
   /// engine and engine version.
   /// </summary>
   /// <param name="engine">Name of the engine.</param>
   /// <param name="engineVersion">Version of the engine.</param>
   /// <returns>List of OrderableDBInstanceOptions.</returns>
   public async Task<List<OrderableDBInstanceOption>>
DescribeOrderableDBInstanceOptionsPagedAsync(string engine, string
engineVersion)
   {
       // Use a paginator to get a list of DB instance options.
       var results = new List<OrderableDBInstanceOption>();
       var paginateInstanceOptions =
_amazonRDS.Paginators.DescribeOrderableDBInstanceOptions(
           new DescribeOrderableDBInstanceOptionsRequest()
           {
               Engine = engine,
               EngineVersion = engineVersion,
```

```
            });
        // Get the entire list using the paginator.
        await foreach (var instanceOptions in
 paginateInstanceOptions.OrderableDBInstanceOptions)
        {
            results.Add(instanceOptions);
        }
        return results;
    }
```

- Pour plus de détails sur l'API, consultez la section DescribeOrderableDBInstanceOptions du manuel de référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);


//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
```

```
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
   std::vector<Aws::String> instanceClasses;
   Aws::String marker; // The marker is used for pagination.
   do {
       Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
       request.SetEngine(engine);
       request.SetEngineVersion(engineVersion);
       if (!marker.empty()) {
           request.SetMarker(marker);
       }

       Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
               client.DescribeOrderableDBInstanceOptions(request);

       if (outcome.IsSuccess()) {
           const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption>
&options =
                   outcome.GetResult().GetOrderableDBInstanceOptions();
           for (const Aws::RDS::Model::OrderableDBInstanceOption &option:
options) {
               const Aws::String &instanceClass = option.GetDBInstanceClass();
               if (std::find(instanceClasses.begin(), instanceClasses.end(),
                             instanceClass) == instanceClasses.end()) {
                   instanceClasses.push_back(instanceClass);
               }
           }
           marker = outcome.GetResult().GetMarker();
       }
       else {
           std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions.
"
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
   } while (!marker.empty());

   std::cout << "The available DB instance classes for your database engine
are:"
             << std::endl;
   for (int i = 0; i < instanceClasses.size(); ++i) {
```

```
        std::cout << "   " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
            "Which DB instance class do you want to use? ",
            1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}
```

- Pour plus de détails sur l'API, consultez la section DescribeOrderableDBInstanceOptions du manuel de référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
 AuroraClient *rds.Client
}
```

```
// GetOrderableInstances uses a paginator to get DB instance options that can be
 used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
 string, engineVersion string) (
 []types.OrderableDBInstanceOption, error) {

 var output *rds.DescribeOrderableDBInstanceOptionsOutput
 var instances []types.OrderableDBInstanceOption
 var err error
 orderablePaginator :=
 rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
  &rds.DescribeOrderableDBInstanceOptionsInput{
   Engine:        aws.String(engine),
   EngineVersion: aws.String(engineVersion),
  })
 for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get orderable DB instances: %v\n", err)
   break
  } else {
   instances = append(instances, output.OrderableDBInstanceOptions...)
  }
 }
 return instances, err
}
```

- Pour plus de détails sur l'API, consultez la section DescribeOrderableDBInstanceOptions du manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void describeDBEngines(RdsClient rdsClient) {
       try {
           DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                   .engine("aurora-mysql")
                   .defaultOnly(true)
                   .maxRecords(20)
                   .build();

           DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
           List<DBEngineVersion> engines = response.dbEngineVersions();

           // Get all DBEngineVersion objects.
           for (DBEngineVersion engineOb : engines) {
               System.out.println("The name of the DB parameter group family for
the database engine is "
                       + engineOb.dbParameterGroupFamily());
               System.out.println("The name of the database engine " +
engineOb.engine());
               System.out.println("The version number of the database engine " +
engineOb.engineVersion());
           }

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }
   }
```

- Pour plus de détails sur l'API, consultez la section DescribeOrderableDBInstanceOptions du manuel de référence des AWS SDK for Java 2.x API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cet exemple répertorie les versions de moteur de base de données qui prennent en charge une classe d'instance de base de données spécifique dans une Région AWS.

```
$params = @{
```

```
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Exemple 2 : cet exemple répertorie les classes d'instance de base de données prises en charge pour une version de moteur de base de données spécifique dans une Région AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Pour plus de détails sur l'API, consultez la section DescribeOrderableDBInstanceOptions de la référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cet exemple répertorie les versions de moteur de base de données qui prennent en charge une classe d'instance de base de données spécifique dans une Région AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  DBInstanceClass = 'db.r5.large'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

Exemple 2 : cet exemple répertorie les classes d'instance de base de données prises en charge pour une version de moteur de base de données spécifique dans une Région AWS.

```
$params = @{
  Engine = 'aurora-postgresql'
  EngineVersion = '13.6'
  Region = 'us-east-1'
}
Get-RDSOrderableDBInstanceOption @params
```

- Pour plus de détails sur l'API, consultez la section DescribeOrderableDBInstanceOptions de la référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def get_orderable_instances(self, db_engine, db_engine_version):
        """
        Gets DB instance options that can be used to create DB instances that are
        compatible with a set of specifications.

        :param db_engine: The database engine that must be supported by the DB
 instance.
        :param db_engine_version: The engine version that must be supported by
 the DB instance.
```

```
        :return: The list of DB instance options that can be used to create a
compatible DB instance.
        """
        try:
            inst_opts = []
            paginator = self.rds_client.get_paginator(
                "describe_orderable_db_instance_options"
            )
            for page in paginator.paginate(
                Engine=db_engine, EngineVersion=db_engine_version
            ):
                inst_opts += page["OrderableDBInstanceOptions"]
        except ClientError as err:
            logger.error(
                "Couldn't get orderable DB instances. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return inst_opts
```

- Pour plus de détails sur l'API, consultez la section DescribeOrderableDBInstanceOptions du AWSSDK pour Python (Boto3) API Reference.

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    pub async fn get_instance_classes(&self) -> Result<Vec<String>,
ScenarioError> {
        let describe_orderable_db_instance_options_items = self
            .rds
```

```
            .describe_orderable_db_instance_options(
                DB_ENGINE,
                self.engine_version
                    .as_ref()
                    .expect("engine version for db instance options")
                    .as_str(),
            )
            .await;

        describe_orderable_db_instance_options_items
            .map(|options| {
                options
                    .iter()
                    .filter(|o| o.storage_type() == Some("aurora"))
                    .map(|o|
 o.db_instance_class().unwrap_or_default().to_string())
                    .collect::<Vec<String>>()
            })
            .map_err(|err| ScenarioError::new("Could not get available instance
 classes", &err))
    }

    pub async fn describe_orderable_db_instance_options(
        &self,
        engine: &str,
        engine_version: &str,
    ) -> Result<Vec<OrderableDbInstanceOption>,
 SdkError<DescribeOrderableDBInstanceOptionsError>>
    {
        self.inner
            .describe_orderable_db_instance_options()
            .engine(engine)
            .engine_version(engine_version)
            .into_paginator()
            .items()
            .send()
            .try_collect()
            .await
    }

#[tokio::test]
async fn test_scenario_get_instance_classes() {
    let mut mock_rds = MockRdsImpl::default();
```

```
    mock_rds
        .expect_create_db_cluster_parameter_group()
        .return_once(|_, _, _| {
            Ok(CreateDbClusterParameterGroupOutput::builder()

.db_cluster_parameter_group(DbClusterParameterGroup::builder().build())
                .build())
        });

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Ok(vec![
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t1")
                    .storage_type("aurora-iopt1")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t2")
                    .storage_type("aurora")
                    .build(),
                OrderableDbInstanceOption::builder()
                    .db_instance_class("t3")
                    .storage_type("aurora")
                    .build(),
            ])
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario
        .set_engine("aurora-mysql", "aurora-mysql8.0")
        .await
        .expect("set engine");

    let instance_classes = scenario.get_instance_classes().await;

    assert_eq!(
        instance_classes,
        Ok(vec!["t1".into(), "t2".into(), "t3".into()])
```

```
    );
}

#[tokio::test]
async fn test_scenario_get_instance_classes_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_describe_orderable_db_instance_options()
        .with(eq("aurora-mysql"), eq("aurora-mysql8.0"))
        .return_once(|_, _| {
            Err(SdkError::service_error(

 DescribeOrderableDBInstanceOptionsError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "describe_orderable_db_instance_options_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let mut scenario = AuroraScenario::new(mock_rds);
    scenario.engine_family = Some("aurora-mysql".into());
    scenario.engine_version = Some("aurora-mysql8.0".into());

    let instance_classes = scenario.get_instance_classes().await;

    assert_matches!(
        instance_classes,
        Err(ScenarioError {message, context: _}) if message == "Could not get
 available instance classes"
    );
}
```

- Pour plus de détails sur l'API, consultez la section DescribeOrderableDBInstanceOptions du AWSSDK pour la référence de l'API Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **`ModifyDBClusterParameterGroup`** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `ModifyDBClusterParameterGroup`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- Principes de base

.NET

SDK pour .NET(v4)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
    /// <summary>
    /// Modify the specified integer parameters with new values from user input.
    /// </summary>
    /// <param name="groupName">The group name for the parameters.</param>
    /// <param name="parameters">The list of integer parameters to modify.</
param>
    /// <param name="newValue">Optional int value to set for parameters.</param>
    /// <returns>The name of the group that was modified.</returns>
    public async Task<string> ModifyIntegerParametersInGroupAsync(string
 groupName, List<Parameter> parameters, int newValue = 0)
    {
        foreach (var p in parameters)
        {
            if (p.IsModifiable.GetValueOrDefault() && p.DataType == "integer")
            {
                while (newValue == 0)
                {
                    Console.WriteLine(
                        $"Enter a new value for {p.ParameterName} from the
 allowed values {p.AllowedValues} ");
```

```
                    var choice = Console.ReadLine();
                    int.TryParse(choice, out newValue);
                }

                p.ParameterValue = newValue.ToString();
            }
        }

        var request = new ModifyDBClusterParameterGroupRequest
        {
            Parameters = parameters,
            DBClusterParameterGroupName = groupName,
        };

        var result = await
_amazonRDS.ModifyDBClusterParameterGroupAsync(request);
        return result.DBClusterParameterGroupName;
    }
```

- Pour plus de détails sur l'API, voir Modifier DBCluster ParameterGroup dans la référence des AWS SDK pour .NET API.

C++

SDK pour C++

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
```

```
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
                client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully
 modified."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Pour plus de détails sur l'API, voir Modifier DBCluster ParameterGroup dans la référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/rds"
 "github.com/aws/aws-sdk-go-v2/service/rds/types"
)
```

```
type DbClusters struct {
 AuroraClient *rds.Client
}



// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context,
 parameterGroupName string, params []types.Parameter) error {
 _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
  &rds.ModifyDBClusterParameterGroupInput{
   DBClusterParameterGroupName: aws.String(parameterGroupName),
   Parameters:                  params,
  })
 if err != nil {
  log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
  return err
 } else {
  return nil
 }
}
```

- Pour plus de détails sur l'API, voir Modifier DBCluster ParameterGroup dans la référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   public static void describeDbClusterParameterGroups(RdsClient rdsClient,
String dbClusterGroupName) {
       try {
```

```
            DescribeDbClusterParameterGroupsRequest groupsRequest =
DescribeDbClusterParameterGroupsRequest.builder()
                    .dbClusterParameterGroupName(dbClusterGroupName)
                    .maxRecords(20)
                    .build();

            List<DBClusterParameterGroup> groups =
rdsClient.describeDBClusterParameterGroups(groupsRequest)
                    .dbClusterParameterGroups();
            for (DBClusterParameterGroup group : groups) {
                System.out.println("The group name is " +
group.dbClusterParameterGroupName());
                System.out.println("The group ARN is " +
group.dbClusterParameterGroupArn());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
```

- Pour plus de détails sur l'API, voir Modifier DBCluster ParameterGroup dans la référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
// Modify the auto_increment_offset parameter.
suspend fun modifyDBClusterParas(dClusterGroupName: String?) {
    val parameter1 =
        Parameter {
            parameterName = "auto_increment_offset"
```

```
            applyMethod = ApplyMethod.fromValue("immediate")
            parameterValue = "5"
        }

    val paraList = ArrayList<Parameter>()
    paraList.add(parameter1)
    val groupRequest =
        ModifyDbClusterParameterGroupRequest {
            dbClusterParameterGroupName = dClusterGroupName
            parameters = paraList
        }

    RdsClient.fromEnvironment { region = "us-west-2" }.use { rdsClient ->
        val response = rdsClient.modifyDbClusterParameterGroup(groupRequest)
        println("The parameter group ${response.dbClusterParameterGroupName} was
 successfully modified")
    }
}
```

- Pour plus de détails sur l'API, voir Modifier DBCluster ParameterGroup dans le AWSSDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
class AuroraWrapper:
    """Encapsulates Aurora DB cluster actions."""

    def __init__(self, rds_client):
        """
        :param rds_client: A Boto3 Amazon Relational Database Service (Amazon
 RDS) client.
        """
```

```
        self.rds_client = rds_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        rds_client = boto3.client("rds")
        return cls(rds_client)


    def update_parameters(self, parameter_group_name, update_parameters):
        """
        Updates parameters in a custom DB cluster parameter group.

        :param parameter_group_name: The name of the parameter group to update.
        :param update_parameters: The parameters to update in the group.
        :return: Data about the modified parameter group.
        """
        try:
            response = self.rds_client.modify_db_cluster_parameter_group(
                DBClusterParameterGroupName=parameter_group_name,
                Parameters=update_parameters,
            )
        except ClientError as err:
            logger.error(
                "Couldn't update parameters in %s. Here's why: %s: %s",
                parameter_group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- Pour plus de détails sur l'API, consultez le manuel de référence de l'API Modify DBCluster ParameterGroup in AWSSDK for Python (Boto3).

Rust

SDK pour Rust

> **Note**
>
> Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'exemples de code AWS.

```
   // Modify both the auto_increment_offset and auto_increment_increment
parameters in one call in the custom parameter group. Set their ParameterValue
fields to a new allowable value. rds.ModifyDbClusterParameterGroup.
   pub async fn update_auto_increment(
       &self,
       offset: u8,
       increment: u8,
   ) -> Result<(), ScenarioError> {
       let modify_db_cluster_parameter_group = self
           .rds
           .modify_db_cluster_parameter_group(
               DB_CLUSTER_PARAMETER_GROUP_NAME,
               vec![
                   Parameter::builder()
                       .parameter_name("auto_increment_offset")
                       .parameter_value(format!("{offset}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
                   Parameter::builder()
                       .parameter_name("auto_increment_increment")
                       .parameter_value(format!("{increment}"))
                       .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                       .build(),
               ],
           )
           .await;

       if let Err(error) = modify_db_cluster_parameter_group {
           return Err(ScenarioError::new(
               "Failed to modify cluster parameter group",
               &error,
           ));
```

```
        }

        Ok(())
    }

    pub async fn modify_db_cluster_parameter_group(
        &self,
        name: &str,
        parameters: Vec<Parameter>,
    ) -> Result<ModifyDbClusterParameterGroupOutput,
 SdkError<ModifyDBClusterParameterGroupError>>
    {
        self.inner
            .modify_db_cluster_parameter_group()
            .db_cluster_parameter_group_name(name)
            .set_parameters(Some(parameters))
            .send()
            .await
    }

#[tokio::test]
async fn test_scenario_update_auto_increment() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .withf(|name, params| {
            assert_eq!(name, "RustSDKCodeExamplesDBParameterGroup");
            assert_eq!(
                params,
                &vec![
                    Parameter::builder()
                        .parameter_name("auto_increment_offset")
                        .parameter_value("10")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                    Parameter::builder()
                        .parameter_name("auto_increment_increment")
                        .parameter_value("20")
                        .apply_method(aws_sdk_rds::types::ApplyMethod::Immediate)
                        .build(),
                ]
            );
            true
```

```
        })
        .return_once(|_, _|
 Ok(ModifyDbClusterParameterGroupOutput::builder().build()));

    let scenario = AuroraScenario::new(mock_rds);

    scenario
        .update_auto_increment(10, 20)
        .await
        .expect("update auto increment");
}

#[tokio::test]
async fn test_scenario_update_auto_increment_error() {
    let mut mock_rds = MockRdsImpl::default();

    mock_rds
        .expect_modify_db_cluster_parameter_group()
        .return_once(|_, _| {
            Err(SdkError::service_error(

 ModifyDBClusterParameterGroupError::unhandled(Box::new(Error::new(
                    ErrorKind::Other,
                    "modify_db_cluster_parameter_group_error",
                ))),
                Response::new(StatusCode::try_from(400).unwrap(),
 SdkBody::empty()),
            ))
        });

    let scenario = AuroraScenario::new(mock_rds);

    let update = scenario.update_auto_increment(10, 20).await;
    assert_matches!(update, Err(ScenarioError { message, context: _}) if message
 == "Failed to modify cluster parameter group");
}
```

- Pour plus de détails sur l'API, voir Modifier DBCluster ParameterGroup dans le AWSSDK pour la référence de l'API Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Scénarios d'utilisation d'Aurora AWS SDKs

Les exemples de code suivants vous montrent comment implémenter des scénarios courants dans Aurora avec AWSSDKs. Ces scénarios vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions dans Aurora ou en les combinant avec d'autres Services AWS. Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code.

Les scénarios ciblent un niveau d'expérience intermédiaire pour vous aider à comprendre les actions de service dans leur contexte.

Exemples

- Créer une API REST de bibliothèque de prêt
- Créer un outil de suivi des éléments de travail sans serveur Aurora

Créer une API REST de bibliothèque de prêt

L'exemple de code suivant montre comment créer une bibliothèque de prêt dans laquelle les clients peuvent emprunter et retourner des livres à l'aide d'une API REST soutenue par une base de données Amazon Aurora.

Python

Kit SDK for Python (Boto3)

Montre comment utiliser l'AWS SDK pour Python (Boto3)API Amazon Relational Database Service (Amazon RDS) et AWS Chalice pour créer une API REST soutenue par une base de données Amazon Aurora. Le service Web est entièrement sans serveur et représente une bibliothèque de prêt simple où les clients peuvent emprunter et retourner des livres. Découvrez comment :

- Créer et gérer un cluster de bases de données Aurora sans serveur.
- AWS Secrets ManagerÀ utiliser pour gérer les informations d'identification de base de données.

- Implémenter une couche de stockage de données qui utilise Amazon RDS pour déplacer des données vers et hors de la base de données.
- Utilisez AWS Chalice pour déployer une API REST sans serveur sur Amazon API Gateway et. AWS Lambda
- Utiliser le package Requests (Requêtes) pour envoyer des requêtes au service web.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- API Gateway
- Aurora
- Lambda
- Secrets Manager

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Créer un outil de suivi des éléments de travail sans serveur Aurora

Les exemples de code suivants montrent comment créer une application web qui suit des éléments de travail dans une base de données Amazon Aurora sans serveur et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES).

.NET

SDK pour .NET

Montre comment utiliser le AWS SDK pour .NET pour créer une application Web qui suit les éléments de travail dans une base de données Amazon Aurora et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES). Cet exemple utilise un front-end créé avec React.js pour interagir avec un backend RESTful .NET.

- Intégrez une application Web React à AWS des services.
- Listez, ajoutez et mettez à jour des éléments dans une table Aurora.
- Envoyez un rapport par e-mail sur les éléments de travail filtrés à l'aide d'Amazon SES.

- Déployez et gérez des exemples de ressources à l'aide du AWS CloudFormation script inclus.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

C++

SDK pour C++

Montre comment créer une application web qui suit et génère des rapports sur les éléments de travail stockés dans une base de données Amazon Aurora sans serveur.

Pour obtenir le code source complet et les instructions sur la façon de configurer une API REST C++ qui interroge les données Amazon Aurora Serverless et à utiliser par une application React, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Java

SDK pour Java 2.x

Montre comment créer une application web qui suit et génère des rapports sur les éléments de travail stockés dans une base de données Amazon RDS.

Pour obtenir le code source complet et les instructions sur la façon de configurer une API Spring REST qui interroge les données Amazon Aurora Serverless et pour une utilisation par une application React, consultez l'exemple complet sur GitHub.

Pour obtenir le code source complet et les instructions sur la façon de configurer et d'exécuter un exemple utilisant l'API JDBC, consultez l'exemple complet sur. GitHub

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

JavaScript

SDK pour JavaScript (v3)

Montre comment utiliser le AWS SDK pour JavaScript (v3) pour créer une application Web qui suit les éléments de travail dans une base de données Amazon Aurora et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES). Cet exemple utilise un front end créé avec React.js pour interagir avec un backend Express Node.js.

- Intégrez une application Web React.js àServices AWS.
- Lister, ajouter et mettre à jour des éléments dans une table Aurora.
- Envoyez un rapport par e-mail sur les éléments de travail filtrés en utilisant Amazon SES.
- Déployez et gérez des exemples de ressources à l'aide du AWS CloudFormation script inclus.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Kotlin

SDK pour Kotlin

Montre comment créer une application web qui suit et génère des rapports sur les éléments de travail stockés dans une base de données Amazon RDS.

Pour obtenir le code source complet et les instructions sur la façon de configurer une API Spring REST qui interroge les données Amazon Aurora Serverless et pour une utilisation par une application React, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

PHP

Kit SDK pour PHP

Montre comment utiliser le AWS SDK pour PHP pour créer une application Web qui suit les éléments de travail dans une base de données Amazon RDS et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES). Cet exemple utilise un front-end créé avec React.js pour interagir avec un backend RESTful PHP.

- Intégrez une application Web React.js à AWS des services.
- Répertoriez, ajoutez, mettez à jour et supprimez des éléments dans une table Amazon RDS.
- Envoyez un rapport par e-mail sur les éléments de travail filtrés à l'aide d'Amazon SES.
- Déployez et gérez des exemples de ressources à l'aide du AWS CloudFormation script inclus.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS

- Services de données Amazon RDS
- Amazon SES

Python

SDK pour Python (Boto3)

Montre comment utiliser le AWS SDK pour Python (Boto3) pour créer un service REST qui suit les éléments de travail dans une base de données Amazon Aurora Serverless et envoie des rapports par e-mail à l'aide d'Amazon Simple Email Service (Amazon SES). Cet exemple utilise la structure web Flask pour gérer le routage HTTP et s’intègre à une page web React pour présenter une application web entièrement fonctionnelle.

- Créez un service Flask REST qui s'intègre àServices AWS.
- Lisez, écrivez et mettez à jour les éléments de travail stockés dans une base de données Aurora sans serveur.
- Créez un AWS Secrets Manager secret contenant les informations d'identification de la base de données et utilisez-le pour authentifier les appels à la base de données.
- Utilisez Amazon SES pour envoyer des rapports par e-mail sur les éléments de travail.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur GitHub.

Les services utilisés dans cet exemple

- Aurora
- Amazon RDS
- Services de données Amazon RDS
- Amazon SES

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultezUtilisation de ce service avec un AWS SDK. Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Bonnes pratiques avec Amazon Aurora

Vous trouverez ci-après des informations sur les bonnes pratiques générales et les options d'utilisation des données ou de leur migration vers un cluster de bases de données Amazon Aurora.

Certaines bonnes pratiques avec Amazon Aurora sont spécifiques à un moteur de base de données particulier. Pour plus d'informations sur les bonnes pratiques Aurora spécifiques à un moteur de base de données, consultez les sections suivantes.

Moteur de base de données	Bonnes pratiques
Amazon Aurora MySQL	Consultez Bonnes pratiques avec Amazon Aurora MySQL
Amazon Aurora PostgreSQL	Consultez Bonnes pratiques avec Amazon Aurora PostgreSQL

> **Note**
>
> Pour obtenir des recommandations communes pour Aurora, consultez Recommandations d'Amazon Aurora.

Rubriques

Directives opérationnelles de base pour Amazon Aurora

Voici les directives opérationnelles de base que toute personne doit suivre lorsqu'elle utilise Amazon Aurora. Le contrat de niveau de service (SLA) Amazon RDS exige que vous suiviez les directives ci-après :

- Surveillez votre utilisation de la mémoire, du processeur et du stockage. Vous pouvez configurer Amazon CloudWatch pour être informé quand les habitudes d'utilisation changent ou quand vous arrivez au bout de votre capacité de déploiement. De cette façon, vous pouvez maintenir les performances et la disponibilité du système.
- Si votre application cliente met en cache les données DNS (Domain Name Service) de vos instances de base de données, définissez une valeur durée de vie (TTL) de moins de 30 secondes. L'adresse IP sous-jacente d'une instance de base de données peut changer après un basculement. Ainsi, la mise en cache des données DNS pour une durée prolongée peut entraîner des échecs de connexion si votre application tente de se connecter à une adresse IP qui n'est plus en service. Les clusters de bases de données Aurora avec plusieurs réplicas en lecture peuvent également rencontrer des problèmes de connexion lorsque les connexions utilisent le point de terminaison du lecteur et que l'une des instances de réplica en lecture est en maintenance ou est supprimée.
- Testez le basculement pour votre cluster de bases de données afin de connaître la durée du processus pour votre cas d'utilisation. Le test de basculement peut vous aider à veiller à ce que l'application qui accède à votre cluster de bases de données puisse automatiquement se connecter à la nouvelle instance de base de données suite au basculement.

Recommandations RAM d'une instance de base de données

Pour optimiser les performances, attribuez suffisamment de RAM pour que votre ensemble de travail réside presque totalement en mémoire. Pour déterminer si votre ensemble de travail est presque totalement dans la mémoire, examinez les métriques suivantes dans Amazon CloudWatch :

- `VolumeReadIOPS` – Cette métrique mesure le nombre moyen d'opérations d'E/S de lecture depuis un volume de cluster, rapportées par intervalles de 5 minutes. La valeur de `VolumeReadIOPS` doit être faible et stable. Dans certains cas, vous pouvez constater que vos E/S en lecture augmentent ou sont plus élevées que d'habitude. Dans ce cas, examinez les instances de base de données de votre cluster de bases de données pour voir quelles instances de base de données provoquent l'augmentation des E/S.

> **Tip**
>
> Si votre cluster Aurora MySQL utilise une requête parallèle, vous pouvez voir une augmentation des valeurs de `VolumeReadIOPS`. Les requêtes parallèles n'utilisent pas

le pool de mémoires tampons. Ainsi, bien que les requêtes soient rapides, ce traitement optimisé peut entraîner une augmentation des opérations de lecture et des frais associés.

- `BufferCacheHitRatio` – Cette métrique mesure le pourcentage de demandes qui sont traitées par le cache des tampons d'uns instance de base de données dans votre cluster de bases de données. Cette métrique vous donnes des informations sur la quantité de données traitées par la mémoire.

  Un taux de réussite élevé indique que votre instance de base de données dispose de suffisamment de mémoire. Un faible taux de réussite indique que vos requêtes sur cette instance de base de données vont fréquemment sur le disque. Examinez votre charge de travail afin de savoir quelles requêtes entraînent ce comportement.

Si, après avoir examiné votre charge de travail, vous découvrez que vous avez besoin de plus de mémoire, envisagez de mettre à l'échelle vers le haut la classe d'instance de base de données vers une classe disposant de davantage de RAM. Vous pouvez ensuite examiner les métriques dont il a été question précédemment et poursuivre la mise à l'échelle si nécessaire. Si votre cluster Aurora est supérieur à 40 To, n'utilisez pas les classes d'instance db.t2, db.t3 ou db.t4g.

Pour plus d'informations, consultez CloudWatch Métriques Amazon pour Amazon Aurora.

Pilotes de base de données AWS

Nous recommandons la suite de pilotes AWS pour la connectivité des applications. Les pilotes ont été conçus pour accélérer les temps de bascule et de basculement, ainsi que pour l'authentification avec AWS Secrets Manager, Gestion des identités et des accès AWS (IAM) et l'identité fédérée. Les pilotes AWS s'appuient sur la surveillance de l'état du cluster de bases de données et sur la connaissance de la topologie du cluster pour déterminer le nouvel enregistreur. Cette approche réduit les temps de bascule et de basculement à moins de 10 secondes, contre des dizaines de secondes pour les pilotes open source.

À mesure que de nouvelles fonctionnalités de service sont introduites, l'objectif de la suite de pilotes AWS est de prendre en charge ces fonctionnalités de service de manière intégrée.

Pour plus d'informations, consultez Connexion aux clusters de bases de données Aurora avec les pilotes AWS.

Surveillance de Amazon Aurora

Amazon Aurora propose diverses métriques et analyses que vous pouvez surveiller pour connaître l'état et les performances de votre cluster de bases de données Aurora. Vous pouvez utiliser divers outils, tels que l'AWS Management Console, l'AWS CLI et l'API CloudWatch, pour afficher les métriques Aurora. Vous pouvez consulter les métriques Performance Insights et CloudWatch combinées dans le tableau de bord Performance Insights et surveiller votre instance de base de données. Si vous souhaitez utiliser cette vue de surveillance, Performance Insights doit être activé pour votre instance de base de données. Pour obtenir des informations sur cette vue de surveillance, consultez Affichage des métriques combinées avec le tableau de bord Performance Insights.

Vous pouvez créer un rapport d'analyse des performances pour une période spécifique et consulter les informations identifiées et les recommandations pour résoudre les problèmes. Pour plus d'informations, consultez Création d'un rapport d'analyse des performances dans Performance Insights.

Utilisation de groupes de paramètres de base de données et de groupes de paramètres de cluster de bases de données

Nous vous recommandons de tester les modifications apportées aux groupes de paramètres de base de données et aux groupes de paramètres de cluster de bases de données sur un cluster de bases de données test avant d'appliquer ces modifications à votre cluster de bases de données de production. La configuration incorrecte des paramètres de moteur de base de données peut avoir des effets contraires involontaires, notamment une dégradation de la performance et une instabilité du système.

Montrez-vous toujours prudent lorsque vous modifiez des paramètres de moteur de base de données et sauvegardez votre cluster de bases de données avant de modifier un groupe de paramètres de base de données. Pour plus d'informations sur la sauvegarde de votre cluster de bases de données, consultez Sauvegarde et restauration d'un cluster de bases de données Amazon Aurora.

Vidéo des bonnes pratiques pour Amazon Aurora

La chaîne AWS Online Tech Talks sur YouTube inclut une vidéo de présentation des bonnes pratiques pour créer et configurer un cluster de bases de données Amazon Aurora davantage sécurisé et hautement disponible. Consultez Bonnes pratiques Amazon Aurora pour la haute disponibilité.

Réalisation d’une démonstration de faisabilité avec Amazon Aurora

Vous trouverez ci-après une explication de la manière de configurer et d’exécuter une démonstration de faisabilité pour Aurora. Une démonstration de faisabilité est une investigation que vous faites pour voir si Aurora convient bien à votre application. La démonstration de faisabilité peut vous aider à comprendre les fonctions d’Aurora dans le contexte de vos propres applications de base de données, ainsi qu’à comparer Aurora à votre environnement de base de données actuel. Elle peut également vous montrer le niveau d’effort dont vous avez besoin pour déplacer les données, transposer le code SQL, ajuster les performances et adapter vos procédures de gestion actuelles.

Dans cette rubrique, vous trouverez une présentation et une description pas à pas des décisions et des procédures de haut niveau impliquées dans l’exécution d’une démonstration de faisabilité, répertoriées ci-après. Pour obtenir des instructions détaillées, vous pouvez suivre les liens vers la documentation complète pour des sujets spécifiques.

Présentation d’une démonstration de faisabilité Aurora

Lorsque vous réalisez une démonstration de faisabilité pour Amazon Aurora, vous voyez ce qu’il faut faire pour déplacer vos données et vos applications SQL existantes vers Aurora. Vous étudiez les aspects importants d’Aurora à grande échelle, en utilisant un volume de données et d’activités représentatif de votre environnement de production. L’objectif est de prendre confiance dans le fait que les points forts d’Aurora concordent bien avec les défis qui vous amènent à dépasser votre infrastructure de base de données précédente. À la fin d’une démonstration de faisabilité, vous disposez d’un plan concret pour effectuer des tests d’application et une comparaison des performances à plus grande échelle. À ce stade, vous comprenez les principaux éléments de travail qui vous attendent sur le chemin d’un déploiement en production.

Les conseils qui vous seront donnés sur les bonnes pratiques peuvent vous éviter des erreurs courantes qui posent problème lors des essais comparatifs. Toutefois, cette rubrique ne couvre pas les processus pas à pas de réalisation des essais comparatifs ni d’ajustement des performances. Ces procédures varient selon votre charge de travail et les fonctions Aurora que vous utilisez. Pour obtenir des informations détaillées, consultez la documentation relative aux performances, telle que Gestion des performances et dimensionnement des clusters de bases de données Aurora,

Améliorations des performances Amazon Aurora MySQL, Performance et mise à l'échelle d'Amazon Aurora PostgreSQL et Surveillance de la charge de la base de données avec Performance Insights sur .

Les informations de cette rubrique s'appliquent principalement aux applications dans lesquelles votre organisation écrit du code et conçoit un schéma, et qui prennent en charge les moteurs de bases de données open source MySQL et PostgreSQL. Si vous testez une application commerciale ou du code généré par une infrastructure d'application, vous n'avez peut-être pas la flexibilité d'appliquer toutes les consignes. Dans ce cas, tournez-vous vers votre représentant AWS pour voir s'il existe des études de cas ou de bonnes pratiques Aurora pour votre type d'application.

1. Identifier vos objectifs

Lorsque vous évaluez Aurora dans le cadre d'une démonstration de faisabilité, vous choisissez les mesures à effectuer et la manière d'évaluer la réussite de l'exercice.

Vous devez vous assurez que toutes les fonctionnalités de votre application sont compatibles avec Aurora. Les versions majeures d'Aurora étant compatibles avec les versions majeures correspondantes de MySQL et PostgreSQL, la plupart des applications développées pour ces moteurs sont également compatibles avec Aurora. Toutefois, vous devez encore valider la compatibilité application par application.

Par exemple, certains choix de configuration que vous faites lorsque vous configurez un cluster Aurora influencent le fait que vous puissiez ou deviez utiliser des fonctions de base de données particulières. Vous pouvez commencer avec le type de cluster Aurora à l'usage le plus général, appelé provisionné. Vous pouvez alors décider si une configuration spécialisée, telle qu'une requête parallèle ou sans serveur, offre des avantages pour votre charge de travail.

Utilisez les questions suivantes pour mieux identifier et quantifier vos objectifs :

- Aurora prend-il en charge tous les cas d'utilisation fonctionnels de votre charge de travail ?
- Quels niveau de charge ou taille de jeu de données voulez-vous ? Pouvez-vous effectuer une mise à l'échelle pour atteindre ce niveau ?
- Quelles sont vos besoins spécifiques en matière de latence et de débit de requêtes ? Pouvez-vous parvenir à les satisfaire ?
- Quelles sont les durées d'indisponibilité planifiées et non planifiées minimum acceptables pour votre charge de travail ? Pouvez-vous parvenir à cela ?

- Quelles sont les métriques nécessaires pour assurer l'efficacité opérationnelle ? Pouvez-vous les surveiller avec précision ?
- Aurora prend-il en charge vos objectifs professionnels spécifiques, tels que la réduction des coûts, la croissance du déploiement ou la vitesse de mise en service ? Avez-vous une possibilité de quantifier ces objectifs ?
- Pouvez-vous satisfaire toutes les exigences de sécurité et de compatibilité pour votre charge de travail ?

Prenez le temps d'acquérir les connaissances nécessaires sur les capacités de plateforme et les moteurs de base de données Aurora, et passez en revue la documentation du service. Prenez bonne note de toutes les fonctionnalités qui peuvent vous aider à atteindre les résultats que vous souhaitez. L'une d'elles peut être la consolidation de la charge de travail, décrite dans le billet de blog AWS Database How to plan and optimize Amazon Aurora with MySQL compatibility for consolidated workloads. Une autre peut être la mise à l'échelle basée sur la demande, décrite dans Amazon Aurora Auto Scaling avec des réplicas Aurora, dans le Guide de l'utilisateur Amazon Aurora. D'autres peuvent être liées aux gains de performances ou à la simplification des opérations de base de données.

2. Comprendre les caractéristiques de votre charge de travail

Évaluez Aurora dans le contexte du cas d'utilisation que vous envisagez. Aurora est un choix judicieux pour les charges de travail de traitement de transaction en ligne (OLTP). Vous pouvez également exécuter des rapports sur le cluster qui détient les données OLTP en temps réel sans mettre en service un cluster d'entrepôt de données séparé. Vous pouvez déterminer si votre cas d'utilisation appartient à l'une de ces catégories en recherchant les caractéristiques suivantes :

- Haute simultanéité avec des dizaines, des centaines ou des milliers de clients simultanés.
- Grand volume de requêtes à faible latence (de quelques millisecondes à quelques secondes).
- Transactions brèves en temps réel.
- Modèles de requêtes hautement sélectifs avec recherches basées sur les index.
- Pour HTAP, requêtes analytiques qui peuvent tirer profit des requêtes parallèles d'Aurora.

L'un des facteurs clés affectant vos choix de base de données est la vitesse des données. Une grande vitesse implique des insertions et des mises à jour très fréquentes des données. Un tel système peut compter des milliers de connexions et des centaines de milliers de requêtes

simultanées de lecture et d'écriture dans une base de données. Dans les systèmes à grande vitesse, les requêtes affectent habituellement un nombre relativement faible de lignes et accèdent généralement à plusieurs colonnes d'une même ligne.

Aurora est conçu pour traiter les données à grande vitesse. Selon la charge de travail, un cluster Aurora doté d'une instance de base de données r4.16xlarge individuelle peut traiter plus de 600 000 instructions `SELECT` par seconde. Toujours en fonction de la charge de travail, un tel cluster peut traiter 200 000 instructions `INSERT`, `UPDATE` et `DELETE` par seconde. Aurora est une base de données de stockage de lignes parfaitement adaptée aux charges de travail OLTP à volume élevé, à haut débit et hautement parallélisées.

Aurora peut également exécuter des requêtes de rapport sur le cluster qui traite la charge de travail OLTP. Aurora prend en charge jusqu'à 15 réplicas, chacun intervenant en moyenne à 10–20 millisecondes de l'instance principale. Les analystes peuvent interroger les données OLTP en temps réel sans copier les données dans un cluster d'entrepôt de données séparé. Avec les clusters Aurora qui utilisent la fonction de requête parallèle, vous pouvez décharger une grande partie du travail de traitement, de filtrage et d'agrégation dans le sous-système de stockage Aurora distribué massivement.

Utilisez cette phase de planification pour vous familiariser avec les capacités d'Aurora, d'autres services AWS, de l'AWS Management Console et de l'AWS CLI. De plus, vérifiez comment ces éléments fonctionnent avec les autres outils que vous envisagez d'utiliser dans la démonstration de faisabilité.

3. Acquérir de l'expérience avec la AWS Management Console ou l'AWS CLI

Dans un deuxième temps, vous vous exercez à utiliser l'AWS Management Console ou l'AWS CLI afin de vous familiariser avec ces outils et Aurora.

Acquérir de l'expérience avec la AWS Management Console

Les activités initiales suivantes avec les clusters de bases de données Aurora ont pour but principal de vous familiariser avec l'environnement de l'AWS Management Console et de vous laisser vous exercer à la configuration et à la modification des clusters Aurora. Si vous utilisez les moteurs de base de données compatibles avec MySQL et PostgreSQL avec Amazon RDS, vous pouvez vous appuyer sur ces connaissances lorsque vous utilisez Aurora.

En tirant profit du modèle de stockage partagé Aurora et de fonctions telles que la réplication et les instantanés, vous pouvez traiter des clusters de bases de données complets comme un autre type d'objet que vous pouvez manipuler librement. Vous pouvez configurer, supprimer et modifier fréquemment la capacité des clusters Aurora au cours de la démonstration de faisabilité. Vous n'êtes pas tenu de conserver vos choix précoces en matière de capacité, de paramètres de base de données et de disposition de données physiques.

Pour commencer, configurez un cluster Aurora vide. Choisissez le type de capacité provisionné et un emplacement régional pour vos expériences initiales.

Connectez-vous à ce cluster en utilisant un programme client tel que l'application de ligne de commande SQL. Au départ, vous vous connectez à l'aide du point de terminaison de cluster. Vous vous connectez à ce point de terminaison pour effectuer toutes les opérations d'écriture, telles que les instructions en langage de manipulation de données (DDL) et les processus d'extraction, de transformation et de chargement. Ultérieurement dans la démonstration de faisabilité, vous connectez des sessions impliquant beaucoup de requêtes à l'aide du point de terminaison de lecteur, lequel distribue la charge de travail des requêtes entre plusieurs instances de base de données dans le cluster.

Effectuez une montée en charge du cluster en ajoutant d'autres réplicas Aurora. Pour ces procédures, consultez Réplication avec Amazon Aurora. Mettez à l'échelle, à la hausse ou à la baisse, les instances de base de données en modifiant la classe d'instance AWS. Découvrez comment Aurora simplifie ces types d'opérations, de sorte que si vos estimations initiales de capacité système sont inexactes, vous pouvez effectuer des ajustements ultérieurs sans tout recommencer.

Créez un instantané et restaurez-le dans un cluster différent.

Examinez les métriques du cluster pour voir ses activités au fil du temps et la manière dont ces métriques s'appliquent aux instances de base de données dans le cluster.

Il est utile de vous familiariser avec la manière d'effectuer ces opérations via la AWS Management Console au début. Une fois que vous comprenez ce que vous pouvez faire avec Aurora, vous pouvez progresser et automatiser ces opérations à l'aide de AWS CLI. Dans les sections suivantes, vous trouverez plus de détails sur les procédures et les bonnes pratiques relatives à ces activités, à effectuer au cours de la période de démonstration de faisabilité.

Acquérir de l'expérience avec la AWS CLI

Nous recommandons l'automatisation des procédures de déploiement et de gestion, même dans un paramétrage de démonstration de faisabilité. Pour cela, familiarisez-vous avec l'AWS CLI si vous

ne l'avez pas déjà fait. Si vous utilisez les moteurs de base de données compatibles avec MySQL et PostgreSQL avec Amazon RDS, vous pouvez vous appuyer sur ces connaissances lorsque vous utilisez Aurora.

Aurora implique généralement des groupes d'instances de base de données organisés en clusters. Ainsi, de nombreuses opérations impliquent de déterminer quelles instances de base de données sont associées à un cluster, puis d'effectuer les opérations administratives dans une boucle pour toutes ces instances.

Par exemple, vous pouvez automatiser des étapes telles que la création de clusters Aurora, puis leur mise à l'échelle ascendante avec des classes d'instance plus grandes ou leur montée en charge avec des instances de base de données supplémentaires. Cela vous aidera à répéter des phases quelconques de votre démonstration de faisabilité et à explorer des scénarios hypothétiques avec différents types de configuration de clusters Aurora.

Découvrez les capacités et les limites d'outils de déploiement d'infrastructure tels qu AWS CloudFormation. Vous pouvez constater que des activités que vous effectuez dans le contexte d'une démonstration de faisabilité ne sont pas adaptées à une utilisation en production. Par exemple, le comportement d'CloudFormation pour la modification consiste à créer une instance et à supprimer l'instance actuelle, y compris ses données. Pour plus de détails sur ce comportement, consultez Comportements de mise à jour des ressources d'une pile dans le Guide de l'utilisateur AWS CloudFormation.

4. Créer votre cluster Aurora

Avec Aurora, vous pouvez explorer des scénarios hypothétiques en ajoutant des instances de base de données au cluster et en effectuant une mise à l'échelle ascendante des instances de base de données vers des classes d'instance plus puissantes. Vous pouvez également créer des clusters avec différents paramètres de configuration pour exécuter côte à côte la même charge de travail. Avec Aurora, vous disposez d'une grande flexibilité pour configurer, supprimer et reconfigurer des clusters de bases de données. Partant de là, il est utile de pratiquer ces techniques dans les premières phases du processus de démonstration de faisabilité. Pour découvrir les procédures générales permettant de créer des clusters Aurora, consultez Création d'un cluster de bases de données Amazon Aurora.

Là où cela est possible, commencez avec un cluster en utilisant les paramètres suivants. Ignorez cette étape seulement si vous avez certains cas d'utilisation spécifiques en tête. Par exemple, vous pouvez ignorer cette étape si votre cas d'utilisation requiert un type de cluster Aurora spécialisé.

Vous pouvez également l’ignorer si vous avez besoin d’une combinaison particulière de version et de moteur de base de données.

- Désactivez Easy create (Création facile). Pour la démonstration de faisabilité, nous vous recommandons d’être conscient de tous les paramètres que vous choisissez afin de pouvoir créer ultérieurement des clusters identiques ou légèrement différents.
- Utilisez une version de moteur de base de données récente. Ces combinaisons de version et de moteur de base de données présentent une compatibilité étendue avec les autres fonctionnalités Aurora, ainsi qu’une utilisation importante des clients pour les applications de production.
  - Aurora MySQL version 3.x (compatible avec MySQL 8.0)
  - Aurora PostgreSQL version 15.x ou 16.x
- Choisissez le modèle Dev/Test. Ce choix n’est pas significatif pour vos activités de démonstration de faisabilité.
- Pour DB instance class (Classe d’instance de base de données), choisissez Memory optimized classes (Classes à mémoire optimisée) et l’une des classes d’instance xlarge. Vous pouvez ajuster la classe d’instance ultérieurement, vers le haut ou le bas.
- Sous Multi-AZ Deployment (Déploiement multi-AZ), choisissez Create an Aurora Replica or Reader node in a different AZ (Créer un réplica Aurora ou un nœud de lecteur dans une autre AZ. Un grand nombre des aspects les plus utiles d’Aurora impliquent des clusters de plusieurs instances de base de données. Il est judicieux de toujours commencer avec au moins deux instances de base de données dans tout nouveau cluster. L’utilisation d’une autre zone de disponibilité pour la seconde instance de base de données permet de mieux tester les différents scénarios à haute disponibilité.
- Lorsque vous sélectionnez des noms pour les instances de base de données, utilisez une convention de nommage générique. Ne faites référence à aucune instance de base de données de cluster en tant qu’« enregistreur », car différentes instances de base de données assument ces rôles, selon les besoins. Nous vous recommandons d’utiliser quelque chose comme `clustername-az-serialnumber`, par exemple `myprodappdb-a-01`. Ces éléments identifient de manière unique l’instance de base de données et son placement.
- Définissez une valeur élevée de conservation des sauvegardes pour le cluster Aurora. Avec une longue période de conservation, vous pouvez effectuer une reprise ponctuelle sur une période de 35 jours. Vous pouvez réinitialiser votre base de données à un état connu après l’exécution de tests impliquant des instructions DDL et DML (langage de manipulation de données). Vous pouvez également effectuer une récupération si vous supprimez ou modifiez des données par erreur.

- Activez des fonctions de récupération, de journalisation et de surveillance supplémentaires à la création du cluster. Activez toutes les options disponibles sous Retour en arrière, Performance Insights, Surveillance et Exportations des journaux. Lorsque ces fonctions sont activées, vous pouvez tester l'adéquation de fonctions telles que le retour en arrière, la surveillance améliorée et Performance Insights pour votre charge de travail. Vous pouvez facilement étudier les performances et effectuer une résolution des problèmes au cours de la démonstration de faisabilité.

5. Configurer votre schéma

Sur le cluster Aurora, configurez des bases de données, des tables, des index, des clés étrangères et d'autres objets de schéma pour votre application. Si vous effectuez une transition à partir d'un autre système de base de données compatible MySQL ou PostgreSQL, cette phase s'avèrera simple. Vous utilisez les mêmes ligne de commande et syntaxe SQL, ou d'autres applications clientes qui vous sont familières pour votre moteur de base de données.

Pour exécuter les instructions SQL sur votre cluster, recherchez son point de terminaison de cluster et fournissez cette valeur en tant que paramètre de connexion à votre application cliente. Vous trouverez le point de terminaison de cluster dans l'onglet Connectivity (Connectivité) de la page de détails de votre cluster. Le point de terminaison de cluster est celui intitulé Writer (Rédacteur). L'autre point de terminaison, intitulé Reader (Lecteur), représente une connexion en lecture seule que vous pouvez fournir aux utilisateurs finaux qui exécutent des rapports ou d'autres requêtes en lecture seule. Pour obtenir de l'aide face à des problèmes de connexion à votre cluster, consultez Connexion à un cluster de bases de données Amazon Aurora.

Si vous déplacez votre schéma et vos données à partir d'un système de base de données différent, préparez-vous à apporter à ce stade certaines modifications à votre schéma. Ces modifications de schéma doivent correspondre à la syntaxe SQL et aux capacités disponibles dans Aurora. Vous pouvez exclure certains déclencheurs, colonnes, contraintes ou autres objets de schéma à ce stade. Cela peut s'avérer utile, notamment si ces objets nécessitent des adaptations pour la compatibilité d'Aurora et ne sont pas significatifs pour vos objectifs en matière de démonstration de faisabilité.

Si vous effectuez une migration à partir d'un système de base de données avec un moteur sous-jacent autre que celui d'Aurora, envisagez d'utiliser l'AWS Schema Conversion Tool (AWS SCT) pour simplifier le processus. Pour plus de détails, consultez le Guide de l'utilisateur AWS Schema Conversion Tool. Pour obtenir des détails généraux sur les activités de migration et de portage, consultez le livre blanc AWS Migration de vos bases de données vers Amazon Aurora.

Au cours de cette phase, vous pouvez évaluer la présence ou l'absence d'inefficacités dans votre configuration de schéma, par exemple dans votre stratégie d'indexation ou dans d'autres structures de table, telles que les tables partitionnées. De telles inefficacités peuvent être amplifiées lorsque vous déployez votre application sur un cluster doté de plusieurs instances de base de données et d'une charge de travail importante. Déterminez si vous pouvez optimiser actuellement de tels aspects de performances, ou durant des activités ultérieures, telles qu'un test complet d'évaluation.

6. Importer vos données

Durant la démonstration de faisabilité, vous étudiez les données, ou un échantillon représentatif, provenant de votre système de base de données antérieur. Si possible, configurez au moins certaines données dans chacune de vos tables. Cela vous aide à tester la compatibilité de tous les types de données et fonctions de schéma. Après vous être exercé à l'utilisation des fonctions Aurora de base, effectuez une mise à l'échelle ascendante de la quantité de données. D'ici la fin de votre démonstration de faisabilité, vous devez tester vos outils ETL, les requêtes et la charge de travail globale avec un jeu de données suffisamment grand pour pouvoir en tirer des conclusions précises.

Vous pouvez utiliser plusieurs techniques pour apporter des données de sauvegarde physique ou logique dans Aurora. Pour obtenir des détails, consultez Migration de données vers un cluster de bases de données Amazon Aurora MySQL ou Migration des données vers Amazon Aurora avec compatibilité PostgreSQL selon le moteur de base de données que vous utilisez dans la démonstration de faisabilité.

Faites des expériences avec les technologies et les outils ETL que vous prenez en considération. Déterminez ce qui répond le mieux à vos besoins. Prenez en compte à la fois le débit et la flexibilité. Par exemple, certains outils ETL effectuent un transfert en une seule fois, alors que d'autres impliquent une réplication continue à partir de l'ancien système vers Aurora.

Si vous effectuez une migration à partir d'un système compatible MySQL vers Aurora MySQL, vous pouvez utiliser les outils natifs de transfert de données. La même chose s'applique si vous effectuez une migration à partir d'un système compatible PostgreSQL vers Aurora PostgreSQL. Si vous effectuez une migration à partir d'un système de base de données qui utilise un moteur sous-jacent autre que celui qu'Aurora utilise, vous pouvez faire des expériences avec l'AWS Database Migration Service (AWS DMS). Pour plus d'informations sur AWS DMS, consultez le Guide de l'utilisateur AWS Database Migration Service.

Pour obtenir des détails sur les activités de migration et de portage, consultez le livre blanc AWS Manuel de migration Aurora.

7. Déplacer votre code SQL

Essayer les applications SQL et associées requiert différents niveaux d'effort, selon les cas. En particulier, le niveau d'effort varie selon que vous effectuez un déplacement à partir d'un système compatible MySQL ou PostgreSQL, ou d'un autre type.

- Si vous effectuez un déplacement à partir de RDS for MySQL ou RDS pour PostgreSQL, les modifications SQL sont suffisamment petites pour que vous puissiez essayer d'utiliser le code SQL d'origine avec Aurora et d'incorporer manuellement les modifications nécessaires.
- De même, si vous effectuez un déplacement à partir d'une base de données locale compatible avec MySQL ou PostgreSQL, vous pouvez essayer d'utiliser le code SQL d'origine et d'incorporer manuellement les modifications.
- Si vous partez d'une base de données commerciale différente, les modifications SQL requises sont trop importantes. Dans ce cas, envisagez d'utiliser AWS SCT.

Au cours de cette phase, vous pouvez évaluer la présence ou l'absence d'inefficacités dans votre configuration de schéma, par exemple dans votre stratégie d'indexation ou dans d'autres structures de table, telles que les tables partitionnées. Déterminez si vous pouvez optimiser actuellement de tels aspects de performances, ou durant des activités ultérieures, telles qu'un test complet d'évaluation.

Vous pouvez vérifier la logique de connexion de base de données dans votre application. Pour tirer profit du traitement distribué d'Aurora, vous pouvez avoir besoin d'utiliser des connexions distinctes pour les opérations de lecture et d'écriture, et d'utiliser des sessions relativement courtes pour les opérations de requête. Pour obtenir des informations sur les connexions, consultez 9. Se connecter à Aurora.

Réfléchissez aux concessions et compromis éventuels que vous avez dû faire pour contourner les problèmes dans votre base de données de production. Prévoyez du temps dans votre calendrier de démonstration de faisabilité pour apporter des améliorations à vos requêtes et à la conception de votre schéma. Pour juger si vous pouvez obtenir des victoires faciles en matière de performances, de coût d'exploitation et d'évolutivité, essayez les applications d'origine et modifiées côte à côte sur différents clusters Aurora.

Pour obtenir des détails sur les activités de migration et de portage, consultez le livre blanc AWS Manuel de migration Aurora.

8. Spécifier les paramètres de configuration

Vous pouvez également passer en revue vos paramètres de configuration de base de données dans le cadre de l'exercice de démonstration de faisabilité d'Aurora. Vos paramètres de configuration MySQL ou PostgreSQL sont peut-être déjà réglés pour favoriser les performances et l'évolutivité dans votre environnement actuel. Le sous-système de stockage d'Aurora est adapté et réglé pour un environnement distribué basé sur le cloud avec un sous-système de stockage à grande vitesse. Par conséquent, de nombreux anciens paramètres de moteur de base de données ne s'appliquent pas. Nous vous recommandons de conduire vos expériences initiales avec les paramètres de configuration Aurora par défaut. Réappliquez les paramètres de votre environnement actuel seulement si vous rencontrez des goulots d'étranglement de performances et d'évolutivité. Si vous êtes intéressé, vous pouvez approfondir ce sujet en consultant l'article Introducing the Aurora Storage Engine du blog AWS Database.

Aurora favorise la réutilisation des paramètres de configuration optimaux pour une application particulière ou un cas d'utilisation particulier. Au lieu de modifier un fichier de configuration distinct pour chaque instance de base de données, vous gérez des ensembles de paramètres que vous assignez à des clusters entiers ou à des instances de base de données spécifiques. Par exemple, le paramètre de fuseau horaire s'applique à toutes les instances de base de données du cluster, et vous pouvez ajuster le paramètre de taille du cache de page pour chaque instance de base de données.

Vous commencez avec l'un des ensembles de paramètres par défaut et appliquez les modifications aux seuls paramètres dont vous devez optimiser le réglage. Pour obtenir des détails sur l'utilisation des groupes de paramètres, consultez Paramètres de cluster de bases de données et d'instance de base de données Amazon Aurora. Pour découvrir les paramètres de configuration qui sont applicables ou non aux clusters Aurora, consultez Paramètres de configuration d'Aurora MySQL ou Paramètres Amazon Aurora PostgreSQL. selon votre moteur de base de données.

9. Se connecter à Aurora

Comme vous pouvez le constater en effectuant votre configuration initiale de schéma et de données et en exécutant des exemples de requête, vous pouvez vous connecter à différents points de terminaison dans un cluster Aurora. Le point de terminaison à utiliser varie selon que l'opération correspond à une lecture, telle qu'une instruction `SELECT`, ou à une écriture, telle qu'une instruction `CREATE` ou `INSERT`. Lorsque vous augmentez la charge de travail sur un cluster Aurora et essayez les fonctions Aurora, il est important pour votre application d'affecter chaque opération au point de terminaison approprié.

En utilisant le point de terminaison de cluster pour les opérations d'écriture, vous vous connectez toujours à une instance de base de données dans le cluster qui possède des capacités de lecture/écriture. Par défaut, seule une instance de base de données d'un cluster Aurora possède des capacités de lecture/écriture. Cette instance de base de données est appelée instance principale. Si l'instance principale d'origine devient non disponible, Aurora active un mécanisme de basculement et une autre instance de base de données prend la relève comme instance principale.

De même, en dirigeant les instructions `SELECT` vers le point de terminaison de lecteur, vous répartissez le travail de traitement des requêtes entre les instances de base de données du cluster. Chaque connexion de lecteur est assignée à une instance de base de données différente au moyen de la résolution DNS de type tourniquet (round-robin). La réalisation de la plus grande partie du travail de requête sur les réplicas Aurora de base de données en lecture seule réduit la charge qui s'exerce sur l'instance principale, ce qui libère cette dernière pour traiter les instructions DDL et DML.

L'utilisation de ces points de terminaison réduit la dépendance sur les noms d'hôte codés en dur et aide votre application à récupérer plus rapidement après des échecs d'instance de base de données.

> **Note**
>
> Aurora possède également des points de terminaison personnalisés que vous créez. Ces points de terminaison ne sont généralement pas nécessaires au cours d'une démonstration de faisabilité.

Les réplicas Aurora sont sujets à un retard de réplication, généralement compris entre 10 et 20 millisecondes. Vous pouvez surveiller ce retard de réplication et décider s'il figure dans la plage de vos exigences de cohérence des données. Dans certains cas, vos requêtes de lecture peuvent exiger une forte cohérence de lecture (cohérence de type lecture après écriture). Dans ces cas, vous pouvez continuer à utiliser le point de terminaison de cluster et non pas le point de terminaison de lecteur.

Pour tirer pleinement parti des capacités d'Aurora pour l'exécution parallèle distribuée, vous pouvez être amené à modifier la logique de connexion. Votre objectif est d'éviter d'envoyer toutes les demandes de lecture à l'instance principale. Les réplicas Aurora en lecture seule sont en attente, tous avec les mêmes données, prêts à traiter les instructions `SELECT`. Codez votre logique d'application pour utiliser le point de terminaison approprié pour chaque type d'opération. Suivez ces instructions générales :

- Évitez d'utiliser une seule chaîne de connexion codée en dur pour toutes les sessions de base de données.
- Si possible, placez les opérations d'écriture, telles que les instructions DDL et DML, dans des fonctions, dans le code de votre application cliente. De cette manière, vous pouvez faire en sorte que différents types d'opérations utilisent des connexions spécifiques.
- Élaborez des fonctions distinctes pour les opérations de requête. Aurora affecte chaque nouvelle connexion au point de terminaison de lecteur à un réplica Aurora différent, afin d'équilibrer la charge pour les applications nécessitant beaucoup d'opérations de lecture.
- Pour les opérations impliquant des ensembles de requêtes, fermez et rouvrez la connexion au point de terminaison de lecteur lorsque chaque ensemble de requêtes associées se termine. Utilisez un regroupement de connexions si cette fonction est disponible dans votre pile logicielle. Le fait de diriger les requêtes vers différentes connexions aide Aurora à distribuer la charge de travail de lecture entre les instances de base de données du cluster.

Pour obtenir des informations générales sur la gestion des connexions et les points de terminaison pour Aurora, consultez Connexion à un cluster de bases de données Amazon Aurora. Pour une découverte approfondie de ce sujet, consultez le Manuel d'administrateur de base de données Aurora MySQL – Gestion des connexions.

10. Exécuter votre charge de travail

Une fois que les paramètres de schéma, de données et de configuration sont en place, vous pouvez commencer à vous exercer à utiliser le cluster en exécutant votre charge de travail. Dans la démonstration de faisabilité, utilisez une charge de travail qui reflète les aspects principaux de votre charge de travail de production. Nous vous recommandons de toujours prendre des décisions concernant les performances en utilisant des charges de travail et des tests du monde réel, plutôt que des systèmes de référence synthétiques tels que Sysbench ou TPC-C. Autant que possible, collectez des mesures basées sur vos propres schéma, modèles de requête et volume d'utilisation.

Autant que possible, reproduisez les conditions réelles dans lesquelles l'application s'exécutera. Par exemple, vous exécutez généralement votre code d'application sur des instances Amazon EC2 dans la même région AWS et le même cloud privé virtuel (VPC) que le cluster Aurora. Si votre application de production s'exécute sur plusieurs instances EC2 réparties dans plusieurs zones de disponibilité, configurez votre environnement de démonstration de faisabilité de la même manière. Pour plus d'informations sur les régions AWS, consultez Régions et zones de disponibilité dans le Guide de

l'utilisateur Amazon RDS. Pour en savoir plus sur le service Amazon VPC, consultez Qu'est-ce qu'Amazon VPC ? dans le Guide de l'utilisateur Amazon VPC.

Une fois que vous avez vérifié que les fonctions de base de votre application fonctionnent et que vous pouvez accéder aux données via Aurora, vous pouvez étudier les aspects du cluster Aurora. Certaines fonctions que vous pouvez essayer mettent en jeu des connexions simultanées à l'équilibrage de la charge, à des transactions simultanées et à la réplication automatique.

À ce stade, les mécanismes de transfert de données doivent être familiers et tels que vous puissiez exécuter des tests avec une plus grande proportion d'échantillons de données.

Cette phase permet de voir les effets du changement des paramètres de configuration, tels que les limites de mémoire et les limites de connexion. Réexaminez les procédures que vous avez explorées dans 8. Spécifier les paramètres de configuration.

Vous pouvez également effectuer des essais avec des mécanismes tels que la création et la restauration d'instantanés. Par exemple, vous pouvez créer des clusters avec différentes classes d'instance AWS, plusieurs réplicas AWS, etc. Ensuite, dans chaque cluster, vous pouvez restaurer le même instantané contenant votre schéma et toutes vos données. Pour découvrir les détails de ce cycle, consultez Création d'un instantané de cluster de bases de données et Restauration à partir d'un instantané de cluster de bases de données.

11. Mesurer les performances

Dans ce domaine, les bonnes pratiques sont conçues pour garantir que tous les processus et outils appropriés soient configurés de manière à isoler rapidement les comportements anormaux au cours des opérations mettant en jeu les charges de travail. Elles sont également élaborées pour vous assurer de pouvoir identifier de façon fiable toutes les causes applicables.

Vous pouvez toujours voir l'état actuel de votre cluster ou examiner les tendances au fil du temps en affichant l'onglet Surveillance. Cet onglet est disponible à partir de la page de détails de la console pour chaque cluster ou instance de base de données Aurora. Il affiche les métriques provenant du service de surveillance Amazon CloudWatch sous la forme de graphiques. Vous pouvez filtrer les métriques par leur nom, par l'instance de base de données et par période.

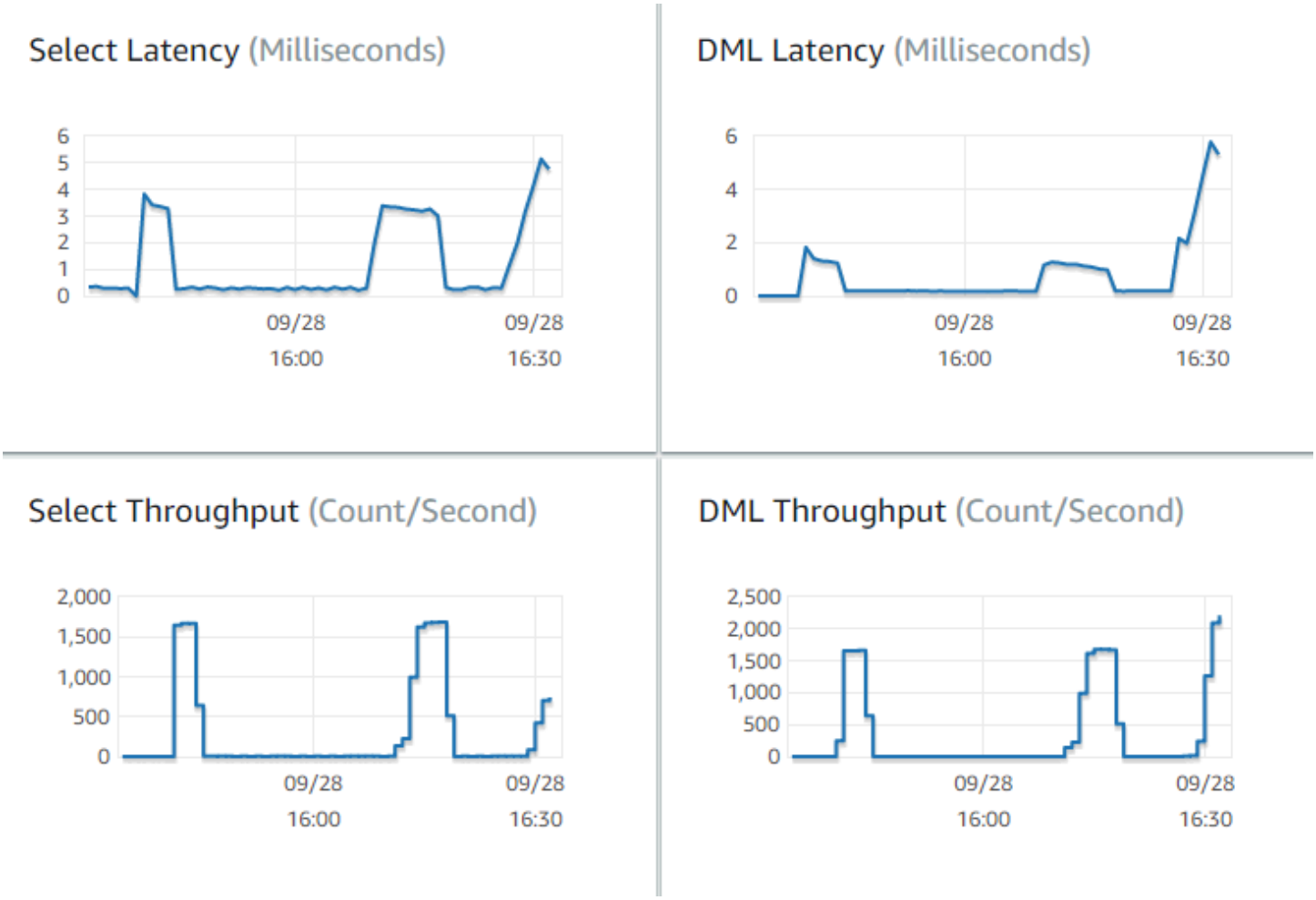
Pour disposer de plus de choix dans l'onglet Surveillance, activez les options de surveillance améliorée et Performance Insights dans les paramètres du cluster. Vous pouvez également activer ultérieurement ces choix si vous ne l'avez pas fait lors de la configuration du cluster.

Pour mesurer les performances, vous pouvez principalement vous appuyer sur les graphiques qui montrent les activités pour le cluster Aurora tout entier. Vous pouvez vérifier si les réplicas Aurora ont une charge et des temps de réponse similaires. Vous pouvez également voir comment le travail est divisé entre l'instance principale de lecture/écriture et les réplicas Aurora en lecture seule. Dans le cas d'un déséquilibre entre les instances de base de données ou d'un problème affectant seulement une instance de base de données, vous pouvez examiner l'onglet Surveillance pour cette instance spécifique.

Une fois que l'environnement et la charge de travail réelle ont été configurés pour émuler votre application de production, vous pouvez mesurer la façon dont Aurora fonctionne. Les questions les plus importantes auxquelles il convient de répondre sont les suivantes :

- Combien de requêtes par seconde Aurora traite-t-il ? Vous pouvez examiner les métriques de débit (Throughput) pour voir les chiffres pour divers types d'opérations.
- Combien de temps faut-il en moyenne à Aurora pour traiter une requête donnée ? Vous pouvez examiner les métriques de latence (Latency) pour voir les chiffres pour divers types d'opérations.

Pour consulter les métriques de débit et de latence, consultez l'onglet Surveillance d'un cluster Aurora donné dans la console Amazon RDS. La capture d'écran suivante montre un exemple des métriques Sélectionnez Latence, Latence DML, Sélectionner le débit et Débit DML dans l'onglet Surveillance.



Si vous le pouvez, établissez des valeurs de référence pour ces métriques dans votre environnement actuel. Si ce n'est pas possible, établissez une référence sur le cluster Aurora en exécutant une charge de travail équivalente à votre application de production. Par exemple, exécutez votre charge de travail Aurora avec un nombre similaire d'utilisateurs et de requêtes simultanés. Ensuite, observez la manière dont les valeurs changent lorsque vous essayez différents paramètres de configuration, classes d'instance, tailles de cluster, etc.

Si les chiffres de débit sont inférieurs à ce que vous attendiez, poursuivez vos investigations pour déterminer les facteurs affectant les performances de base de données pour votre charge de travail. De même, si les chiffres de latence sont supérieurs à ce que vous attendiez, poursuivez vos investigations. Pour cela, surveillez les métriques secondaires du serveur de base de données (UC, mémoire, etc.). Vous pouvez voir si les instances de base de données sont proches de leurs limites. Vous pouvez également voir les capacités supplémentaires dont vos instances de base de données disposent pour traiter plus de requêtes simultanées, des requêtes portant sur des tables plus grandes, etc.

> **Tip**
>
> Pour détecter les valeurs de métriques qui sortent des plages de valeurs attendues, configurez des alarmes CloudWatch.

Lorsque vous évaluez la capacité et la taille idéales d'un cluster Aurora, vous pouvez trouver la configuration qui permet d'atteindre des performances d'application de pointe sans sur-approvisionnement de ressources. Un facteur important est de trouver la taille appropriée pour les instances de base de données du cluster Aurora. Commencez par sélectionner une taille d'instance qui possède une capacité d'UC et de mémoire similaire à celle de votre environnement de production actuel. Collectez les chiffres de débit et de latence pour la charge de travail à cette taille d'instance. Ensuite, mettez à l'échelle l'instance jusqu'à la taille supérieure suivante. Observez si les chiffres de débit et de latence s'améliorent. Réduisez également la taille de l'instance et observez si les chiffres de latence et de débit restent les mêmes. Votre objectif est d'obtenir le plus haut débit avec la plus basse latence sur la plus petite instance possible.

> **Tip**
>
> Dimensionnez vos clusters Aurora et les instances de base de données associées avec une capacité existante suffisante pour traiter les pics de trafic imprévisibles et soudains. Pour les bases de données d'importance capitale, laissez au moins 20 % de capacité d'UC et de mémoire non utilisée.

Exécuter des tests de performances suffisamment longs pour mesurer les performances de base de données dans un état stable et à chaud. Vous pouvez avoir besoin d'exécuter la charge de travail pendant de nombreuses minutes ou même quelques heures avant d'atteindre cet état stable. Il est normal d'avoir une certaine variation en début d'exécution. Cette variation se produit, car chaque réplica Aurora fait chauffer ses caches sur la base des requêtes `SELECT` qu'il traite.

Aurora fonctionne le mieux avec des charges de travail transactionnelles impliquant plusieurs requêtes et utilisateurs simultanés. Pour vérifier que vous utilisez une charge suffisante pour obtenir des performances optimales, effectuez des évaluations qui utilisent le multithreading ou exécutez simultanément plusieurs instances des tests de performances. Mesurez les performances avec des centaines ou même des milliers de threads client simultanés. Simulez le nombre de threads simultanés que vous prévoyez dans votre environnement de production. Vous pouvez également

effectuer des tests de contrainte supplémentaires avec plus de threads pour mesurer l'évolutivité d'Aurora.

12. Étudier la haute disponibilité d'Aurora

Un grand nombre des fonctions Aurora principales impliquent la haute disponibilité. Ces fonctions incluent la réplication automatique, le basculement automatique, les sauvegardes automatiques avec restauration à un instant dans le passé, ainsi que la capacité à ajouter des instances de base de données au cluster. La sécurité et la fiabilité qui émanent de fonctions telles que celles-ci sont importantes pour les applications d'importance capitale.

L'évaluation de ces fonctions requiert un certain état d'esprit. Dans les activités précédentes, telles que le mesurage des performances, vous observez les performances du système quand tout fonctionne correctement. Les tests de haute disponibilité exigent que vous pensiez dans le détail le comportement pour le pire scénario. Vous devez prendre en compte différents types d'échecs, même si de telles conditions sont rares. Vous pouvez introduire intentionnellement des problèmes pour vérifier la capacité du système à récupérer rapidement et correctement.

> **Tip**
>
> Pour une preuve de concept, configurez toutes les instances de base de données d'un cluster Aurora avec la même classe d'instance AWS. Cela permet de tester les fonctions de disponibilité d'Aurora sans changements majeurs des performances et de l'évolutivité lorsque vous mettez hors connexion les instances de base de données pour simuler des défaillances.

Nous vous recommandons d'utiliser au moins deux instances dans chaque cluster Aurora.
Les instances de base de données d'un cluster Aurora peuvent couvrir jusqu'à trois zones de disponibilité. Localisez chacune des deux ou trois premières instances de base de données dans une zone de disponibilité différente. Lorsque vous commencez à utiliser de plus grands clusters, répartissez vos instances de base de données dans toutes les zones de disponibilité de votre région AWS. Cela augmente la capacité de tolérance aux pannes. Même si un problème affecte une zone de disponibilité complète, Aurora peut basculer vers une instance de base de données située dans une autre zone de disponibilité. Si vous utilisez un cluster doté de plus de trois instances, distribuez les instances de base de données aussi uniformément que possible sur les trois zones de disponibilité.

> **Tip**
>
> Le stockage pour un cluster Aurora est indépendant des instances de base de données. Le stockage pour chaque cluster Aurora couvre toujours trois zones de disponibilité.
> Lorsque vous testez des fonctions à haute disponibilité, utilisez toujours des instances de base de données d'une capacité identique dans votre cluster test. Cela permet d'éviter les modifications imprévisibles de performances, de latence, etc., chaque fois qu'une instance de base de données prend la relève d'une autre.

Pour découvrir comment simuler des conditions de défaillance pour tester des fonctions à haute disponibilité, consultez Test d'Amazon Aurora MySQL à l'aide de requêtes d'injection d'erreurs.

Dans le cadre de votre exercice de démonstration de faisabilité, l'un des objectifs est de trouver le nombre idéal d'instances de base de données et la classe d'instance optimale pour ces instances de base de données. Cela requiert d'équilibrer les exigences de performances et de haute disponibilité.

Pour Aurora, plus vous avez d'instances de base de données dans un cluster, plus cela profite à la haute disponibilité. L'augmentation du nombre d'instances de base de données améliore également la capacité de mise à l'échelle des applications nécessitant beaucoup d'opérations de lecture. Aurora peut distribuer plusieurs connexions pour les requêtes `SELECT` entre les réplicas Aurora en lecture seule.

D'un autre côté, la limitation du nombre d'instances de base de données réduit le trafic de réplication à partir du nœud principal. Le trafic de réplication consomme de la bande passante réseau, ce qui constitue un autre aspect des performances et de l'évolutivité globales. Ainsi, pour les applications OLTP qui demandent beaucoup d'opérations d'écriture, privilégiez un plus petit nombre de grandes instances de base de données à un grand nombre de petites instances de base de données.

Dans un cluster Aurora standard, une seule instance de base de données (l'instance principale) traite toutes les instructions DDL et DML. Les autres instances de base de données (les réplicas Aurora) traitent uniquement les instructions `SELECT`. Bien que les instances de base de données n'exécutent pas exactement la même quantité de travail, nous vous recommandons d'utiliser la même classe d'instance pour toutes les instances de base de données du cluster. De cette manière, si une défaillance survient et qu'Aurora promeut l'une des instances de base de données en lecture seule comme nouvelle instance principale, cette instance principale a la même capacité qu'avant.

Si vous avez besoin d'utiliser des instances de base de données de différentes capacités dans un même cluster, configurez des niveaux de basculement pour les instances de base de données.

Ces niveaux déterminent l'ordre dans lequel les réplicas Aurora sont promus par le mécanisme de basculement. Placez les instances de base de données beaucoup plus grandes ou plus petites que les autres à un niveau de basculement inférieur. Cela garantit qu'ils seront choisis en dernier pour une promotion.

Exercez-vous à utiliser les fonctions de récupération de données d'Aurora, telles que la restauration automatique à un instant dans le passé, la restauration et les instantanés manuels, ainsi que le retour en arrière du cluster. Le cas échéant, copiez des instantanés dans d'autres régions AWS et effectuez la restauration dans d'autres régions AWS pour imiter des scénarios de reprise après sinistre.

Vérifiez les exigences de votre organisation en ce qui concerne l'objectif de délai de reprise (RTO), l'objectif de point de reprise (RPO) et la redondance géographique. La plupart des organisations groupent ces éléments dans la catégorie élargie de reprise après sinistre. Évaluez les fonctions de haute disponibilité d'Aurora décrites dans cette section dans le contexte de votre processus de reprise après sinistre pour vous assurer que vos exigences RTO et RPO sont satisfaites.

13. Suite des opérations

À la fin d'un processus de démonstration de faisabilité réussi, vous confirmez qu'Aurora est une solution qui vous convient sur la base de la charge de travail anticipée. Tout au long du processus précédent, vous avez vérifié comment Aurora fonctionne dans un environnement opérationnel réaliste et avez mesuré cela sur la base de vos critères de réussite.

Une fois que votre environnement de base de données est opérationnel avec Aurora, vous pouvez passer à des étapes d'évaluation plus détaillées, qui mènent à votre migration finale et au déploiement en production. Selon votre situation, ces autres étapes peuvent être incluses ou non dans le processus de démonstration de faisabilité. Pour obtenir des détails sur les activités de migration et de portage, consultez le livre blanc AWS Manuel de migration Aurora.

Dans une autre étape ultérieure, prenez en considération les configurations de sécurité pertinentes pour votre charge de travail et conçues pour satisfaire vos exigences de sécurité dans un environnement de production. Planifiez les contrôles à mettre en place pour protéger l'accès aux informations d'identification des utilisateurs principaux du cluster Aurora. Définissez les rôles et responsabilités des utilisateurs de base de données pour contrôler l'accès aux données stockées dans le cluster Aurora. Prenez en compte les exigences d'accès aux bases de données pour les applications, les scripts et les outils ou services tiers. Explorez les fonctions et services AWS, tels que AWS Secrets Manager et l'authentification Gestion des identités et des accès AWS (IAM).

À ce stade, vous devez comprendre les procédures et les bonnes pratiques à utiliser pour effectuer des tests d'évaluation avec Aurora. Vous pouvez constater qu'il vous faut effectuer un réglage supplémentaire des performances. Pour en savoir plus, consultez Gestion des performances et dimensionnement des clusters de bases de données Aurora, Améliorations des performances Amazon Aurora MySQL, Performance et mise à l'échelle d'Amazon Aurora PostgreSQL et Surveillance de la charge de la base de données avec Performance Insights sur . Si vous effectuez un réglage supplémentaire, veillez à bien connaître les métriques que vous avez rassemblées au cours de la démonstration de faisabilité. Dans une étape ultérieure, vous pouvez créer des clusters en faisant des choix différents de paramètres de configuration, de moteur de base de données et de version de base de données. Vous pouvez également créer des types spécialisés de clusters Aurora pour répondre aux besoins de cas d'utilisation spécifiques.

Par exemple, vous pouvez explorer des clusters Aurora compatibles avec les requêtes parallèles pour les applications de traitement hybride d'analyse/de transaction (HTAP). Si une large distribution géographique est essentielle pour la reprise après sinistre ou pour réduire au maximum la latence, vous pouvez explorer les bases de données globales Aurora. Si votre charge de travail est intermittente ou que vous utilisez Aurora dans un scénario de développement/test, vous pouvez explorer les clusters Aurora Serverless.

Vos clusters de production peuvent également avoir besoin de traiter des volumes élevés de connexions entrantes. Pour apprendre ces techniques, consultez le livre blanc AWS Manuel d'administrateur de base de données Aurora MySQL – Gestion des connexions.

Si, après la preuve de concept, vous décidez que votre cas d'utilisation n'est pas adapté pour Aurora, envisagez d'utiliser les autres services AWS suivants :

- Pour les cas d'utilisation purement analytiques, les charges de travail tirent profit d'un format de stockage en colonnes et d'autres fonctionnalités mieux adaptées aux charges de travail OLAP. Les services AWS qui répondent à de tels cas d'utilisation sont les suivants :
  - Amazon Redshift
  - Amazon EMR
  - Amazon Athena
- De nombreuses charges de travail profitent d'une combinaison d'Aurora et d'un ou de plusieurs de ces services. Vous pouvez déplacer les données entre ces services en utilisant les solutions suivantes :
  - AWS Glue
  - AWS DMS

- Importation à partir d’Amazon S3, comme décrit dans le Guide de l’utilisateur Amazon Aurora
- Exportation vers Amazon S3, comme décrit dans le Guide de l’utilisateur Amazon Aurora
- De nombreux autres outils ETL courants

Sécurité dans Amazon Aurora

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de conformitéAWS. Pour découvrir les programmes de conformité qui s'appliquent à Amazon Aurora (Aurora), consultez Services AWS concernés par le programme de conformité.
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre organisation, et la législation et la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation de Amazon Aurora. Les rubriques suivantes vous montrent comment configurer Amazon Aurora pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui vous aident à surveiller et à sécuriser vos ressources Amazon Aurora.

Vous pouvez gérer l'accès à vos ressources Amazon Aurora et à vos bases de données sur un cluster de bases de données. La méthode que vous utilisez pour gérer l'accès dépend du type de tâche que l'utilisateur doit effectuer avec Amazon Aurora :

- Exécutez votre cluster de bases de données dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC pour disposer du meilleur contrôle d'accès réseau possible. Pour plus d'informations sur la création d'un cluster de bases de données dans un VPC, consultez Amazon VPC et Amazon Aurora.
- Utilisez des politiques Gestion des identités et des accès AWS (IAM) pour attribuer des autorisations qui déterminent qui est autorisé à gérer les ressources Amazon Aurora. Par exemple, vous pouvez utiliser IAM pour déterminer qui est autorisé à créer, décrire, modifier et supprimer

des clusters de bases de données, attribuer des balises à des ressources ou modifier des groupes de sécurité.

Pour passer en revue des exemples de politiques IAM, consultez Exemples de politiques basées sur l'identité pour Amazon Aurora.

- Utilisez des groupes de sécurité pour contrôler les adresses IP ou les EC2 instances Amazon qui peuvent se connecter à vos bases de données sur un cluster d' de base de données. Quand vous créez un cluster de bases de données pour la première fois, son pare-feu empêche tout accès aux bases de données sauf via les règles spécifiées par un groupe de sécurité associé.
- Utilisez les connexions Secure Socket Layer (SSL) ou du protocole TLS (Transport Layer Security) avec des clusters de bases de données exécutant Aurora MySQL ou Aurora PostgreSQL. Pour plus d'informations sur l'utilisation SSL/TLS avec un cluster de base de données, consultezUtilisation SSL/TLS pour chiffrer une connexion à une de clusters.
- Utilisez le chiffrement Amazon Aurora pour sécuriser vos clusters de bases de données et instantanés au repos. Le chiffrement Amazon Aurora utilise l'algorithme de chiffrement AES-256 standard pour chiffrer vos données sur le serveur qui héberge votre cluster de bases de données. Pour plus d'informations, consultez Chiffrement des ressources Amazon Aurora.
- Utilisez les fonctionnalités de sécurité de votre moteur de base de données pour contrôler qui peut se connecter aux bases de données sur un cluster de bases de données. Ces fonctions agissent comme si la base de données se trouvait sur votre réseau local.

  Pour en savoir plus sur la sécurité avec Aurora MySQL, consultez Sécurité avec Amazon Aurora MySQL. Pour en savoir plus sur la sécurité avec Aurora PostgreSQL, consultez Sécurité avec Amazon Aurora PostgreSQL.

Aurora fait partie du service de base de données géré Amazon Relational Database Service (Amazon RDS). Amazon RDS est un service web qui facilite la configuration, l'exploitation et la mise à l'échelle d'une base de données relationnelle dans le cloud. Si vous connaissez déjà Amazon RDS, consultez le Guide de l'utilisateur Amazon RDS.

Aurora comprend un sous-système de stockage très performant. Ses moteurs de bases de données compatibles avec MySQL et PostgreSQL sont personnalisés afin de tirer parti de ce stockage distribué et rapide. Aurora automatise et standardise également le clustering et la réplication des bases de données. Ces aspects figurent généralement parmi ceux qui représentent un défi dans le cadre de la configuration et de l'administration des bases de données.

Pour Amazon RDS et Aurora, vous pouvez accéder à l'API RDS par programmation, et vous pouvez l'utiliser AWS CLI pour accéder à l'API RDS de manière interactive. Certaines opérations d'API RDS et commandes d' AWS CLI s'appliquent à Amazon RDS et à Aurora, alors que d'autres s'appliquent soit à Amazon RDS, soit à Aurora. Pour obtenir des informations sur les opérations d'API RDS, consultez Référence d'API Amazon RDS. Pour plus d'informations à ce sujet AWS CLI, consultez la AWS Command Line Interface référence relative à Amazon RDS.

> **Note**
>
> Vous devez uniquement configurer la sécurité de vos cas d'utilisation. Vous n'avez pas à configurer l'accès de sécurité pour les processus gérés par Amazon Aurora. Il s'agit notamment de la création de sauvegardes, du basculement automatique et d'autres processus.

Pour plus d'informations sur la gestion de l'accès aux ressources Amazon Aurora et à vos bases de données sur une un cluster de bases de données, consultez les rubriques suivantes.

Rubriques

- Authentification de base de données avec Amazon Aurora
- Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager
- Protection des données dans Amazon RDS
- Identity and Access Management pour Amazon Aurora
- Journalisation et surveillance dans Amazon Aurora
- Validation de la conformité pour Amazon Aurora
- Résilience dans Amazon Aurora
- Sécurité de l'infrastructure dans Amazon Aurora
- API Amazon RDS et points de terminaison d'un VPC d'interface (AWS PrivateLink)
- Bonnes pratiques de sécurité pour Amazon Aurora
- Contrôle d'accès par groupe de sécurité
- Privilèges du compte utilisateur principal
- Utilisation des rôles liés à un service pour Amazon Aurora
- Amazon VPC et Amazon Aurora

Authentification de base de données avec Amazon Aurora

Amazon Aurora prend en charge plusieurs façons d'authentifier les utilisateurs de base de données.

L'authentification par mot de passe est disponible par défaut pour tous les clusters de bases de données. Pour Aurora MySQL et Aurora PostgreSQL, vous pouvez également ajouter l'authentification de base de données IAM ou Kerberos ou les deux pour le même cluster de bases de données.

L'authentification par mot de passe, Kerberos et IAM utilisent différentes méthodes d'authentification auprès de la base de données. Par conséquent, un utilisateur spécifique peut se connecter à une base de données en utilisant une seule méthode d'authentification.

Pour PostgreSQL, utilisez un seul des paramètres de rôle suivants pour un utilisateur d'une base de données spécifique :

- Pour utiliser l'authentification de base de données IAM, affectez le rôle `rds_iam` à l'utilisateur.
- Pour utiliser l'authentification Kerberos, affectez le rôle `rds_ad` à l'utilisateur.
- Pour utiliser l'authentification par mot de passe, n'affectez pas les rôles `rds_iam` ou `rds_ad` à l'utilisateur.

N'affectez pas à la fois les rôles `rds_iam` et `rds_ad` à un utilisateur d'une base de données PostgreSQL, directement ou indirectement par l'intermédiaire d'un accès accordé imbriqué. Si le rôle `rds_iam` est ajouté à l'utilisateur principal, l'authentification IAM a priorité sur l'authentification par mot de passe, de sorte que l'utilisateur principal doit se connecter en tant qu'utilisateur IAM.

> **Important**
>
> Nous vous recommandons vivement de ne pas avoir recours au rôle d'utilisateur principal directement dans vos applications. Au lieu de cela, respectez la bonne pratique qui consiste à avoir recours à un utilisateur de base de données doté des privilèges minimum requis pour votre application.

Rubriques

- Authentification Kerberos

Authentification par mot de passe

Avec l'authentification par mot de passe, votre base de données se charge de toute l'administration des comptes utilisateurs. Vous créez des utilisateurs avec des instructions SQL telles que `CREATE USER`, avec la clause appropriée requise par le moteur de base de données pour spécifier des mots de passe. Par exemple, dans MySQL, l'instruction est `CREATE USER` *`nom`* `IDENTIFIED BY` *`mot de passe`*, tandis que dans PostgreSQL, l'instruction est `CREATE USER` *`nom`* `WITH PASSWORD` *`mot de passe`*.

Avec l'authentification par mot de passe, votre base de données contrôle et authentifie les comptes d'utilisateurs. Si un moteur de base de données dispose de fonctionnalités de gestion de mot de passe solides, il peut améliorer la sécurité. L'authentification de base de données peut être plus facile à administrer en utilisant l'authentification par mot de passe lorsque vous avez de petites communautés d'utilisateurs. Étant donné que des mots de passe en texte clair sont générés dans ce cas, l'intégration avec AWS Secrets Manager peut améliorer la sécurité.

Pour plus d'informations sur l'utilisation de Secrets Manager avec Amazon Aurora, consultez Création d'un secret de base et Rotation de secrets pour les bases de données Amazon RDS prises en charge dans le Guide de l'utilisateur AWS Secrets Manager. Pour plus d'informations sur la récupération par programme de vos secrets dans vos applications personnalisées, consultez Récupération de la valeur de secret dans le Guide de l'utilisateur AWS Secrets Manager.

Authentification de base de données IAM

Vous pouvez vous authentifier auprès de votre cluster de bases de données à l'aide de l'authentification de base de données Gestion des identités et des accès AWS (IAM). Grâce à cette méthode d'authentification, vous n'avez plus besoin de mot de passe pour vous connecter à un cluster de bases de données. En revanche, un jeton d'authentification est nécessaire.

Pour plus d'informations sur l'authentification de base de données IAM, y compris sur la disponibilité de moteurs de base de données spécifiques, consultez Authentification de base de données IAM.

Authentification Kerberos

Amazon Aurora prend en charge l'authentification externe des utilisateurs de bases de données avec Kerberos et Microsoft Active Directory. Kerberos est un protocole d'authentification réseau qui utilise

les tickets et la cryptographie de clé symétrique pour vous éviter d'acheminer vos mots de passe via le réseau. Intégré dans Active Directory, Kerberos est conçu pour authentifier les utilisateurs sur les ressources réseau, par exemple les bases de données.

La prise en charge de Kerberos et Active Directory par Amazon Aurora procure les avantages d'une authentification unique et centralisée des utilisateurs de bases de données. Vous pouvez conserver vos informations d'identification utilisateur dans Active Directory. Active Directory vous offre un endroit centralisé de stockage et de gestion des informations d'identification pour plusieurs d'instances de base de données.

Pour utiliser les informations d'identification de votre Active Directory autogéré, vous devez configurer une relation de confiance avec l'Directory Service de Microsoft Active Directory auquel le d'instance de base de données est joint.

Aurora PostgreSQL et Aurora MySQL prennent en charge les relations de confiance unidirectionnelles et bidirectionnelles avec une authentification à l'échelle de la forêt ou une authentification sélective.

Dans certains scénarios, vous pouvez configurer l'authentification Kerberos via une relation de confiance externe. Cela nécessite que votre Active Directory autogéré dispose de paramètres supplémentaires. Cela inclut, mais sans s'y limiter, l'ordre de recherche Kerberos Forest.

Aurora prend en charge l'authentification Kerberos pour les clusters de bases de données Aurora MySQL et Aurora PostgreSQL. Pour plus d'informations, consultez Utilisation de l'authentification Kerberos pour Aurora MySQL et Utilisation de l'authentification Kerberos avec Aurora PostgreSQL.

Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager

Amazon Aurora s'intègre à Secrets Manager pour gérer les mots de passe d'utilisateur principal de vos clusters de bases de données.

Rubriques

Disponibilité des régions et des versions

La disponibilité et la prise en charge des fonctionnalités varient selon les versions spécifiques de chaque moteur de base de données, et selon les Régions AWS. Pour plus d'informations sur la disponibilité des versions et des régions avec l'intégration de Secrets Manager avec Amazon Aurora, consultez Régions et moteurs de base de données Aurora pris en charge pour l'intégration de Secrets Manager.

Limites de l'intégration de Secrets Manager avec Amazon Aurora

La gestion des mots de passe d'utilisateur principal à l'aide de Secrets Manager n'est pas prise en charge pour les fonctionnalités suivantes :

- Déploiements Amazon RDS Blue/Green
- Clusters de bases de données qui font partie d'une base de données globale Aurora
- Clusters DB Aurora Serverless v1

- Réplicas en lecture entre Régions
- Réplication externe du journal binaire

Présentation de la gestion des mots de passe des utilisateurs principaux avec AWS Secrets Manager

Vous pouvez AWS Secrets Manager ainsi remplacer les informations d'identification codées en dur dans votre code, y compris les mots de passe de base de données, par un appel d'API à Secrets Manager pour récupérer le secret par programmation. Pour plus d'informations sur Secrets Manager, consultez le Guide de l'utilisateur AWS Secrets Manager.

Lorsque vous stockez des secrets de base de données dans Secrets Manager, des frais Compte AWS vous sont facturés. Pour plus d'informations sur la tarification, consultez la section AWS Secrets ManagerTarification.

Vous pouvez spécifier qu'Aurora doit gérer le mot de passe d'utilisateur principal dans Secrets Manager pour un cluster de bases de données Amazon Aurora quand vous effectuez l'une des opérations suivantes :

- Création d'un cluster de bases de données
- Modification d'un cluster de bases de données
- Restauration d'un cluster de bases de données à partir d'Amazon S3 (Aurora MySQL uniquement)

Quand vous spécifiez qu'Aurora doit gérer le mot de passe d'utilisateur principal dans Secrets Manager, Aurora génère le mot de passe et le stocke dans Secrets Manager. Vous pouvez interagir directement avec le secret pour récupérer les informations d'identification de l'utilisateur principal. Vous pouvez également spécifier une clé gérée par le client pour chiffrer le secret, ou utiliser la clé KMS fournie par Secrets Manager.

Aurora gère les paramètres du secret et effectue la rotation du secret tous les sept jours, par défaut. Vous pouvez modifier certains paramètres, tels que la planification de la rotation. Si vous supprimez un cluster de bases de données qui gère un secret dans Secrets Manager, le secret et les métadonnées associées sont également supprimés.

Pour vous connecter à un cluster de base de données avec les informations d'identification contenues dans un secret, vous pouvez récupérer le secret à partir de Secrets Manager. Pour plus d'informations, voir Extraire des secrets depuis une base de données SQL AWS Secrets Manager

et Se connecter à une base de données SQL avec des informations d'identification inscrites dans un AWS Secrets Manager secret dans le Guide de AWS Secrets Manager l'utilisateur.

Avantages de la gestion des mots de passe d’utilisateur principal avec Secrets Manager

La gestion des mots de passe d’utilisateur principal Aurora avec Secrets Manager présente les avantages suivants :

- Aurora génère automatiquement des informations d’identification de base de données.
- Aurora stocke et gère automatiquement les informations d'identification de la base de données dansAWS Secrets Manager.
- Aurora effectue une rotation régulière des informations d’identification de base de données, sans exiger de modifications d’application.
- Secrets Manager sécurise les informations d’identification de base de données contre tout accès humain et tout affichage en texte brut.
- Secrets Manager permet de récupérer les informations d’identification de base de données dans des secrets pour les connexions à une base de données.
- Secrets Manager permet un contrôle précis de l’accès aux informations d’identification de base de données dans des secrets à l’aide d’IAM.
- Vous pouvez éventuellement séparer le chiffrement d’une base de données du chiffrement des informations d’identification à l’aide de clés KMS différentes.
- Vous pouvez éliminer la gestion et la rotation manuelles des informations d’identification de base de données.
- Vous pouvez facilement surveiller les informations d'identification de la base AWS CloudTrail de données avec Amazon CloudWatch.

Pour en savoir plus sur les avantages de Secrets Manager, consultez le Guide de l’utilisateur AWS Secrets Manager.

Autorisations requises pour l’intégration de Secrets Manager

Les utilisateurs doivent disposer des autorisations requises pour effectuer des opérations liées à l’intégration de Secrets Manager. Vous pouvez créer des politiques IAM qui accordent des autorisations pour effectuer des opérations API spécifiques sur les ressources spécifiées dont ils

ont besoin. Vous pouvez ensuite attacher ces politiques aux jeux d'autorisations ou rôles IAM qui requièrent ces autorisations. Pour plus d'informations, consultez Identity and Access Management pour Amazon Aurora.

Pour les opérations de création, de modification ou de restauration, l'utilisateur qui spécifie qu'Aurora doit gérer le mot de passe d'utilisateur principal dans Secrets Manager doit avoir les autorisations nécessaires pour effectuer les opérations suivantes :

- `kms:DescribeKey`
- `secretsmanager:CreateSecret`
- `secretsmanager:TagResource`

L'autorisation `kms:DescribeKey` est requise pour accéder à votre clé gérée par le client pour `MasterUserSecretKmsKeyId` et pour décrire `aws/secretsmanager`.

Pour les opérations de création, de modification ou de restauration, l'utilisateur qui spécifie la clé gérée par le client pour chiffrer le secret dans Secrets Manager doit avoir les autorisations nécessaires pour effectuer les opérations suivantes :

- `kms:Decrypt`
- `kms:GenerateDataKey`
- `kms:CreateGrant`

Pour les opérations de modification, l'utilisateur qui effectue la rotation du mot de passe d'utilisateur principal dans Secrets Manager doit être autorisé à effectuer l'opération suivante :

- `secretsmanager:RotateSecret`

Application de la gestion du mot de passe de l'utilisateur principal par dans AWS Secrets Manager

Vous pouvez utiliser des clés de condition IAM pour mettre en œuvre la gestion par Aurora du mot de passe d'utilisateur principal dans AWS Secrets Manager. La politique suivante n'autorise pas les utilisateurs à créer ni à restaurer des instances de base de données ou des clusters de bases de données , à moins que le mot de passe d'utilisateur principal soit géré par Aurora dans Secrets Manager.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": ["rds:CreateDBInstance", "rds:CreateDBCluster",
 "rds:RestoreDBInstanceFromS3", "rds:RestoreDBClusterFromS3"],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "rds:ManageMasterUserPassword": false
                }
            }
        }
    ]
}
```

> **Note**
>
> Cette politique impose la gestion des mots de passe dès AWS Secrets Manager leur création. Toutefois, vous pouvez toujours désactiver l'intégration de Secrets Manager et définir manuellement un mot de passe principal en modifiant le cluster.
> Pour éviter cela, incluez `rds:ModifyDBInstance`, `rds:ModifyDBCluster` dans le bloc action de la politique. Sachez que cela empêche l'utilisateur d'appliquer d'autres modifications aux clusters existant(e)s n'ayant pas l'intégration de Secrets Manager activée.

Pour plus d'informations sur l'utilisation de clés de condition dans les politiques IAM, consultez Clés de condition de politique pour Aurora et Exemples de politiques : Utilisation des clés de condition.

Gestion du mot de passe d'utilisateur principal pour un cluster de bases de données avec Secrets Manager

Vous pouvez configurer la gestion Aurora du mot de passe d'utilisateur principal dans Secrets Manager lorsque vous effectuez les actions suivantes :

- Création d’un cluster de bases de données Amazon Aurora
- Modification d’un cluster de bases de données Amazon Aurora
- Migration des données d'une base de données MySQL externe vers un cluster de bases de données Amazon Aurora MySQL

Vous pouvez utiliser la console RDSAWS CLI, ou l'API RDS pour effectuer ces actions.

Console

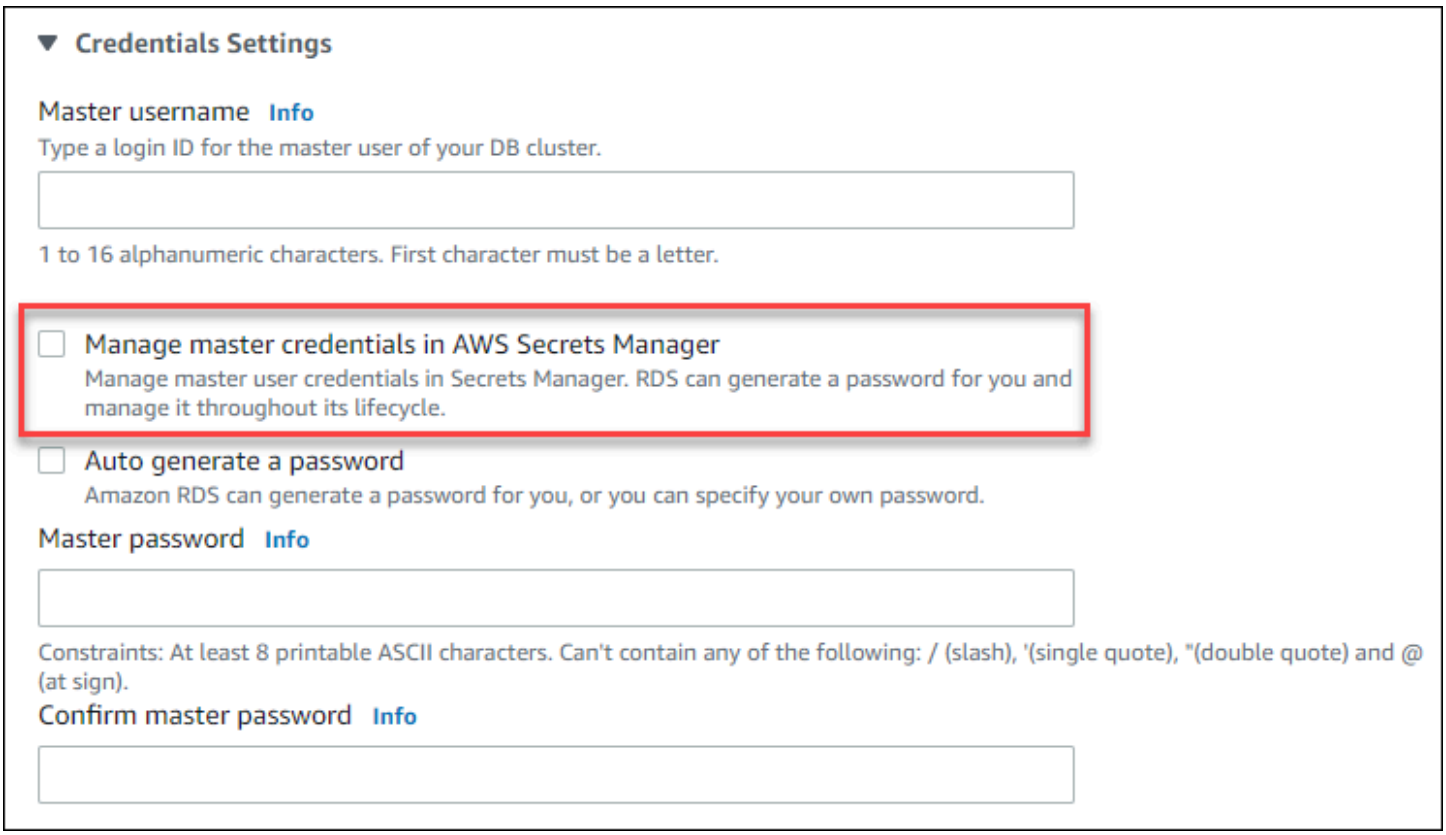
Suivez les instructions pour créer ou modifier un cluster de bases de données à l’aide de la console RDS :

- Création d’un cluster de bases de données
- Modification d’une instance de base de données dans un cluster de bases de données

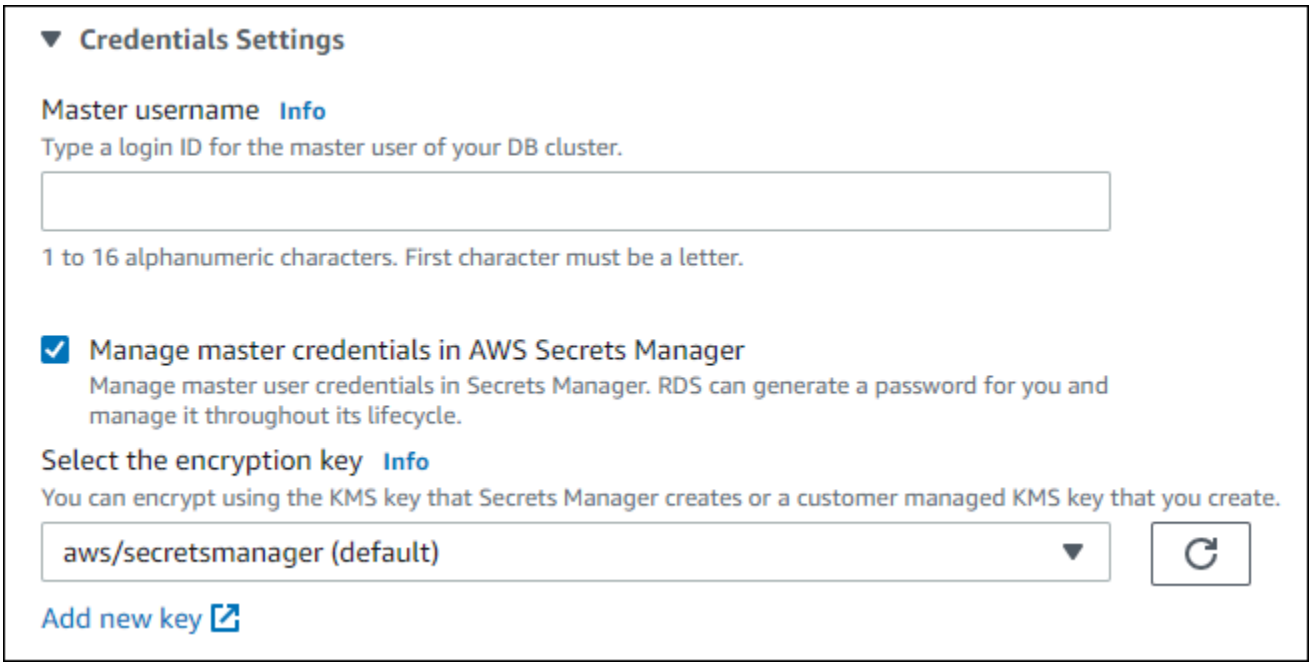
  Dans la console RDS, vous pouvez modifier n’importe quelle instance de base de données pour spécifier les paramètres de gestion des mots de passe d’utilisateur principal pour l’ensemble du cluster de bases de données.
- Restauration d’un cluster de bases de données Amazon Aurora MySQL à partir d’un compartiment Amazon S3

Lorsque vous utilisez la console RDS pour effectuer l’une de ces opérations, vous pouvez spécifier que le mot de passe d’utilisateur principal est géré par Aurora dans Secrets Manager. Pour ce faire, lorsque vous créez ou restaurez un cluster de bases de données, sélectionnez Manage master credentials in AWS Secrets Manager (Gérer les informations d’identification principales dans) dans Credential settings (Paramètres des informations d’identification). Lorsque vous modifiez un cluster de bases de données, sélectionnez Manage master credentials in AWS Secrets Manager (Gérer les informations d’identification principales dans) dans Settings (Paramètres).

L’image suivante est un exemple du paramètre Manage master credentials in AWS Secrets Manager (Gérer les informations d’identification principales dans) lors de la création ou de la restauration d’un cluster de bases de données.



Lorsque vous sélectionnez cette option, Aurora génère le mot de passe d'utilisateur principal et le gère tout au long de son cycle de vie dans Secrets Manager.



Vous pouvez choisir de chiffrer le secret à l'aide d'une clé KMS fournie par Secrets Manager ou d'une clé gérée par le client que vous créez. Quand Aurora gère les informations d'identification de base de

données pour un cluster de bases de données, vous ne pouvez pas modifier la clé KMS utilisée pour chiffrer le secret.

Vous pouvez choisir d'autres paramètres en fonction de vos besoins.

Pour plus d'informations sur les paramètres disponibles quand vous créez un cluster de bases de données, consultez Paramètres pour les clusters de bases de données Aurora. Pour plus d'informations sur les paramètres disponibles quand vous modifiez un cluster de bases de données, consultez Paramètres pour Amazon Aurora.

AWS CLI

Pour spécifier qu'Aurora doit gérer le mot de passe d'utilisateur principal dans Secrets Manager, spécifiez l'option `--manage-master-user-password` dans l'une des commandes suivantes :

- create-db-cluster
- modify-db-cluster
- restore-db-cluster-from-s3

Lorsque vous spécifiez l'option `--manage-master-user-password` dans ces commandes, Aurora génère le mot de passe d'utilisateur principal et le gère tout au long de son cycle de vie dans Secrets Manager.

Pour chiffrer le secret, vous pouvez spécifier une clé gérée par le client ou utiliser la clé KMS par défaut, fournie par Secrets Manager. Utilisez l'option `--master-user-secret-kms-key-id` pour spécifier une clé gérée par le client. L'identifiant de clé AWS KMS est l'ARN de la clé, l'ID de clé, l'alias ARN ou le nom d'alias de la clé KMS. Pour utiliser une clé KMS dans une autreCompte AWS, spécifiez l'ARN de la clé ou l'alias ARN. Quand Aurora gère les informations d'identification de base de données pour un cluster de bases de données, vous ne pouvez pas modifier la clé KMS utilisée pour chiffrer le secret.

Vous pouvez choisir d'autres paramètres en fonction de vos besoins.

Pour plus d'informations sur les paramètres disponibles quand vous créez un cluster de bases de données, consultez Paramètres pour les clusters de bases de données Aurora. Pour plus d'informations sur les paramètres disponibles quand vous modifiez un cluster de bases de données, consultez Paramètres pour Amazon Aurora.

Cet exemple crée un cluster de bases de données et spécifie qu'Aurora doit gérer le mot de passe dans Secrets Manager. Ce secret est chiffré à l'aide de la clé KMS fournie par Secrets Manager.

Example

Pour Linux, macOS ou Unix :

```
aws rds create-db-cluster \
     --db-cluster-identifier sample-cluster \
     --engine aurora-mysql \
     --engine-version 8.0 \
     --master-username admin \
     --manage-master-user-password
```

Pour Windows :

```
aws rds create-db-cluster ^
     --db-cluster-identifier sample-cluster ^
     --engine aurora-mysql ^
     --engine-version 8.0 ^
     --master-username admin ^
     --manage-master-user-password
```

API RDS

Pour spécifier que Aurora doit gérer le mot de passe d'utilisateur principal dans Secrets Manager, affectez au paramètre `ManageMasterUserPassword` la valeur `true` dans l'une des opérations suivantes :

- CréerDBCluster
- ModifyDBCluster
- Restaurer à DBCluster partir de S3

Lorsque vous affectez au paramètre `ManageMasterUserPassword` la valeur `true` dans l'une de ces opérations, Aurora génère le mot de passe d'utilisateur principal et le gère tout au long de son cycle de vie dans Secrets Manager.

Pour chiffrer le secret, vous pouvez spécifier une clé gérée par le client ou utiliser la clé KMS par défaut, fournie par Secrets Manager. Utilisez le paramètre `MasterUserSecretKmsKeyId` pour spécifier une clé gérée par le client. L'identifiant de clé AWS KMS est l'ARN de la clé, l'ID de clé, l'alias ARN ou le nom d'alias de la clé KMS. Pour utiliser une clé KMS dans un autre Compte AWS, spécifiez l'ARN de la clé ou l'ARN de l'alias. Quand Aurora gère les informations d'identification de

base de données pour un cluster de bases de données, vous ne pouvez pas modifier la clé KMS utilisée pour chiffrer le secret.

Rotation du secret de mot de passe d'utilisateur principal pour un cluster de bases de données

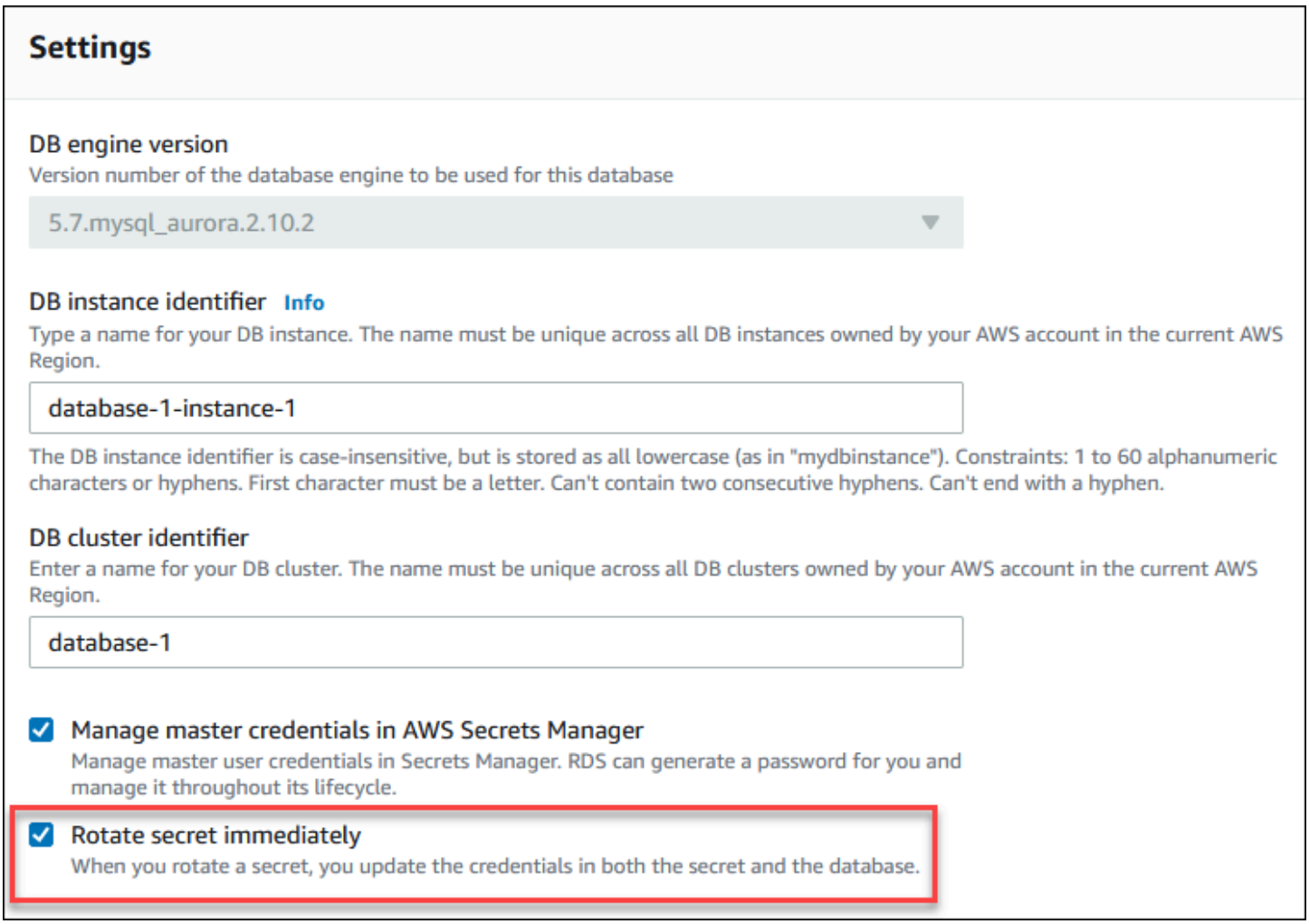
Quand Aurora effectue la rotation d'un secret de mot de passe d'utilisateur principal, Secrets Manager génère une nouvelle version de secret pour le secret existant. La nouvelle version du secret contient le nouveau mot de passe d'utilisateur principal. Aurora modifie le mot de passe d'utilisateur principal du cluster de bases de données pour qu'il corresponde au mot de passe de la nouvelle version de secret.

Vous pouvez effectuer immédiatement la rotation d'un secret au lieu d'attendre une rotation planifiée. Pour effectuer la rotation d'un secret de mot de passe d'utilisateur principal dans Secrets Manager, modifiez le cluster de bases de données . Pour plus d'informations sur la modification d'un cluster de bases de données, consultez Modification d'un cluster de bases de données Amazon Aurora.

Vous pouvez modifier immédiatement le secret d'un mot de passe utilisateur principal à l'aide de la console RDS, de l'AWS CLIAPI RDS ou de l'API RDS. Le nouveau mot de passe comporte toujours 28 caractères et contient au moins un caractère majuscule et minuscule, un chiffre et un caractère de ponctuation.

Console

Pour effectuer la rotation d'un secret de mot de passe d'utilisateur principal à l'aide de la console RDS, modifiez le cluster de bases de données et sélectionnez Rotate secret immediately (Effectuer immédiatement une rotation du secret) dans Settings (Paramètres).



Suivez les instructions pour modifier un cluster de bases de données avec la console RDS dans Modification du cluster de bases de données à partir de la console, de l'CLI (CLI) et de l'API. Vous devez choisir Apply immediately (Appliquer immédiatement) sur la page de confirmation.

AWS CLI

Pour faire pivoter le secret du mot de passe d'un utilisateur principal à l'aide deAWS CLI, utilisez la modify-db-clustercommande et spécifiez l'`--rotate-master-user-password`option. Vous devez spécifier l'option `--apply-immediately` lorsque vous effectuez la rotation du mot de passe principal.

Cet exemple effectue la rotation d'un secret de mot de passe d'utilisateur principal.

Example

Pour Linux, macOS ou Unix :

```
aws rds modify-db-cluster \
    --db-cluster-identifier mydbcluster \
    --rotate-master-user-password \
    --apply-immediately
```

Pour Windows :

```
aws rds modify-db-cluster ^
    --db-cluster-identifier mydbcluster ^
    --rotate-master-user-password ^
    --apply-immediately
```

API RDS

Vous pouvez faire pivoter le secret du mot de passe d'un utilisateur principal à l'aide de DBCluster l'opération Modifier et en définissant le `RotateMasterUserPassword` paramètre sur`true`. Vous devez affecter au paramètre `ApplyImmediately` la valeur `true` lorsque vous effectuez la rotation du mot de passe principal.

Affichage des détails concernant un secret pour un cluster de bases de données

Vous pouvez récupérer vos secrets à l'aide de la console (https://console.aws.amazon.com/secretsmanager/) ou de la commande AWS CLI (get-secret-valueSecrets Manager).

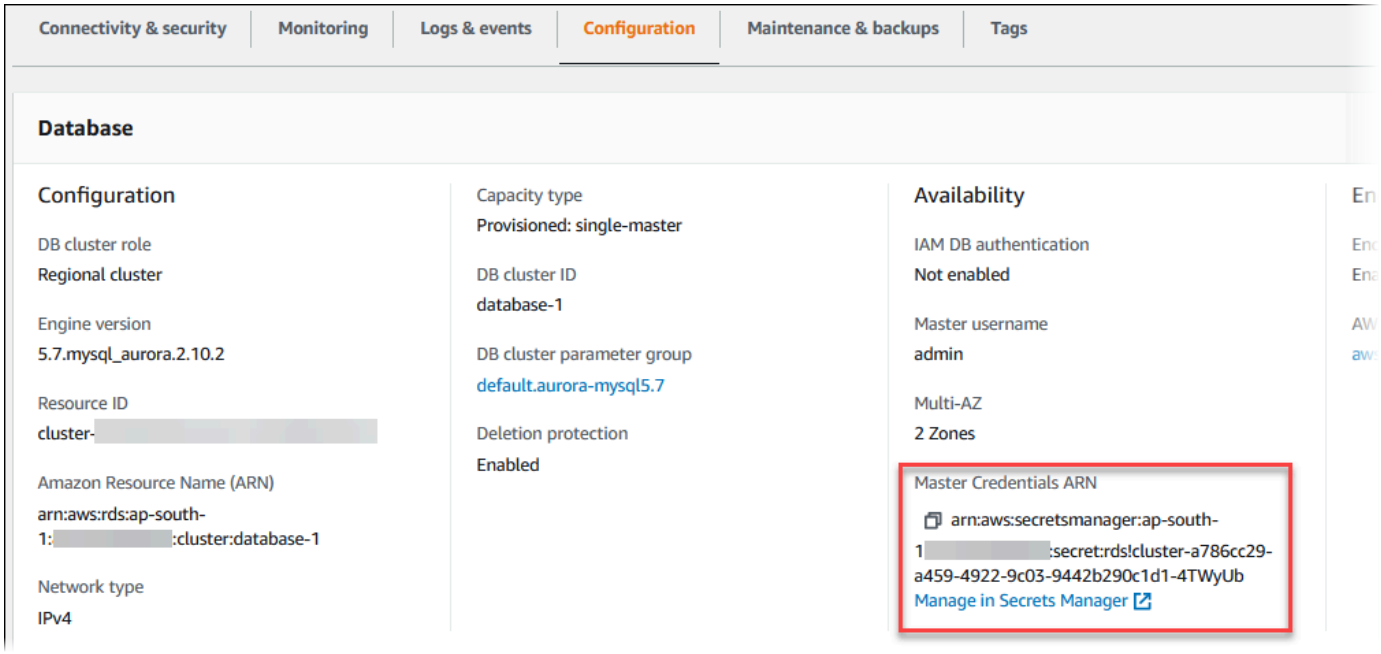
Vous pouvez trouver le Amazon Resource Name (ARN) d'un secret géré par Aurora dans Secrets Manager avec la console RDSAWS CLI, ou l'API RDS.

Console

Pour afficher les détails d’un secret géré par Aurora dans Secrets Manager

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez le nom du cluster de bases de données pour afficher ses détails.
4. Cliquez sur l’onglet Configuration.

   Dans Master Credentials ARN (ARN des informations d’identification principales), vous pouvez consulter l’ARN du secret.



Vous pouvez suivre le lien Manage in Secrets Manager (Gérer dans Secrets Manager) pour consulter et gérer le secret dans la console Secrets Manager.

AWS CLI

Vous pouvez utiliser la AWS CLI describe-db-clusterscommande RDS pour trouver les informations suivantes sur un secret géré par dans Secrets Manager :

- `SecretArn` : l'ARN du secret
- `SecretStatus` : le statut du secret

  Les valeurs de statut possibles incluent les suivantes :

  - `creating` : le secret est en cours de création.
  - `active` : le secret est disponible pour une utilisation et une rotation normales.
  - `rotating` : la rotation du secret est en cours.
  - `impaired` : le secret peut être utilisé pour accéder aux informations d'identification de base de données, mais il est impossible d'effectuer sa rotation. Un secret peut avoir ce statut si, par exemple, les autorisations sont modifiées de telle sorte que RDS ne puisse plus accéder au secret ou à la clé KMS associée à ce secret.

    Lorsqu'un secret possède ce statut, vous pouvez corriger la condition à l'origine de ce statut. Si vous corrigez la condition à l'origine du statut, celui-ci reste `impaired` jusqu'à la rotation

suivante. Vous pouvez également modifier le cluster de bases de données pour désactiver la gestion automatique des informations d'identification de base de données, puis modifier à nouveau le cluster de bases de données pour activer la gestion automatique des informations d'identification de base de données. Pour modifier le cluster de base de données, utilisez l'`--manage-master-user-password`option de la modify-db-clustercommande.

- `KmsKeyId` : l'ARN de la clé KMS utilisée pour chiffrer le secret

Spécifiez l'option `--db-cluster-identifier` permettant d'afficher la sortie pour un cluster de bases de données spécifique. Cet exemple montre la sortie d'un secret utilisé par un cluster de bases de données.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydbcluster
```

L'exemple suivant montre la sortie pour un secret :

```
"MasterUserSecret": {
                "SecretArn": "arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx",
                "SecretStatus": "active",
                "KmsKeyId": "arn:aws:kms:eu-
west-1:123456789012:key/0987dcba-09fe-87dc-65ba-ab0987654321"
            }
```

Lorsque vous disposez de l'ARN secret, vous pouvez consulter les détails du secret à l'aide de la commande get-secret-valueSecrets Manager CLI.

Cet exemple montre les détails du secret dans l'exemple de sortie précédent.

Example

Pour Linux, macOS ou Unix :

```
aws secretsmanager get-secret-value \
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

Pour Windows :

```
aws secretsmanager get-secret-value ^
    --secret-id 'arn:aws:secretsmanager:eu-west-1:123456789012:secret:rds!
cluster-033d7456-2c96-450d-9d48-f5de3025e51c-xmJRDx'
```

API RDS

Vous pouvez consulter l'ARN, le statut et la clé KMS d'un secret géré par Aurora dans Secrets Manager à l'aide de l'opération Describe DBClusters RDS et en définissant le `DBClusterIdentifier` paramètre sur un identifiant de cluster de base de données. Les détails sur le secret sont inclus dans la sortie.

Lorsque vous disposez de l'ARN secret, vous pouvez consulter les détails du secret à l'aide de l'opération GetSecretValueSecrets Manager.

Protection des données dans Amazon RDS

Le modèle de responsabilité partagée AWS s'applique à Amazon Relational Database Service. Comme décrit dans ce modèle, AWS est responsable de la protection de l'infrastructure globale sur laquelle l'ensemble du AWS Cloud s'exécute. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécurité AWS.

À des fins de protection des données, nous vous recommandons de protéger les informations d'identification Compte AWS et de configurer les comptes utilisateur individuels avec AWS IAM Identity Center ou Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- Utilisez les certificats SSL/TLS pour communiquer avec les ressources AWS. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez une API (Interface de programmation) et la journalisation des activités des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des sentiers CloudTrail pour

capturer des activités AWS, consultez la section Utilisation des sentiers CloudTrail dans le Guide de l'utilisateur AWS CloudTrail.

- Utilisez des solutions de chiffrement AWS, ainsi que tous les contrôles de sécurité par défaut au sein des Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules de chiffrement validés FIPS (Federal Information Processing Standard) 140-3 lorsque vous accédez à AWS via une interface de ligne de commande ou une API (interface de programmation), utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez Norme FIPS (Federal Information Processing Standard) 140-3.

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela s'applique aussi lorsque vous utilisez Amazon RDS ou d'autres Services AWS à l'aide de la console, de l'API, de l'AWS CLI ou des kits SDK AWS. Toutes les données que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Rubriques

Protection des données à l'aide du chiffrement

Vous pouvez activer le chiffrement pour vos ressources de base de données. Vous pouvez également chiffrer les connexions aux clusters de bases de données.

Rubriques

Chiffrement des ressources Amazon Aurora

Amazon Aurora peut chiffrer vos Amazon Aurora clusters de bases de données. Les données chiffrées au repos incluent le stockage sous-jacent pour les clusters de bases de données, les sauvegardes automatiques, les réplicas en lecture et les instantanés.

Les clusters de bases de données chiffrée Amazon Aurora utilisent l'algorithme de chiffrement AES-256 standard pour chiffrer vos données sur le serveur qui héberge vos clusters de bases de données Amazon Aurora. Une fois que vos données ont été chiffrées, Amazon Aurora traite l'authentification de l'accès et le déchiffrement de vos données de façon transparente, avec un impact minimal sur les performances. Vous n'avez pas besoin de modifier vos applications clientes de base de données pour utiliser le chiffrement.

> **Note**
>
> Pour les clusters de bases de données chiffrés et non chiffrés, les données en transit entre la source et le réplica en lecture sont chiffrées, même en cas de réplication entre régions AWS.

Rubriques

Présentation du chiffrement des ressources Amazon Aurora

Les clusters de base de données chiffrée Amazon Aurora fournissent une couche supplémentaire de protection des données en sécurisant vos données contre tout accès non autorisé au stockage sous-jacent. Vous pouvez utiliser le chiffrement Amazon Aurora pour renforcer la protection des données de vos applications déployées dans le cloud et pour satisfaire aux exigences de conformité pour le chiffrement au repos. Pour un cluster de bases de données chiffrée Amazon Aurora, les instances de bases de données, journaux, sauvegardes et instantanés sont tous chiffrés. Pour plus d'informations sur la disponibilité et les limites du chiffrement, consultez Disponibilité du chiffrement Amazon Aurora et Limitations des clusters de bases de données chiffrées Amazon Aurora.

Amazon Aurora utilise une clé AWS Key Management Service pour chiffrer ces ressources. AWS KMS combine du matériel et des logiciels sécurisés et hautement disponibles pour fournir un système de gestion de clés à l'échelle du cloud. Vous pouvez utiliser une Clé gérée par AWS ou créer des clés gérées par le client.

Lorsque vous créez un cluster de bases de données chiffrées, vous pouvez choisir une clé gérée par le client ou la Clé gérée par AWS pour Amazon Aurora pour chiffrer votre cluster de bases de données. Si vous ne spécifiez pas l'identifiant de clé d'une clé gérée par le client, Amazon Aurora utilise la Clé gérée par AWS pour votre nouveau cluster de bases de données. Amazon Aurora crée une Clé gérée par AWS pour Amazon Aurora pour votre compte AWS. Votre compte AWS dispose d'une Clé gérée par AWS pour Amazon Aurora différente pour chaque région AWS.

Pour gérer les clés gérées par le client qui sont utilisées pour le chiffrement et le déchiffrement de vos ressources Amazon Aurora, vous utilisez AWS Key Management Service (AWS KMS).

En utilisant AWS KMS, vous pouvez créer des clés gérées par le client de chiffrement et définir les stratégies pour contrôler l'utilisation de ces clés gérées par le client. AWS KMS prend en charge CloudTrail afin de vous permettre d'auditer l'utilisation de la clé KMS et de vérifier que les clés gérées par le client sont utilisées de manière appropriée. Vous pouvez utiliser vos clés gérées par le client avec Amazon Aurora et les services AWS pris en charge tels que Amazon S3, Amazon EBS et Amazon Redshift. Pour obtenir la liste des services intégrés à AWS KMS, consultez Intégration de services AWS. Quelques considérations relatives à l'utilisation des clés KMS :

- Une fois que vous avez créé une instance de base de données chiffrée, vous ne pouvez pas modifier la clé KMS utilisée par cette instance de base de données. Vous devez donc prendre soin de déterminer vos besoins en termes de clés KMS avant de créer votre instance de base de données chiffrée.

  Si vous devez modifier la clé de chiffrement de votre cluster de bases de données, créez un instantané manuel de votre cluster et activez le chiffrement lors de la copie de l'instantané. Pour plus d'informations, consultez cet article re:Post Knowledge.
- Si vous copiez un instantané chiffré, vous pouvez utiliser une clé KMS différente pour chiffrer l'instantané cible que celle utilisée pour chiffrer l'instantané source.
- Vous ne pouvez pas partager un instantané chiffré à l'aide de la Clé gérée par AWS du compte AWS qui a partagé l'instantané.
- Chaque instance de base de données du cluster de bases de données est chiffrée à l'aide de la même clé KMS que le cluster de bases de données.
- Vous pouvez également chiffrer un réplica en lecture d'un cluster Amazon Aurora chiffré.

Important

Dans certains cas, Amazon Aurora peut perdre l'accès à la clé KMS pour un cluster de bases de données lorsque vous désactivez la clé KMS. Dans ce cas, le cluster de bases de données chiffré entre dans l'état `inaccessible-encryption-credentials-recoverable`. Le cluster de bases de données reste dans cet état pendant sept jours, pendant lesquels l'instance est arrêtée. Les appels d'API effectués vers le cluster de bases de données pendant cette période risquent d'échouer. Pour restaurer le cluster de bases de données, activez la clé KMS, puis redémarrez le cluster. Activez la clé KMS depuis la AWS Management Console, l'AWS CLI, ou l'API RDS. Redémarrez le cluster de bases de données à l'aide de la commande AWS CLI start-db-cluster ou la AWS Management Console. L'état `inaccessible-encryption-credentials-recoverable` s'applique uniquement aux clusters de bases de données qui peuvent être interrompus. Cet état récupérable ne s'applique pas aux instances qui ne peuvent pas s'arrêter, telles que les clusters contenant des réplicas en lecture inter-régions. Pour plus d'informations, consultez the section called "Limites".

Si le cluster de bases de données n'est pas récupéré dans un délai de sept jours, il passe à l'état `inaccessible-encryption-credentials` terminal. Dans cet état, le cluster de bases de données n'est plus utilisable et vous ne pouvez le restaurer qu'à partir d'une sauvegarde. Nous vous recommandons vivement de toujours activer les sauvegardes pour les clusters de bases de données chiffrées afin de vous prémunir contre la perte de données chiffrées dans vos bases de données.

Lors de la création d'un cluster de bases de données, Aurora vérifie si le principal appelant dispose a accès à la clé KMS, puis génère une autorisation à partir de cette clé KMS, qu'il utilisera pendant toute la durée de vie du cluster de bases de données. La révocation de l'accès du principal appelant à la clé KMS n'affecte pas la base de données en cours d'exécution. Lorsque vous utilisez des clés KMS dans des scénarios entre comptes, tels que la copie d'un instantané sur un autre compte, la clé KMS doit être partagée avec l'autre compte. Si vous créez un cluster de bases de données à partir de l'instantané sans spécifier de clé KMS différente, le nouveau cluster utilise la clé KMS du compte source. Le fait de révoquer l'accès à la clé après la création du cluster de bases de données n'a aucun impact sur ce cluster. Toutefois, la désactivation de la clé affecte tous les clusters de bases de données chiffrés avec cette clé. Afin d'éviter cette situation, indiquez une clé différente au moment de la copie de l'instantané.

Pour plus d'informations sur les clés KMS, consultez AWS KMS keys dans le Guide du développeur AWS Key Management Service et Gestion AWS KMS key.

Chiffrement d'un cluster de bases de données Amazon Aurora

Pour chiffrer un nouveau cluster de base de données, sélectionnez Enable encryption (Activer le chiffrement) dans la console. Pour plus d'informations sur la création d'un cluster de bases de données, consultez Création d'un cluster de bases de données Amazon Aurora.

Si vous utilisez la commande d'AWS CLI create-db-cluster pour créer un cluster de bases de données chiffrée, définissez le paramètre `--storage-encrypted`. Si vous utilisez l'opération d'API CreateDBCluster, affectez au paramètre `StorageEncrypted` la valeur true.

Une fois que vous avez créé un cluster de bases de données chiffrées, vous ne pouvez pas modifier la clé KMS pour ce cluster de bases de données. Vous devez donc prendre soin de déterminer vos besoins en termes de clés KMS avant de créer votre cluster de base de données chiffrées.

Si vous utilisez la commande AWS CLI `create-db-cluster` pour créer un cluster de bases de données chiffré avec une clé gérée par le client, définissez le paramètre `--kms-key-id` sur n'importe quel identifiant de clé pour la clé KMS. Si vous utilisez l'opération Amazon RDS de l'API `CreateDBInstance`, définissez le paramètre `KmsKeyId` sur n'importe quel identifiant de clé pour la clé KMS. Pour utiliser une clé gérée par le client dans un autre compte AWS, spécifiez l'ARN de clé ou ARN d'alias.

Détermination si le chiffrement est activé pour un cluster de bases de données

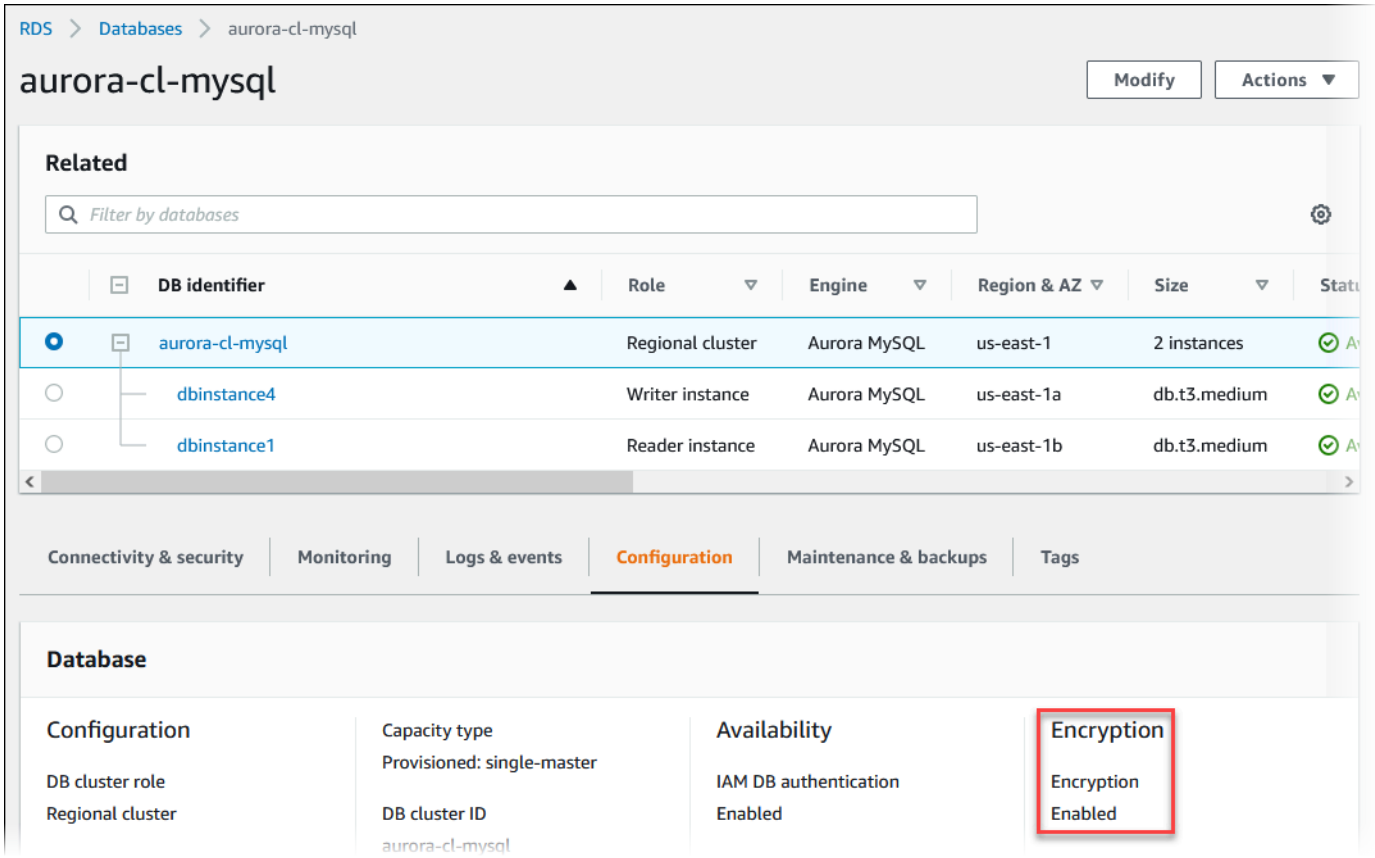
Vous pouvez utiliser la AWS Management Console, l'AWS CLI ou l'API RDS pour déterminer si le chiffrement au repos est activé pour un cluster de bases de données.

Console

Pour déterminer si le chiffrement au repos est activé pour un cluster de bases de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Bases de données.
3. Sélectionnez le nom du cluster de base de données que vous souhaitez vérifier pour en voir les détails.
4. Cliquez sur l'onglet Configuration et cochez la case Encryption (Chiffrement).

   Il indique Enabled (Activé) ou Not enabled (Non activé).



AWS CLI

Pour déterminer si le chiffrement au repos est activé pour un cluster de bases de données à l'aide de la AWS CLI, appelez la commande describe-db-clusters avec l'option suivante :

- `--db-cluster-identifier` : nom du cluster de bases de données.

L'exemple suivant utilise une requête pour renvoyer `TRUE` ou `FALSE` concernant le chiffrement au repos pour le cluster de bases de données `mydb`.

Example

```
aws rds describe-db-clusters --db-cluster-identifier mydb --query "*[].
{StorageEncrypted:StorageEncrypted}" --output text
```

API RDS

Pour déterminer si le chiffrement au repos est activé pour un cluster de bases de données à l'aide de l'API Amazon RDS, appelez l'opération DescribeDBClusters avec le paramètre suivant :

- `DBClusterIdentifier` : nom du cluster de bases de données.

Disponibilité du chiffrement Amazon Aurora

Le chiffrement Amazon Aurora est actuellement disponible pour tous les moteurs de base de données et types de stockage.

> **Note**
>
> Le chiffrement Amazon Aurora n'est pas disponible pour la classe d'instance de base de données db.t2.micro.

Chiffrement en transit

Chiffrement au niveau de la couche physique

Toutes les données circulant à travers les Régions AWS sur le réseau global AWS sont automatiquement chiffrées au niveau de la couche physique avant qu'elles ne quittent les installations sécurisées AWS. Tout le trafic entre zones de disponibilité est chiffré. Des couches supplémentaires de chiffrement, y compris celles présentées dans cette section, peuvent fournir des protections supplémentaires.

Chiffrement assuré par un appairage Amazon VPC et un appairage entre régions de la passerelle de transit

Tout le trafic entre régions qui utilise un appairage VPC Amazon et Transit Gateway est automatiquement chiffré en bloc quand il quitte une région. Une couche supplémentaire de chiffrement est automatiquement fournie au niveau de la couche physique pour tout le trafic avant que celui-ci quitte les installations sécurisées AWS.

Chiffrement entre instances

AWS assure une connectivité sécurisée et privée entre les instances de base de données de tous types. En outre, certains types d'instances utilisent les capacités de déchargement du matériel du système Nitro sous-jacent pour chiffrer automatiquement le trafic en transit entre instances. Ce chiffrement utilise des algorithmes de chiffrement authentifié avec données associées (AEAD), avec un chiffrement 256 bits. Il n'y a aucun impact sur les performances du réseau. Pour prendre en charge ce chiffrement supplémentaire du trafic en transit entre les instances, les exigences suivantes doivent être satisfaites :

- Les instances utilisent les types d’instance suivants :
  - Usage général : M6i, M6id, M6in, M6idn, M7g
  - Optimisé pour la mémoire : R6i, R6id, R6in, R6idn, R7g, X2idn, X2iedn, X2iezn
- Les instances se trouvent dans la même Région AWS.
- Les instances se trouvent dans le même VPC ou dans des VPC appairés, et le trafic ne passe pas par un service ou un périphérique de réseau virtuel, tel qu’un équilibreur de charge ou une passerelle de transit.

Limitations des clusters de bases de données chiffrées Amazon Aurora

Les limitations suivantes existent pour les clusters de bases de données chiffrés Amazon Aurora :

- Vous ne pouvez pas désactiver le chiffrement d’un cluster de bases de données chiffré.
- Vous ne pouvez pas créer d’instantané chiffré de cluster de bases de données non chiffrées.
- Un instantané de cluster de bases de données chiffrées doit être chiffré à l’aide de la même clé KMS que le cluster de bases de données.
- Vous ne pouvez pas convertir un cluster de base de données non chiffrée vers un cluster chiffré. Toutefois, vous pouvez restaurer un instantané non chiffré dans un cluster de bases de données Aurora chiffré. Pour ce faire, spécifiez une clé KMS lorsque vous procédez à la restauration à partir de l’instantané non chiffré.
- Vous ne pouvez pas créer de réplica Aurora chiffré à partir d’un cluster de bases de données Aurora non chiffré. Vous ne pouvez pas créer de réplica Aurora non chiffré à partir d’un cluster de bases de données Aurora chiffré.
- Pour copier un instantané chiffré d’une région AWS à une autre, vous devez spécifier la clé KMS de la région AWS de destination. En effet, les clés KMS sont spécifiques à la régionAWS dans laquelle elles sont créées.

  L’instantané source reste chiffré pendant tout le processus de copie. Amazon Aurora utilise un chiffrement d’enveloppe pour protéger les données pendant le processus de copie. Pour plus d’informations sur le chiffrement d’enveloppe, consultez Chiffrement d’enveloppe dans le Guide du développeurAWS Key Management Service.
- Vous ne pouvez pas déchiffrer un d’instances de base de données chiffrées. Vous pouvez cependant exporter des données à partir d’un d’instances de base de données et importer les données dans un d’instances de base de données non chiffrées.

Gestion AWS KMS key

Amazon Aurora s'intègre automatiquement avec AWS Key Management Service (AWS KMS) pour la gestion des clés. Amazon Aurora utilise le chiffrement d'enveloppe. Pour plus d'informations sur le chiffrement d'enveloppe, consultez Chiffrement d'enveloppe dans le Guide du développeurAWS Key Management Service.

Vous pouvez utiliser deux types de clés AWS KMS pour chiffrer vos clusters de bases de données.

- Si vous souhaitez un contrôle total sur une clé KMS, vous devez créer une clé gérée par le client. Pour plus d'informations sur les clés gérées par le client, consultez Clés gérées par le client dans le Guide du développeur AWS Key Management Service.
- Clés gérées par AWS sont des clés KMS de votre compte qui sont créées, gérées et utilisées en votre nom par un service AWS intégré à AWS KMS. Par défaut, la Clé gérée par AWS RDS (`aws/rds`) est utilisée pour le chiffrement. Vous ne pouvez pas gérer, faire pivoter ni supprimer la Clé gérée par AWS RDS. Pour plus d'informations sur les Clés gérées par AWS, consultez Clés gérées par AWS dans le Guide du développeur AWS Key Management Service.

Pour gérer les clés KMS utilisées pour les clusters de bases de données chiffrés Amazon Aurora, utilisez AWS Key Management Service (AWS KMS) dans la console AWS KMS, l'AWS CLI ou l'API AWS KMS. Pour consulter les journaux d'audit de chaque action effectuée à l'aide d'une clé gérée par le client ou par AWS, utilisez AWS CloudTrail. Pour plus d'informations sur la rotation des clés, consultez Rotation des clés AWS KMS.

Autoriser l'utilisation d'une clé gérée par le client

Quand Aurora utilise une clé gérée par le client dans le cadre d'opérations de chiffrement, il agit au nom de l'utilisateur qui crée ou modifie la ressource Aurora.

Pour créer une ressource Aurora à l'aide d'une clé gérée par un client, un utilisateur doit avoir les autorisations nécessaires pour appeler les opérations suivantes sur la clé gérée par le client :

- `kms:CreateGrant`
- `kms:DescribeKey`

Vous pouvez spécifier les autorisations requises dans une politique de clé ou dans une IAM politique si la politique de clé le permet.

> **Important**
>
> Lorsque vous utilisez des déclarations de refus explicites pour toutes les ressources (*) dans les stratégies de clé AWS KMS avec des services gérés tels qu'Amazon RDS, vous devez spécifier une condition pour autoriser le compte propriétaire de la ressource. Les opérations peuvent échouer sans cette condition, même si la règle de refus inclut des exceptions pour votre utilisateur IAM.

> **Tip**
>
> Pour suivre le principe du moindre privilège, n'autorisez pas l'accès complet à `kms:CreateGrant`. Au lieu de cela, utilisez la clé de condition kms:ViaService pour autoriser l'utilisateur à créer des octrois sur la clé KMS uniquement lorsque l'octroi est créé pour le compte de l'utilisateur par un service AWS.

Vous pouvez renforcer la politique IAM de différentes manières. Par exemple, si vous voulez limiter l'utilisation de la clé gérée par le client aux seules demandes provenant de Aurora, utilisez la clé de condition kms:ViaService avec la valeur `rds.<region>.amazonaws.com`. Vous pouvez également utiliser les clés ou les valeurs du contexte Contexte de chiffrement Amazon RDS comme condition d'utilisation de la clé gérée par le client pour le chiffrement.

Pour plus d'informations, consultez Autoriser des utilisateurs d'autres comptes à utiliser une clé KMS dans le Guide du développeur AWS Key Management Service.

Contexte de chiffrement Amazon RDS

Quand Aurora utilise votre clé KMS ou quand Amazon EBS utilise la clé KMS pour le compte d'Aurora, le service spécifie un contexte de chiffrement. Le contexte de chiffrement représente des informations authentifiées supplémentaires (AAD) qu'AWS KMS utilise afin de garantir l'intégrité des données. Autrement dit, quand un contexte de chiffrement est spécifié pour une opération de chiffrement, le service doit spécifier le même contexte de chiffrement pour l'opération de déchiffrement. Dans le cas contraire, le déchiffrement échoue. Le contexte de chiffrement est également écrit dans vos journaux AWS CloudTrail pour vous aider à comprendre pourquoi une clé KMS donnée a été utilisée. Vos journaux CloudTrail peuvent contenir de nombreuses entrées décrivant l'utilisation d'une clé KMS, mais le contexte de chiffrement figurant dans chaque entrée de journal peut vous aider à déterminer la raison de cette utilisation particulière.

Au minimum, Aurora utilise toujours l'ID de cluster de bases de données pour le contexte de chiffrement, comme dans l'exemple au format JSON suivant :

```
{ "aws:rds:dbc-id": "cluster-CQYSMDPBRZ7BPMH7Y3RTDG5QY" }
```

Ce contexte de chiffrement peut vous aider à identifier le cluster de bases de données pour lequel votre clé KMS a été utilisée.

Quand votre clé KMS est utilisée pour un cluster de bases de données spécifique et un volume Amazon EBS spécifique, l'ID du cluster de bases de données et l'ID de volume Amazon EBS sont utilisés pour le contexte de chiffrement, comme dans l'exemple au format JSON suivant :

```
{
  "aws:rds:dbc-id": "cluster-BRG7VYS3SVIFQW7234EJQOM5RQ",
  "aws:ebs:id": "vol-ad8c6542"
}
```

Utilisation SSL/TLS pour chiffrer une connexion à une de clusters

Vous pouvez utiliser le protocole SSL (Secure Socket Layer) ou le protocole TLS (Transport Layer Security) à partir de votre application pour chiffrer une connexion à un cluster de bases de données exécutant Aurora MySQL ou Aurora PostgreSQL.

Les connexions SSL/TLS fournissent une couche de sécurité en chiffrant les données en transit entre votre client et le cluster de bases de données. Votre SSL/TLS connexion peut éventuellement vérifier l'identité du serveur en validant le certificat de serveur installé sur votre base de données. Pour exiger la vérification de l'identité du serveur, suivez ce processus général :

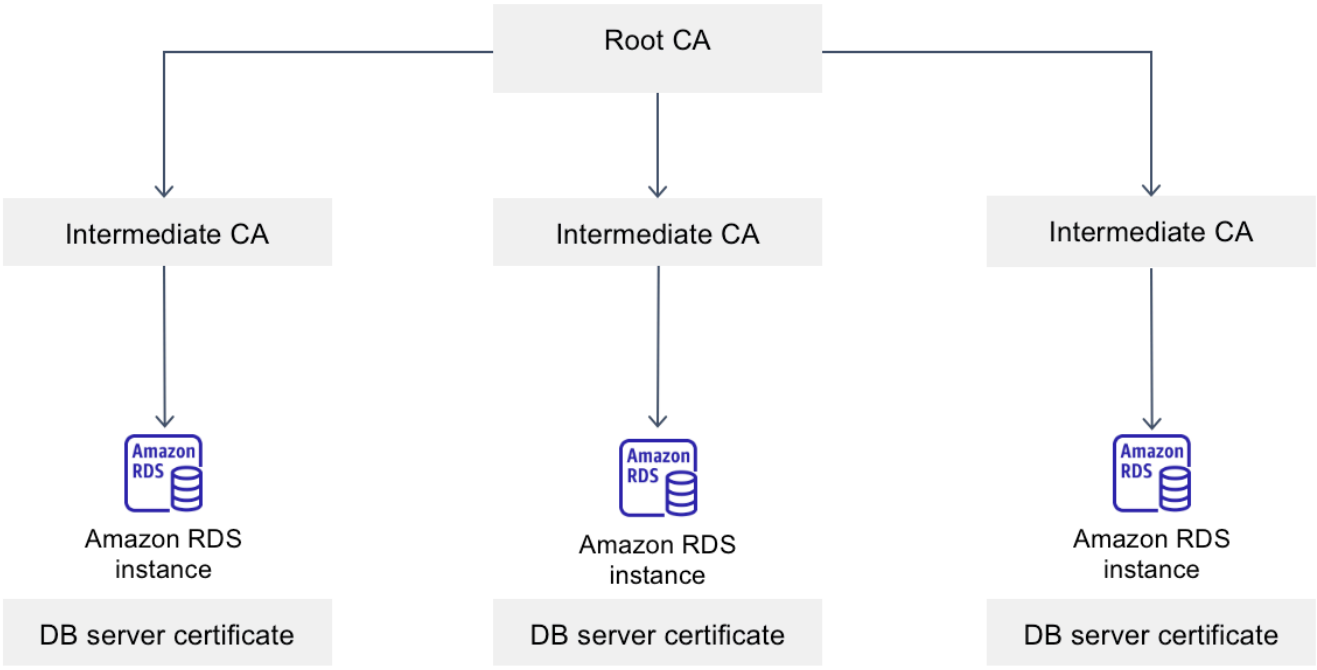
1. Choisissez l'autorité de certification (CA) qui signe le certificat de serveur de base de données pour votre base de données. Pour plus d'informations sur les autorités de certification, consultez Autorités de certification .
2. Téléchargez une offre groupée de certificats à utiliser lorsque vous vous connectez à la base de données. Pour télécharger une offre groupée de certificats, consultez Forfaits de certificats par Région AWS .

   > **Note**
   >
   > Tous les certificats sont disponibles uniquement pour le téléchargement sur des connexions SSL/TLS.

3. Connectez-vous à la base de données en utilisant le processus de votre moteur de base de données pour implémenter SSL/TLS les connexions. Chaque moteur de base de données possède son propre processus d'implémentation SSL/TLS. To learn how to implement SSL/TLS pour votre base de données. Suivez le lien correspondant à votre moteur de base de données :
   - Sécurité avec Amazon Aurora MySQL
   - Sécurité avec Amazon Aurora PostgreSQL

Autorités de certification

L'autorité de certification (CA) est le certificat qui identifie l'autorité de certification racine en haut de la chaîne de certificats. L'autorité de certification signe le certificat de serveur de base de données, qui est installé sur chaque instance de base de données. Le certificat de serveur de base de données identifie l'instance de base de données en tant que serveur approuvé.



Amazon RDS fournit les éléments suivants CAs pour signer le certificat du serveur de base de données d'une base de données.

Autorité de certification (CA)	Description	Nom commun
rds-ca-rsa2048-g1	Utilise une autorité de certification avec l'algorithme de clé privée RSA 2048 et	Amazon RDS *region-id*

Autorité de certification (CA)	Description	Nom commun
	l'algorithme de SHA256 signature dans la plupart des cas. Régions AWS Dans le AWS GovCloud (US) Regions, cette autorité de certification utilise une autorité de certification avec un algorithm e de clé privée et un algorithme de SHA384 signature RSA 2048. Cette autorité de certification prend en charge la rotation automatique des certificats de serveur.	*entifier* Root CA RSA2 048 G1
rds-ca-rsa4096-g1	Utilise une autorité de certification avec l'algorithme de clé privée et l'algorithme de SHA384 signature RSA 4096. Cette autorité de certification prend en charge la rotation automatique des certificats de serveur.	Amazon RDS *region-id* *entifier* Root CA RSA4 096 G1
rds-ca-ecc384-g1	Utilise une autorité de certification avec un algorithme de clé privée et un algorithme de SHA384 signature ECC 384. Cette autorité de certification prend en charge la rotation automatique des certificats de serveur.	Amazon RDS *region-id* *entifier* Root CA G1 ECC384

> **Note**
>
> Si vous utilisez le AWS CLI, vous pouvez vérifier la validité des autorités de certification répertoriées ci-dessus en utilisant describe-certificates.

Ces certificats de CA sont inclus dans la solution groupée de certificats régionaux et mondiaux. Lorsque vous utilisez l'autorité de certification rds-ca-rsa 2048-g1, rds-ca-rsa 4096-g1 ou rds-ca-ecc

384-g1 avec une base de données, RDS gère le certificat du serveur de base de données sur la base de données. RDS effectue automatiquement la rotation du certificat de serveur de base de données avant son expiration.

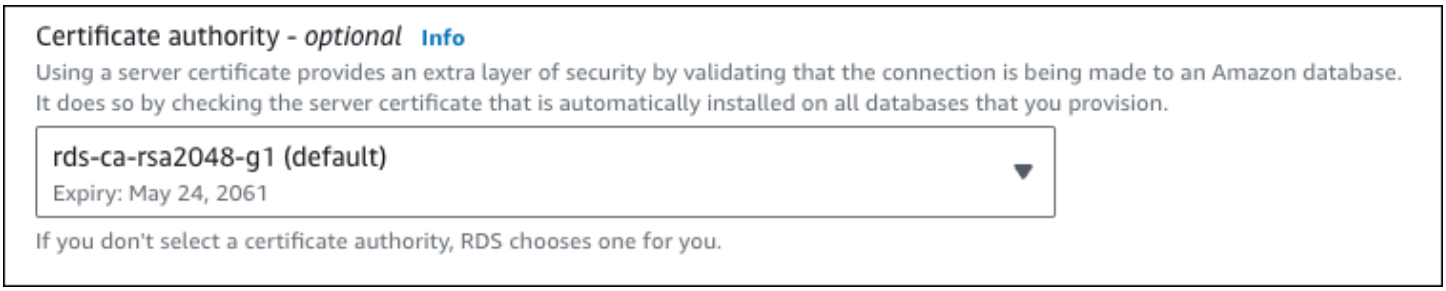
Configuration de l’autorité de certification pour votre base de données

Vous pouvez définir l’autorité de certification pour une base de données lorsque vous effectuez les tâches suivantes :

- Créer un cluster de base de données Aurora — Vous pouvez définir l'autorité de certification pour une instance de base de données dans un cluster Aurora lorsque vous créez la première instance de base de données dans le cluster de base de données à l'aide de l'API AWS CLI ou RDS. Actuellement, vous ne pouvez pas configurer l’autorité de certification des instances de base de données dans un cluster de bases de données lorsque vous créez le cluster de bases de données à l’aide de la console RDS. Pour obtenir des instructions, consultez Création d’un cluster de bases de données Amazon Aurora.
- Modification d’une instance de base de données : vous pouvez définir l’autorité de certification d’une instance de base de données dans un cluster de bases de données en la modifiant. Pour obtenir des instructions, consultez Modification d’une instance de base de données dans un cluster de bases de données .

> **Note**
>
> L'autorité de certification par défaut est définie sur rds-ca-rsa 2048-g1. Vous pouvez remplacer l'autorité de certification par défaut pour votre Compte AWS compte à l'aide de la commande modify-certificates.

La disponibilité CAs dépend du moteur de base de données et de la version du moteur de base de données. Lorsque vous utilisez le AWS Management Console, vous pouvez choisir l'autorité de certification à l'aide du paramètre Autorité de certification, comme indiqué dans l'image suivante.



La console affiche uniquement ceux CAs qui sont disponibles pour le moteur de base de données et la version du moteur de base de données. Si vous utilisez le AWS CLI, vous pouvez définir l'autorité de certification pour une instance de base de données à l'aide de la modify-db-instancecommande create-db-instanceor.

Si vous utilisez le AWS CLI, vous pouvez voir ce qui est disponible CAs pour votre compte en utilisant la commande describe-certificates. Cette commande indique également la date d'expiration de chaque autorité de certification dans `ValidTill`, dans la sortie. Vous pouvez trouver ceux CAs qui sont disponibles pour un moteur de base de données et une version de moteur de base de données spécifiques à l'aide de la describe-db-engine-versionscommande.

L'exemple suivant montre la version CAs disponible pour la version par défaut du moteur de base de données RDS pour PostgreSQL.

```
aws rds describe-db-engine-versions --default-only --engine postgres
```

Votre sortie est similaire à ce qui suit. Les options disponibles CAs sont répertoriées dans`SupportedCACertificateIdentifiers`. La sortie indique également si la version du moteur de base de données prend en charge la rotation du certificat sans redémarrage dans `SupportsCertificateRotationWithoutRestart`.

```
{
    "DBEngineVersions": [
        {
            "Engine": "postgres",
            "MajorEngineVersion": "13",
            "EngineVersion": "13.4",
            "DBParameterGroupFamily": "postgres13",
            "DBEngineDescription": "PostgreSQL",
            "DBEngineVersionDescription": "PostgreSQL 13.4-R1",
            "ValidUpgradeTarget": [],
            "SupportsLogExportsToCloudwatchLogs": false,
            "SupportsReadReplica": true,
            "SupportedFeatureNames": [
                "Lambda"
            ],
            "Status": "available",
            "SupportsParallelQuery": false,
            "SupportsGlobalDatabases": false,
            "SupportsBabelfish": false,
            "SupportsCertificateRotationWithoutRestart": true,
```

```
            "SupportedCACertificateIdentifiers": [
                "rds-ca-rsa2048-g1",
                "rds-ca-ecc384-g1",
                "rds-ca-rsa4096-g1"
            ]
        }
    ]
}
```

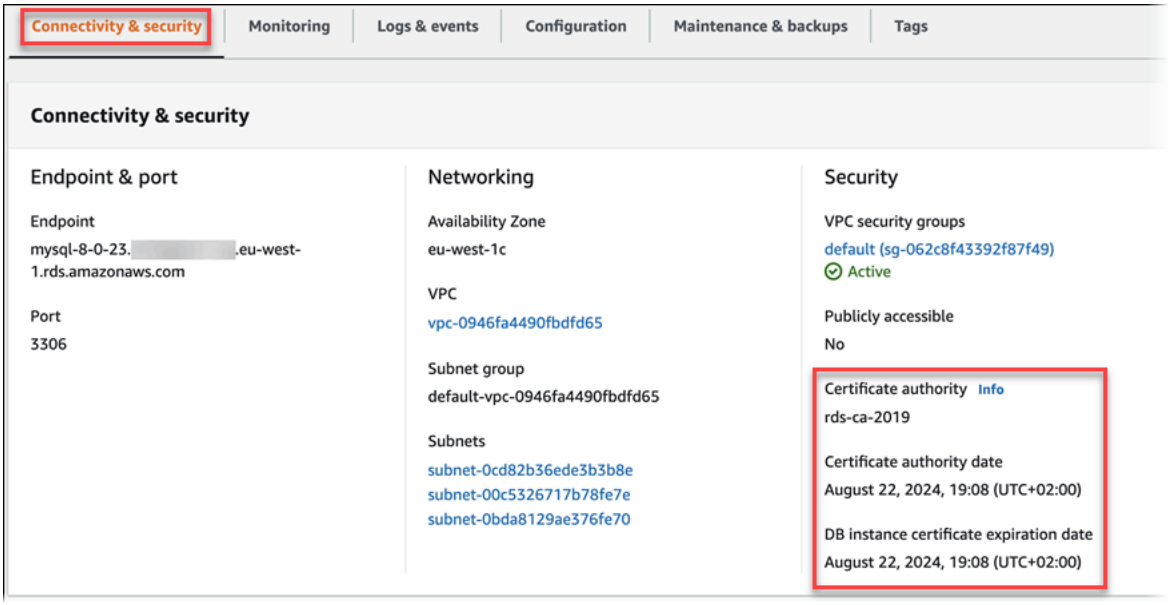
Validité des certificats de serveur de base de données

La validité du certificat de serveur de base de données dépend du moteur de base de données et de la version du moteur de base de données. Si la version du moteur de base de données prend en charge la rotation du certificat sans redémarrage, la validité du certificat de serveur de base de données est de 1 an. Dans le cas contraire, la validité est de 3 ans.

Pour plus d'informations sur la rotation des certificats de serveur de base de données, consultez Rotation automatique du certificat de serveur.

Visualisation de la CA pour votre instance de base de données

Vous pouvez consulter les détails concernant la CA d'une instance de base de données en affichant l'onglet Connectivité et sécurité de la console, comme illustré dans l'image suivante.



Si vous utilisez le AWS CLI, vous pouvez afficher les détails de l'autorité de certification pour une instance de base de données à l'aide de la describe-db-instancescommande.

Téléchargement d’offres groupées de certificats pour Aurora

Lorsque vous vous connectez à votre base de données via SSL ou TLS, l’instance de base de données nécessite un certificat de confiance d’Amazon RDS. Sélectionnez le lien approprié dans le tableau suivant pour télécharger l’offre correspondant à la Région AWS où vous hébergez votre base de données.

Forfaits de certificats par Région AWS

Les ensembles de certificats pour toutes les régions Régions AWS et GovCloud (États-Unis) contiennent les certificats CA racine suivants :

- `rds-ca-rsa2048-g1`
- `rds-ca-rsa4096-g1`
- `rds-ca-ecc384-g1`

Les certificats `rds-ca-ecc384-g1` et `rds-ca-rsa4096-g1` ne sont pas disponibles dans les régions suivantes :

- Asie-Pacifique (Mumbai)
- Asie-Pacifique (Melbourne)
- Canada-Ouest (Calgary)
- Europe (Zurich)
- Europe (Espagne)
- Israël (Tel Aviv)

Le magasin de confiance de votre application doit uniquement enregistrer le certificat CA racine. N'enregistrez pas les certificats CA intermédiaires dans votre magasin de confiance car cela pourrait entraîner des problèmes de connexion lorsque RDS fait automatiquement pivoter votre certificat de serveur de base de données.

> **Note**
>
> Le proxy et l'utilisation d'Amazon RDS Aurora Serverless v1 les certificats du AWS Certificate Manager (ACM). Si vous utilisez le proxy RDS, vous n’avez pas besoin de télécharger des certificats Amazon RDS ou de mettre à jour des applications utilisant des

connexions de proxy RDS. Pour plus d'informations, consultez Utilisation TLS/SSL avec RDS Proxy.

Si vous utilisez Aurora Serverless v1, le téléchargement des certificats Amazon RDS n'est pas nécessaire. Pour plus d'informations, consultez Utilisation de TLS/SSL avec Aurora Serverless v1.

Pour télécharger un bundle de certificats pour un Région AWS, sélectionnez le lien correspondant à celui Région AWS qui héberge votre base de données dans le tableau suivant.

AWS Région	Solution groupée de certificats (PEM)	Ensemble de certificats (PKCS7)
N'importe quelle publicité Région AWS	global-bundle.pem	global-bundle.p7b
USA Est (Virginie du Nord)	us-east-1-bundle.pem	us-east-1-bundle.p7b
US East (Ohio)	us-east-2-bundle.pem	us-east-2-bundle.p7b
US West (N. California)	us-west-1-bundle.pem	us-west-1-bundle.p7b
US West (Oregon)	us-west-2-bundle.pem	us-west-2-bundle.p7b
Africa (Cape Town)	af-south-1-bundle.pem	af-south-1-bundle.p7b
Asia Pacific (Hong Kong)	ap-east-1-bundle.pem	ap-east-1-bundle.p7b
Asie-Pacifique (Hyderabad)	ap-south-2-bundle.pem	ap-south-2-bundle.p7b
Asie-Pacifique (Jakarta)	ap-southeast-3-bundle.pem	ap-southeast-3-bundle.p7b
Asie-Pacifique (Malaisie)	ap-southeast-5-bundle.pem	ap-southeast-5-bundle.p7b
Asie-Pacifique (Melbourne)	ap-southeast-4-bundle.pem	ap-southeast-4-bundle.p7b
Asia Pacific (Mumbai)	ap-south-1-bundle.pem	ap-south-1-bundle.p7b
Asia Pacific (Osaka)	ap-northeast-3-bundle.pem	ap-northeast-3-bundle.p7b

AWS Région	Solution groupée de certificats (PEM)	Ensemble de certificats (PKCS7)
Asie-Pacifique (Thaïlande)	ap-southeast-7-bundle.pem	ap-southeast-7-bundle.p7b
Asie-Pacifique (Tokyo)	ap-northeast-1-bundle.pem	ap-northeast-1-bundle.p7b
Asia Pacific (Seoul)	ap-northeast-2-bundle.pem	ap-northeast-2-bundle.p7b
Asia Pacific (Singapore)	ap-southeast-1-bundle.pem	ap-southeast-1-bundle.p7b
Asia Pacific (Sydney)	ap-southeast-2-bundle.pem	ap-southeast-2-bundle.p7b
Canada (Centre)	ca-central-1-bundle.pem	ca-central-1-bundle.p7b
Canada-Ouest (Calgary)	ca-west-1-bundle.pem	ca-west-1-bundle.p7b
Europe (Frankfurt)	eu-central-1-bundle.pem	eu-central-1-bundle.p7b
Europe (Ireland)	eu-west-1-bundle.pem	eu-west-1-bundle.p7b
Europe (London)	eu-west-2-bundle.pem	eu-west-2-bundle.p7b
Europe (Milan)	eu-south-1-bundle.pem	eu-south-1-bundle.p7b
Europe (Paris)	eu-west-3-bundle.pem	eu-west-3-bundle.p7b
Europe (Espagne)	eu-south-2-bundle.pem	eu-south-2-bundle.p7b
Europe (Stockholm)	eu-north-1-bundle.pem	eu-north-1-bundle.p7b
Europe (Zurich)	eu-central-2-bundle.pem	eu-central-2-bundle.p7b
Israël (Tel Aviv)	il-central-1-bundle.pem	il-central-1-bundle.p7b
Mexique (Centre)	mx-central-1-bundle.pem	mx-central-1-bundle.p7b
Middle East (Bahrain)	me-south-1-bundle.pem	me-south-1-bundle.p7b
Moyen-Orient (EAU)	me-central-1-bundle.pem	me-central-1-bundle.p7b
Amérique du Sud (São Paulo)	sa-east-1-bundle.pem	sa-east-1-bundle.p7b

AWS Région	Solution groupée de certificats (PEM)	Ensemble de certificats (PKCS7)
N'importe quel AWS GovCloud (US) Region s	global-bundle.pem	global-bundle.p7b
AWS GovCloud (USA Est)	us-gov-east-1-bundle.pem	us-gov-east-1-bundle.p7b
AWS GovCloud (US-Ouest)	us-gov-west-1-bundle.pem	us-gov-west-1-bundle.p7b

Affichage du contenu de votre certificat CA

Pour vérifier le contenu de votre offre groupée de certificats d'autorité de certification, utilisez la commande suivante :

```
keytool -printcert -v -file global-bundle.pem
```

Rotation de votre certificat SSL/TLS

Les certificats de l'autorité de certification Amazon RDS rds-ca-2019 sont configurés pour expirer en août 2024. Si vous utilisez ou prévoyez d'utiliser le protocole SSL (Secure Sockets Layer) ou le protocole Transport Layer Security (TLS) avec vérification des certificats pour vous connecter à vos instances de base de données RDS , pensez à utiliser l'un des nouveaux certificats CA rds-ca-rsa 2048-g1, 4096-g1 ou 384-g1. rds-ca-rsa rds-ca-ecc Si vous ne l'utilisez pas actuellement SSL/TLS avec la vérification des certificats, vous avez peut-être encore un certificat CA expiré et vous devez le mettre à jour SSL/TLS avec un nouveau certificat CA si vous prévoyez d'utiliser la vérification des certificats pour vous connecter à vos bases de données RDS.

Amazon RDS fournit de nouveaux certificats CA dans le cadre des meilleures pratiques AWS de sécurité. Pour plus d'informations sur les nouveaux certificats et les AWS régions prises en charge, consultezUtilisation SSL/TLS pour chiffrer une connexion à une de clusters.

Pour mettre à jour le certificat CA de votre base de données, appliquez les méthodes suivantes :

- Mise à jour de votre certificat CA en modifiant votre instance de base de données
- Mise à jour de votre certificat CA en appliquant la maintenance

Avant de mettre à jour vos instances de base de données pour utiliser le nouveau certificat CA, veillez à mettre à jour vos clients ou applications connectés à vos bases de données RDS.

Considérations relatives à la rotation des certificats

Tenez compte des situations suivantes avant de procéder à la rotation de votre certificat :

- Le proxy et l'utilisation d'Amazon RDS Aurora Serverless v1 les certificats du AWS Certificate Manager (ACM). Si vous utilisez un proxy RDS, lorsque vous faites pivoter votre SSL/TLS certificat, vous n'avez pas besoin de mettre à jour les applications qui utilisent des connexions au proxy RDS. Pour plus d'informations, consultez Utilisation TLS/SSL avec RDS Proxy.
- Si vous utilisez Aurora Serverless v1, le téléchargement des certificats Amazon RDS n'est pas nécessaire. Pour plus d'informations, consultez Utilisation de TLS/SSL avec Aurora Serverless v1.
- Si vous utilisez une application Go version 1.15 avec une instance de base de données créé(e) ou mis(e) à jour vers le certificat rds-ca-2019 avant le 28 juillet 2020, vous devez mettre à jour à nouveau le certificat.

  Utilisez la commande `modify-db-instance` , en utilisant le nouvel identifiant de certificat CA. Vous pouvez trouver ceux CAs qui sont disponibles pour un moteur de base de données et une version de moteur de base de données spécifiques à l'aide de la `describe-db-engine-versions` commande.

  Si vous avez créé votre base de données ou mis à jour son certificat après le 28 juillet 2020, aucune action n'est requise. Pour plus d'informations, consultez Go GitHub issue #39568.

Mise à jour de votre certificat CA en modifiant votre instance de base de données

L'exemple suivant met à jour votre certificat CA de rds-ca-2019 à 2048-g1. rds-ca-rsa Vous pouvez choisir un autre certificat. Pour plus d'informations, consultezAutorités de certification.

Mettez à jour le magasin de confiance de votre application afin de réduire la durée d'indisponibilité associés à la mise à jour de votre certificat CA. Pour plus d'informations sur la rotation des certificats CA, consultez Rotation automatique du certificat de serveur.

Pour mettre à jour votre certificat CA en modifiant votre instance de base de données

1. Téléchargez le nouveau SSL/TLS certificat comme décrit dansUtilisation SSL/TLS pour chiffrer une connexion à une de clusters.
2. Mettez à jour vos applications de sorte à utiliser le nouveau certificat SSL/TLS.

Les méthodes de mise à jour des applications pour SSL/TLS les nouveaux certificats dépendent de vos applications spécifiques. Collaborez avec les développeurs de vos applications pour mettre à jour les SSL/TLS certificats de vos applications.

Pour plus d'informations sur la vérification des SSL/TLS connexions et la mise à jour des applications pour chaque moteur de base de données, consultez les rubriques suivantes :

- Mise à jour des applications pour se connecter aux clusters de bases de données Aurora MySQL à l'aide des nouveaux certificats TLS
- Mise à jour des applications pour se connecter aux clusters de bases de données Aurora PostgreSQL à l'aide des nouveaux certificats SSL/TLS

Pour obtenir un exemple de script qui met à jour le magasin d'approbations d'un système d'exploitation Linux, consultez Exemple de script pour importer les certificats dans votre magasin d'approbations.

> **Note**
>
> L'ensemble de certificats contient des certificats pour le nouveau et l'ancien CA, ce qui signifie que vous pouvez mettre à niveau votre application en toute sécurité et conserver la connectivité pendant la période de transition. Si vous utilisez le AWS Database Migration Service pour migrer une base de données vers une de clusters, nous vous recommandons d'utiliser le bundle de certificats pour garantir la connectivité pendant la migration.

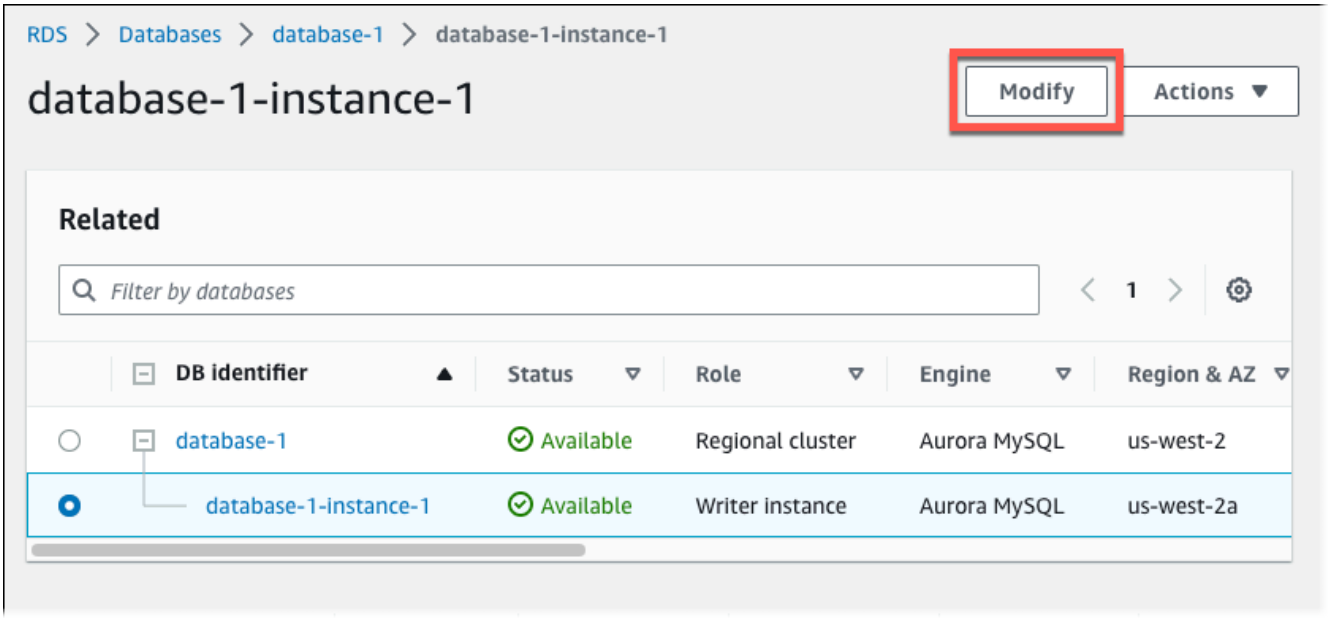
3. Modifiez l'instance de base de données l'autorité de certification de rds-ca-2019 à 2048-g1. rds-ca-rsa Pour vérifier si votre base de données doit être redémarrée pour mettre à jour les certificats CA, utilisez la describe-db-engine-versionscommande et cochez l'`SupportsCertificateRotationWithoutRestart`indicateur.

> **Note**
>
> Redémarrez votre cluster Babelfish après l'avoir modifié pour mettre à jour le certificat CA.

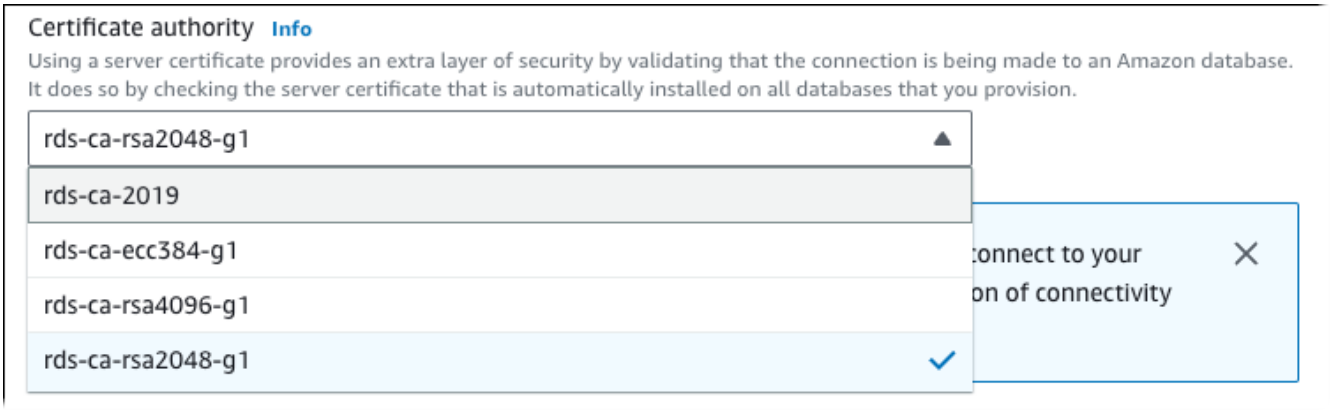
> **Important**
>
> Si vous rencontrez des problèmes de connectivité après l'expiration du certificat, utilisez l'option Appliquer immédiatement en la spécifiant dans la console ou en spécifiant l'option `--apply-immediately` à l'aide d' AWS CLI. Par défaut, il est prévu que cette opération soit exécutée pendant votre prochaine fenêtre de maintenance.
> Pour définir un remplacement pour votre CA de cluster différent de la CA RDS par défaut, utilisez la commande d'interface de ligne de commande modify-certificates.

Console

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Bases de données, puis l'instance de base de données que vous souhaitez modifier.
3. Sélectionnez Modifier.



4. Dans la section Connectivité, choisissez rds-ca-rsa2048-g1.



5. Choisissez Continuer et vérifiez le récapitulatif des modifications.
6. Pour appliquer les modifications immédiatement, choisissez Appliquer immédiatement.
7. Sur la page de confirmation, examinez vos modifications. Si elles sont correctes, choisissez Modifier l'instance de base de données pour enregistrer vos modifications.

> **Important**
>
> Lorsque vous planifiez cette opération, assurez-vous d'avoir mis à jour votre magasin d'approbation côté client au préalable.

Ou choisissez Retour pour revoir vos modifications, ou choisissez Annuler pour les annuler.

AWS CLI

Pour utiliser le pour AWS CLI faire passer l'autorité de certification de rds-ca-2019 à rds-ca-rsa2048-g1 pour une instance de base de données ou un cluster de base de données . modify-db-instancemodify-db-cluster Spécifiez l'identifiant de l'instance de base de données et l'option `--ca-certificate-identifier`.

Pour appliquer la mise à jour immédiatement, utilisez le paramètre `--apply-immediately`. Par défaut, il est prévu que cette opération soit exécutée pendant votre prochaine fenêtre de maintenance.

> **Important**
>
> Lorsque vous planifiez cette opération, assurez-vous d'avoir mis à jour votre magasin d'approbation côté client au préalable.

Example

L'exemple suivant modifie `mydbinstance` en définissant le certificat CA sur `rds-ca-rsa2048-g1`.

Pour Linux, macOS ou Unix :

```
aws rds modify-db-instance \
    --db-instance-identifier mydbinstance \
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

Pour Windows :

```
aws rds modify-db-instance ^
    --db-instance-identifier mydbinstance ^
    --ca-certificate-identifier rds-ca-rsa2048-g1
```

> **Note**
>
> Si votre instance nécessite un redémarrage, vous pouvez utiliser la commande modify-db-instanceCLI et spécifier l'`--no-certificate-rotation-restart`option.

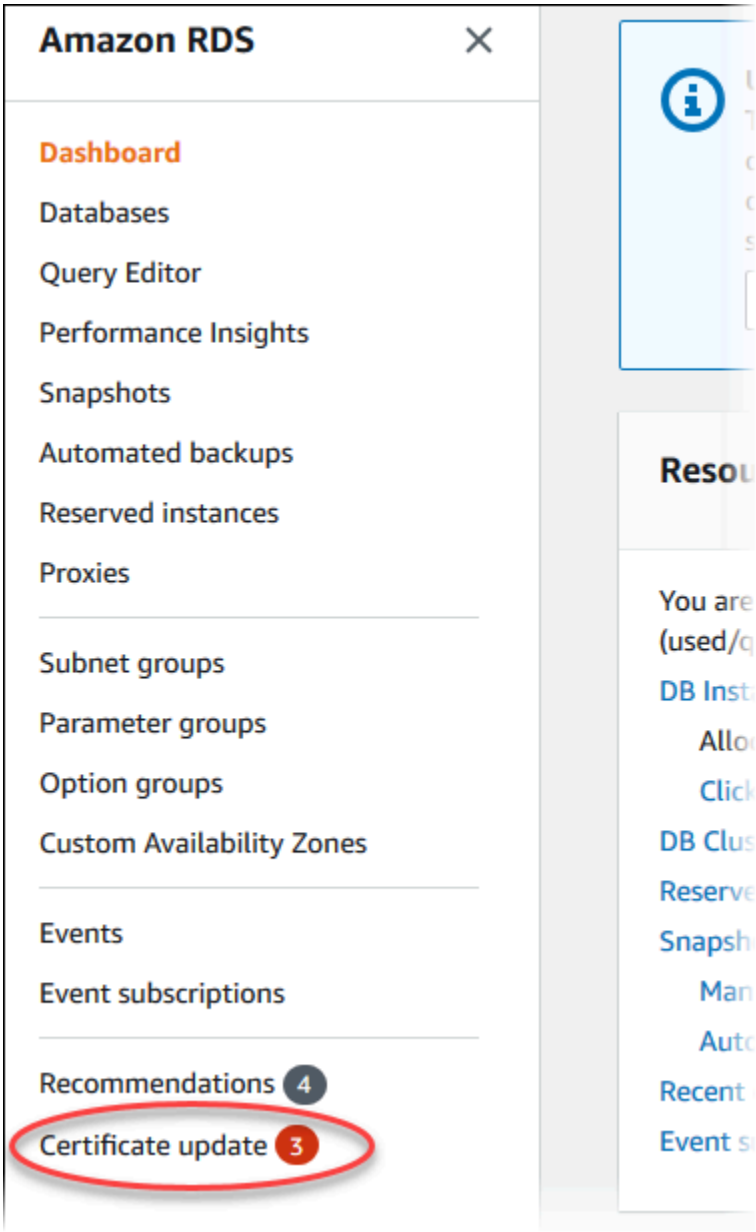
Mise à jour de votre certificat CA en appliquant la maintenance

Procédez comme suit pour mettre à jour votre certificat CA en appliquant la maintenance d'instance de base de données.

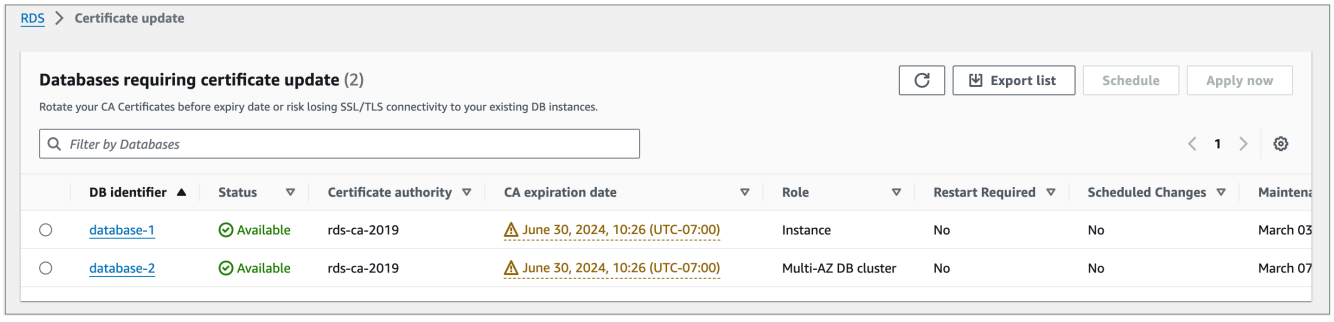
Console

Pour mettre à jour de votre certificat CA en appliquant la maintenance

1. Connectez-vous à la console Amazon RDS AWS Management Console et ouvrez-la à https://console.aws.amazon.com/rds/l'adresse.
2. Dans le panneau de navigation, choisissez Mise à jour du certificat.



La page Bases de données nécessitant une mise à jour de certificat apparaît.



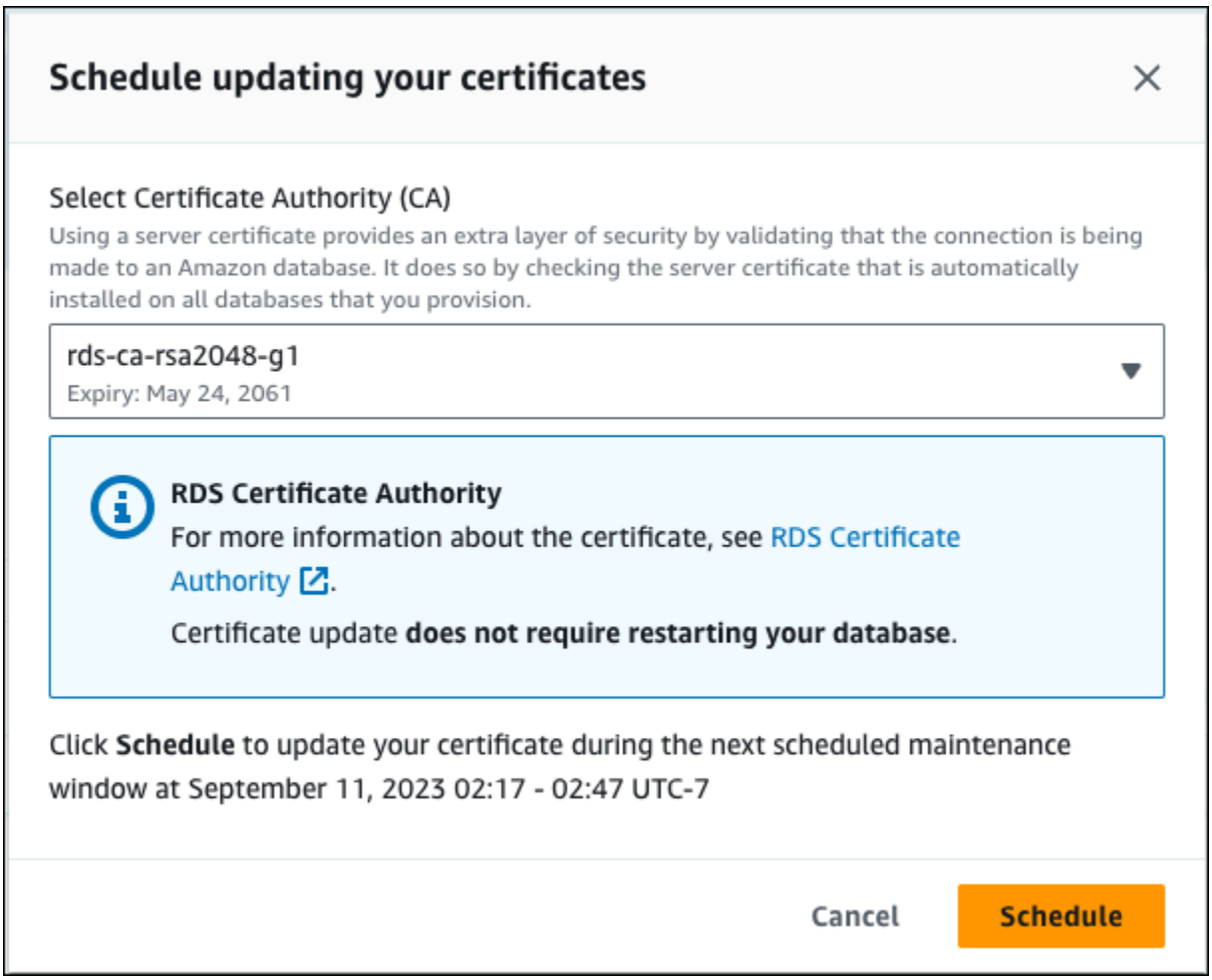
> **Note**
>
> Cette page affiche uniquement les instances de base de données actuels Région AWS. Si vous avez des bases de données dans plusieurs d'entre elles Région AWS, consultez cette page Région AWS pour voir toutes les instances de base de données dotées d'anciens SSL/TLS certificats.

3. Choisissez l'instance de base de données que vous voulez mettre à jour.

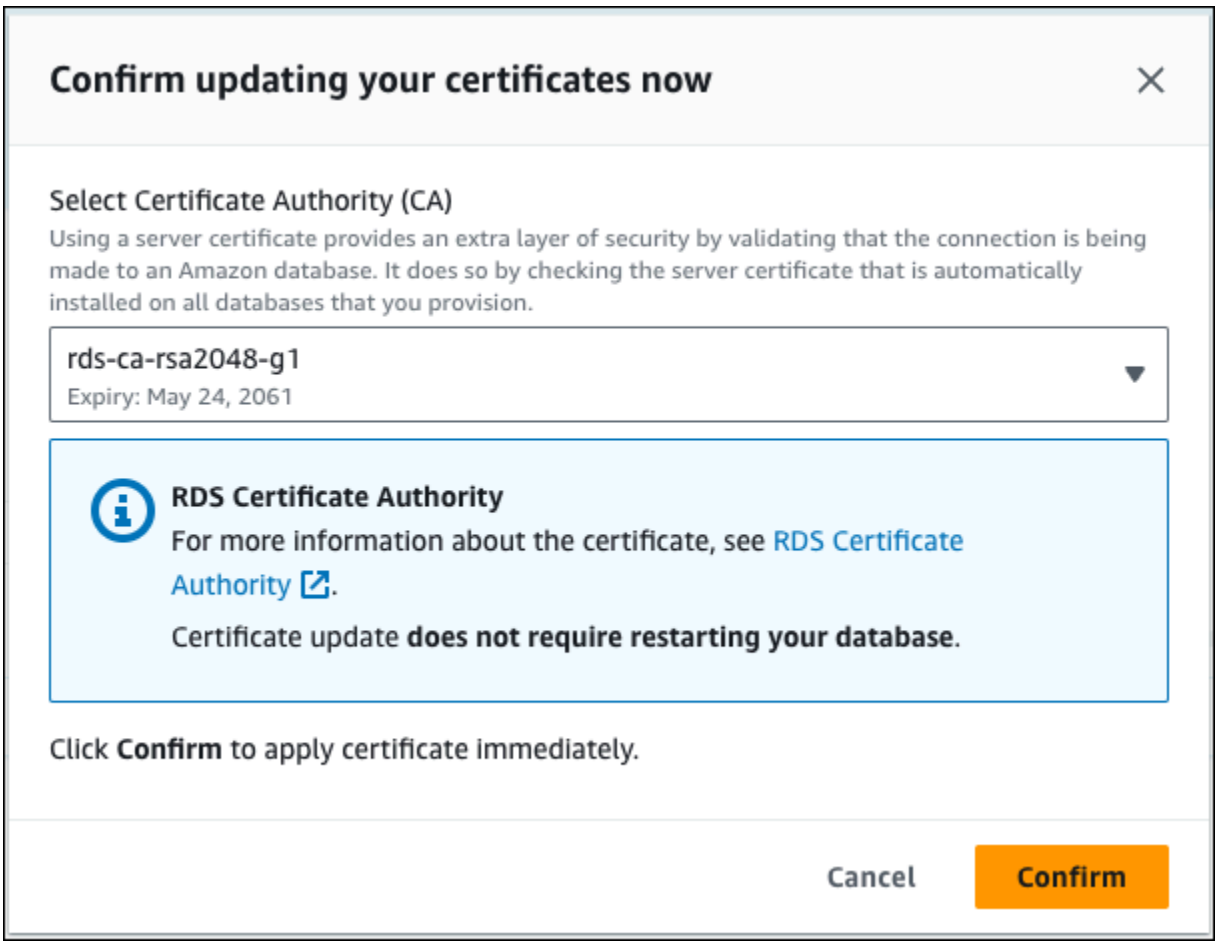
   Vous pouvez planifier la rotation du certificat pour votre prochaine fenêtre de maintenance en choisissant Planification. Appliquez la rotation immédiatement en choisissant Appliquer maintenant.

   > **Important**
   >
   > Si vous rencontrez des problèmes de connectivité après l'expiration du certificat, utilisez l'option Appliquer maintenant.

4. a. Si vous choisissez Planification, vous êtes invité à confirmer la rotation du certificat CA. Cette invite indique également la fenêtre planifiée pour votre mise à jour.



b. Si vous choisissez Appliquer maintenant, vous êtes invité à confirmer la rotation du certificat CA.



> **Important**
>
> Avant de planifier la rotation des certificats CA dans votre base de données, mettez à jour toutes les applications clientes qui les utilisent SSL/TLS et le certificat du serveur pour se connecter. Ces mises à jour sont spécifiques à votre moteur de base de données. Après avoir mis à jour ces applications clientes, vous pouvez confirmer la rotation du certificat CA.

Pour continuer, cochez la case, puis cliquez sur Confirmation.

5. Répétez les étapes 3 et 4 pour chaque instance de base de données que vous souhaitez mettre à jour.

Rotation automatique du certificat de serveur

Si votre CA racine prend en charge la rotation automatique du certificat de serveur, RDS gère automatiquement la rotation du certificat de serveur de base de données. RDS utilise la même autorité de certification racine pour cette rotation automatique. Vous n'avez donc pas besoin de télécharger une nouvelle offre groupée d'autorités de certification. Consultez Autorités de certification.

La rotation et la validité de votre certificat de serveur de base de données dépendent de votre moteur de base de données :

- Si votre moteur de base de données prend en charge la rotation sans redémarrage, RDS effectue automatiquement la rotation du certificat de serveur de base de données sans que vous ayez à intervenir. RDS tente d'effectuer la rotation de votre certificat de serveur de base de données pendant la fenêtre de maintenance de votre choix, à la moitié de la durée de vie du certificat de serveur de base de données. Le nouveau certificat de serveur de base de données est valide pendant 12 mois.
- Si votre moteur de base de données ne prend pas en charge la rotation sans redémarrage, Amazon RDS `server-certificate-rotation` affiche une action de maintenance en attente via l' Describe-pending-maintenance-actionsAPI, à la demi-vie du certificat ou au moins 3 mois avant son expiration. Vous pouvez appliquer la rotation à l'aide de l' apply-pending-maintenance-actionAPI. Le nouveau certificat de serveur de base de données est valide pendant 36 mois.

Utilisez la describe-db-engine-versionscommande et inspectez l'`SupportsCertificateRotationWithoutRestart`indicateur pour déterminer si la version du moteur de base de données prend en charge la rotation du certificat sans redémarrage. Pour plus d'informations, consultez Configuration de l'autorité de certification pour votre base de données.

Exemple de script pour importer les certificats dans votre magasin d'approbations

Voici des exemples de scripts shell qui importent le lot de certificats dans un magasin d'approbations.

Chaque exemple de script shell utilise keytool, qui fait partie du kit de développement Java (JDK). Pour plus d'informations sur l'installation du JDK, consultez le Guide d'installation du JDK.

Linux

Voici un exemple de scripting shell qui importe le lot de certificats vers un magasin d'approbations sur un système d'exploitation Linux.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit

curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
awk 'split_after == 1 {n++;split_after=0} /-----END CERTIFICATE-----/
 {split_after=1}{print > "rds-ca-" n+1 ".pem"}' < ${mydir}/global-bundle.pem

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

macOS

Voici un exemple de scripting shell qui importe le lot de certificats vers un magasin d'approbations sur macOS.

```
mydir=tmp/certs
if [ ! -e "${mydir}" ]
then
mkdir -p "${mydir}"
fi truststore=${mydir}/rds-truststore.jks storepassword=changeit
```

```
curl -sS "https://truststore.pki.rds.amazonaws.com/global/global-bundle.pem">
 ${mydir}/global-bundle.pem
split -p "-----BEGIN CERTIFICATE-----" ${mydir}/global-bundle.pem rds-ca-

for CERT in rds-ca-*; do alias=$(openssl x509 -noout -text -in $CERT | perl -ne
 'next unless /Subject:/; s/.*(CN=|CN = )//; print')
  echo "Importing $alias"
  keytool -import -file ${CERT} -alias "${alias}" -storepass ${storepassword} -
keystore ${truststore} -noprompt
  rm $CERT
done

rm ${mydir}/global-bundle.pem

echo "Trust store content is: "

keytool -list -v -keystore "$truststore" -storepass ${storepassword} | grep Alias |
 cut -d " " -f3- | while read alias
do expiry=`keytool -list -v -keystore "$truststore" -storepass ${storepassword} -
alias "${alias}" | grep Valid | perl -ne 'if(/until: (.*?)\n/) { print "$1\n"; }'`
   echo " Certificate ${alias} expires in '$expiry'"
done
```

Pour en savoir plus sur les bonnes pratiques relatives à l'utilisation du protocole SSL avec Amazon RDS, consultez Bonnes pratiques pour des connexions SSL réussies à Amazon RDS for Oracle.

Confidentialité du trafic inter-réseau

Les connexions sont protégées entre Amazon Aurora et les applications sur site, ainsi qu'entre Amazon Aurora et d'autres ressources AWS dans la même Région AWS.

Trafic entre les clients de service et sur site et les applications

Vous disposez de deux options de connectivité entre votre réseau privé et :AWS

- Une connexion AWS Site-to-Site VPN. Pour plus d'informations, consultez Qu'est-ce qu'AWS Site-to-Site VPN ?
- Une connexion Direct Connect. Pour plus d'informations, consultez Qu'est-ce qu'Direct Connect ?

Vous accédez à Amazon Aurora via le réseau en utilisant des opérations d'API publiées par AWS. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Identity and Access Management pour Amazon Aurora

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Des administrateurs IAM contrôlent les personnes qui peuvent être authentifiées (connectées) et autorisées (disposant d'autorisations) à utiliser des ressources Amazon RDS. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction du travail que vous effectuez dans).

Utilisateur du service – Si vous utilisez le service Aurora pour effectuer votre tâche, votre administrateur vous fournit les informations d'identification et les autorisations dont vous avez besoin. Plus vous utiliserez de fonctionnalités Aurora pour effectuer votre travail, plus vous pourrez avoir besoin d'autorisations supplémentaires. Comprendre la gestion des accès peut vous aider à demander à votre administrateur les autorisations appropriées. Si vous ne pouvez pas accéder à une fonctionnalité dans Aurora, consultez Résolution des problèmes liés à Identity and Access Amazon Aurora.

Administrateur du service – Si vous êtes le responsable des ressources Aurora de votre entreprise, vous bénéficiez probablement d'un accès total à Aurora. C'est à vous de déterminer les fonctionnalités et les ressources Aurora auxquelles vos employés pourront accéder. Vous devez ensuite soumettre les demandes à votre administrateur pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour en savoir plus sur la façon dont votre entreprise peut utiliser IAM avec Aurora, consultez Comment Amazon Aurora fonctionne avec IAM.

Administrateur : si vous êtes un administrateur, vous souhaiterez peut-être obtenir des détails sur la façon dont vous pouvez écrire des politiques pour gérer l'accès à Aurora. Pour obtenir des exemples de stratégies Aurora basées sur l'identité que vous pouvez utiliser dans IAM, consultez Exemples de politiques basées sur l'identité pour Amazon Aurora.

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une authentification unique ou d'informations d'identification. Google/Facebook Pour plus d'informations sur la connexion, consultez Connexion à votre Compte AWS dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez Signature AWS Version 4 pour les demandes d'API dans le Guide de l'utilisateur IAM.

AWS utilisateur root du compte

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez Tâches qui requièrent les informations d'identification de l'utilisateur racine dans le Guide de l'utilisateur IAM.

Identité fédérée

Il est recommandé d'obliger les utilisateurs humains à utiliser la fédération avec un fournisseur d'identité pour accéder à Services AWS l'aide d'informations d'identification temporaires.

Une identité fédérée est un utilisateur provenant de l'annuaire de votre entreprise, de votre fournisseur d'identité Web ou Directory Service qui y accède à Services AWS l'aide d'informations d'identification provenant d'une source d'identité. Les identités fédérées assument des rôles qui fournissent des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Pour plus d'informations, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center .

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires dans le guide de l'utilisateur IAM.

Les groupes IAM spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez la section Cas d'utilisation pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Vous pouvez vous authentifier auprès de votre cluster de bases de données à l'aide de l'authentification de base de données IAM.

L'authentification de base de données IAM fonctionne avec Aurora.Pour plus d'informations sur l'authentification auprès de votre cluster de bases de données avec IAM, consultez Authentification de base de données IAM.

Rôles IAM

Un rôle IAM est une identité au sein de vous Compte AWS dotée d'autorisations spécifiques.
Le concept ressemble à celui d'un utilisateur, mais un rôle n'est pas associé à une personne en particulier. Vous pouvez assumer temporairement un rôle IAM dans le en AWS Management Console changeant de rôle. Vous pouvez assumer un rôle en appelant une opération d' AWS API AWS CLI ou en utilisant une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Utilisation de rôles IAM dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Autorisations utilisateur temporaires : un utilisateur peut endosser un rôle IAM pour accepter différentes autorisations temporaires concernant une tâche spécifique.
- Accès utilisateur fédéré : pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center .
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, dans certains Services AWS cas, vous pouvez associer une politique directement à une ressource (au lieu d'utiliser un rôle comme proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.
- Accès multiservices — Certains Services AWS utilisent des fonctionnalités dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, un rôle de service ou un rôle lié au service.
  - Sessions d'accès par transfert : les sessions d'accès par transfert (FAS) utilisent les autorisations du principal appelant et Service AWS, associées à la demande, Service AWS pour adresser des demandes aux services en aval. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d'accès.
  - Rôle de service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.
  - Rôle lié à un service — Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service apparaissent dans votre Compte AWS fichier et appartiennent au service. Un

administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

- Applications exécutées sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer les informations d'identification temporaires pour les applications qui s'exécutent sur une EC2 instance et qui envoient des demandes AWS CLI d' AWS API. Cela est préférable au stockage des clés d'accès dans l' EC2 instance. Pour attribuer un AWS rôle à une EC2 instance et le rendre disponible pour toutes ses applications, vous devez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes exécutés sur l' EC2 instance d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez Utiliser un rôle IAM pour accorder des autorisations aux applications exécutées sur des EC2 instances Amazon dans le guide de l'utilisateur IAM.

Pour savoir si vous devez utiliser ces rôles IAM ou non, consultez Quand créer un rôle IAM (au lieu d’un utilisateur) dans le Guide de l’utilisateur IAM.

Gestion de l’accès à l’aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant aux identités ou aux AWS ressources IAM. Une politique est un objet AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit leurs autorisations. AWS évalue ces politiques lorsqu'une entité (utilisateur root, utilisateur ou rôle IAM) fait une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d’informations sur la structure et le contenu des documents de politique JSON, consultez Vue d’ensemble des politiques JSON dans le Guide de l’utilisateur IAM.

Un administrateur peut utiliser des politiques pour spécifier qui a accès aux AWS ressources et quelles actions il peut effectuer sur ces ressources. Chaque entité IAM (jeu d’autorisations ou rôle) démarre sans autorisation. En d’autres termes, par défaut, les utilisateurs ne peuvent rien faire, pas même changer leurs propres mots de passe. Pour autoriser un utilisateur à effectuer une opération, un administrateur doit associer une politique d’autorisations à ce dernier. Il peut également ajouter l’utilisateur à un groupe disposant des autorisations prévues. Lorsqu’un administrateur accorde des autorisations à un groupe, tous les utilisateurs de ce groupe se voient octroyer ces autorisations.

Les politiques IAM définissent les autorisations d’une action, quelle que soit la méthode que vous utilisez pour exécuter l’opération. Par exemple, supposons que vous disposiez d’une politique qui autorise l’action `iam:GetRole`. Un utilisateur appliquant cette politique peut obtenir des informations sur le rôle à partir de AWS Management Console AWS CLI, de ou de l' AWS API.

Politiques basées sur l’identité

Les politiques basées sur l’identité sont des documents de politiques d’autorisations JSON que vous pouvez attacher à une identité telle qu’un jeu d’autorisations ou un rôle. Ces politiques contrôlent les actions que peut exécuter cette identité, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l’identité, consultez Création de politiques IAM dans le Guide de l’utilisateur IAM.

Les politiques basées sur l’identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un seul jeu d’autorisations ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez associer à plusieurs ensembles d'autorisations et rôles dans votre AWS compte. Les politiques gérées incluent les politiques AWS gérées et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l’utilisateur IAM.

Pour plus d'informations sur les politiques AWS gérées spécifiques à (Amazon Aurora), consultezPolitiques AWS gérées pour Amazon RDS.

Autres types de politique

AWS prend en charge d'autres types de politiques moins courants. Ces types de politiques peuvent définir le nombre maximum d’autorisations qui vous sont accordées par des types de politiques plus courants.

- Limites des autorisations : une limite des autorisations est une fonction avancée dans laquelle vous définissez le nombre maximal d’autorisations qu’une politique basée sur l’identité peut accorder à une entité IAM (jeu d’autorisations ou rôle). Vous pouvez définir une limite d’autorisations pour une entité. Les autorisations obtenues représentent la combinaison des politiques basées sur l’identité de l’entité et de ses limites d’autorisations. Les politiques basées sur les ressources qui spécifient le jeu d’autorisations ou le rôle dans le champ `Principal` ne sont pas limitées par les limites des autorisations. Un refus explicite dans l’une de ces politiques annule l’autorisation. Pour plus d’informations sur les limites d’autorisations, consultez Limites d’autorisations pour des entités IAM dans le Guide de l’utilisateur IAM.
- Politiques de contrôle des services (SCPs) : SCPs politiques JSON qui spécifient les autorisations maximales pour une organisation ou une unité organisationnelle (UO) dans AWS Organizations. AWS Organizations est un service permettant de regrouper et de gérer de manière centralisée plusieurs AWS comptes détenus par votre entreprise. Si vous activez toutes les fonctionnalités d'une organisation, vous pouvez appliquer des politiques de contrôle des services (SCPs) à l'un

ou à l'ensemble de vos comptes. Le SCP limite les autorisations pour les entités figurant dans les comptes des membres, y compris chacune Utilisateur racine d'un compte AWS d'entre elles. Pour plus d'informations sur les Organizations SCPs, voir Comment SCPs travailler dans le Guide de AWS Organizations l'utilisateur.

- Politiques de session : les politiques de session sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de la session obtenues sont une combinaison des politiques basées sur l'identité du rôle ou des jeux d'autorisations et des politiques de session. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques annule l'autorisation. Pour plus d'informations, consultez Politiques de session dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section Logique d'évaluation des politiques dans le guide de l'utilisateur IAM.

Comment Amazon Aurora fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à Amazon Aurora, vous devez comprendre quelles sont les fonctions IAM disponibles à utiliser avec Aurora.

Le tableau suivant répertorie les fonctionnalités IAM que vous pouvez utiliser avec Amazon Aurora :

Fonctionnalité IAM	Prise en charge d'Amazon Aurora
Politiques basées sur l'identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition de politique (spécifiques au service)	Oui

Fonctionnalité IAM	Prise en charge d’Amazon Aurora
ACLs	Non
Contrôle d’accès par attributs (ABAC) (balises dans les politiques)	Oui
Informations d’identification temporaires	Oui
Transfert des sessions d’accès	Oui
Rôles de service	Oui
Rôles liés à un service	Oui

Pour obtenir une vue d'ensemble de la manière dont , Amazon Aurora et d'autres AWS services fonctionnent avec IAM, consultez la section sur les AWS services compatibles avec IAM dans le guide de l'utilisateur d'IAM.

Rubriques

- Stratégies basées sur l’identité Aurora
- Politiques basées sur les ressources au sein d’Aurora
- Actions de politique pour Aurora
- Ressources de politique pour Aurora
- Clés de condition de politique pour Aurora
- Listes de contrôle d'accès (ACLs) dans
- Contrôle d’accès par attributs (ABAC) dans les politiques avec des balises Aurora
- Utilisation des informations d’identification temporaires avec Aurora
- Transfert des sessions d’accès pour Aurora
- Rôles de service pour Aurora
- Rôles liés à un service pour Aurora

Stratégies basées sur l’identité Aurora

Prend en charge les politiques basées sur l’identité : oui.

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Définition d'autorisations IAM personnalisées avec des politiques gérées par le client dans le Guide de l'utilisateur IAM.

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Pour découvrir tous les éléments que vous utilisez dans une politique JSON, consultez Références des éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

Exemples de politiques basées sur l'identité pour Aurora

Pour voir des exemples de stratégies Aurora basées sur l'identité, consultez Exemples de politiques basées sur l'identité pour Amazon Aurora.

Politiques basées sur les ressources au sein d'Aurora

Prend en charge les politiques basées sur les ressources : non.

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Par exemple, les politiques de confiance de rôle IAM et les politiques de compartiment Amazon S3 sont des politiques basées sur les ressources. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou. Services AWS

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. Pour plus d'informations, consultez Accès intercompte aux ressources dans IAM dans le Guide de l'utilisateur IAM.

Actions de politique pour Aurora

Prend en charge les actions de politique : oui.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Les actions de stratégie dans Aurora utilisent le préfixe suivant avant l'action : `rds:`. Par exemple, pour accorder à une personne l'autorisation de décrire les instances de base de données à l'aide de l'opération d'API Amazon RDS`DescribeDBInstances` , vous incluez l'action `rds:DescribeDBInstances` dans sa stratégie. Les déclarations de politique doivent inclure un élément `Action` ou `NotAction`. Aurora définit son propre ensemble d'actions qui décrivent les tâches que vous pouvez effectuer avec ce service.

Pour spécifier plusieurs actions dans une seule instruction, séparez-les par des virgules, comme suit :

```
"Action": [
      "rds:action1",
      "rds:action2"
```

Vous pouvez aussi préciser plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l'action suivante.

```
"Action": "rds:Describe*"
```

Pour afficher la liste des actions Aurora, consultez Actions définies par Amazon RDS dans Référence de l'autorisation de service.

Ressources de politique pour Aurora

Prend en charge les ressources de politique : oui.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Pour les

actions qui ne sont pas compatibles avec les autorisations de niveau ressource, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

La ressource d'instance de base de données possède l'ARN (Amazon Resource Name) suivant.

```
arn:${Partition}:rds:${Region}:${Account}:{ResourceType}/${Resource}
```

Pour plus d'informations sur le format de ARNs, consultez Amazon Resource Names (ARNs) et espaces de noms AWS de services.

Par exemple, pour spécifier l'instance de base de données `dbtest` dans votre instruction, utilisez l'ARN suivant.

```
"Resource": "arn:aws:rds:us-west-2:123456789012:db:dbtest"
```

Pour spécifier toutes les instances de base de données qui appartiennent à un compte spécifique, utilisez le caractère générique (*).

```
"Resource": "arn:aws:rds:us-east-1:123456789012:db:*"
```

Certaines opérations d'API RDS, telles que la création de ressources, ne peuvent pas être exécutées sur une ressource spécifique. Dans ces cas-là, utilisez le caractère générique (*).

```
"Resource": "*"
```

De nombreuses opérations d'API Amazon RDS nécessitent plusieurs ressources. Par exemple, `CreateDBInstance` crée une instance de base de données. Vous pouvez spécifier qu'un utilisateur doit utiliser un groupe de sécurité spécifique et un groupe de paramètres lors de la création d'une instance de base de données. Pour spécifier plusieurs ressources dans une seule instruction, séparez-les ARNs par des virgules.

```
"Resource": [
      "resource1",
      "resource2"
```

Pour consulter la liste des types de ressources Aurora et leurs caractéristiques ARNs, consultez la section Ressources définies par Amazon RDS dans le Service Authorization Reference. Pour savoir

les actions avec lesquelles vous pouvez spécifier l'ARN de chaque ressource, consultez Actions définies par Amazon RDS.

Clés de condition de politique pour Aurora

Prend en charge les clés de condition de politique spécifiques au service : oui.

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Condition` indique à quel moment les instructions sont exécutées en fonction de critères définis. Vous pouvez créer des expressions conditionnelles qui utilisent des opérateurs de condition, tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

Aurora définit son propre ensemble de clés de condition et prend également en charge l'utilisation des clés de condition globales. Pour voir toutes les clés de condition AWS globales, voir les clés de contexte de condition AWS globales dans le guide de l'utilisateur IAM.

Toutes les opérations d'API RDS prennent en charge la clé de condition `aws:RequestedRegion`.

Pour afficher la liste des clés de condition Aurora, consultez Clés de condition pour Amazon RDS dans la Référence de l'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez Actions définies par Amazon RDS.

Listes de contrôle d'accès (ACLs) dans

Supporte les listes de contrôle d'accès (ACLs) : Non

Les listes de contrôle d'accès (ACLs) contrôlent les principaux (membres du compte, utilisateurs ou rôles) autorisés à accéder à une ressource. ACLs sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Contrôle d'accès par attributs (ABAC) dans les politiques avec des balises Aurora

Prend en charge les balises dans les politiques pour le contrôle d'accès par attributs (ABAC) : oui

Le contrôle d'accès par attributs (ABAC) est une stratégie d'autorisation qui définit les autorisations en fonction des attributs nommés balise. Vous pouvez associer des balises aux entités et aux AWS

ressources IAM, puis concevoir des politiques ABAC pour autoriser les opérations lorsque la balise du principal correspond à la balise de la ressource.

Pour contrôler l'accès basé sur des étiquettes, vous devez fournir les informations d'étiquette dans l'élément de condition d'une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d'informations sur ABAC, consultez Définition d'autorisations avec l'autorisation ABAC dans le Guide de l'utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez Utilisation du contrôle d'accès par attributs (ABAC) dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur le balisage des ressources Aurora, consultez Spécification de conditions : Utilisation de balises personnalisées. Pour visualiser un exemple de politique basée sur l'identité permettant de limiter l'accès à une ressource en fonction des balises de cette ressource, consultez Accorder une autorisation pour des actions sur une ressource à l'aide d'une balise spécifique avec deux valeurs différentes.

Utilisation des informations d'identification temporaires avec Aurora

Prend en charge les informations d'identification temporaires : oui.

Les informations d'identification temporaires fournissent un accès à court terme aux AWS ressources et sont automatiquement créées lorsque vous utilisez la fédération ou que vous changez de rôle. AWS recommande de générer dynamiquement des informations d'identification temporaires au lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez la section Informations d'identification de sécurité temporaires dans IAM et Services AWS compatibles avec IAM dans le Guide de l'utilisateur IAM.

Transfert des sessions d'accès pour Aurora

Prend en charge le transfert des sessions d'accès : oui.

Les sessions d'accès direct (FAS) utilisent les autorisations du principal appelant et Service AWS, combinées Service AWS à la demande d'envoi de demandes aux services en aval. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transfert des sessions d'accès.

Rôles de service pour Aurora

Prend en charge les rôles de service : oui.

Un rôle de service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer un rôle de service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

> **Warning**
>
> La modification des autorisations d'un rôle de service peut altérer la fonctionnalité d'Aurora. Ne modifiez des rôles de service que quand Aurora vous le conseille.

Rôles liés à un service pour Aurora

Prend en charge les rôles liés à un service : oui.

Un rôle lié à un service est un type de rôle de service lié à un. Service AWS Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés au service apparaissent dans votre Compte AWS fichier et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus d'informations l'utilisation des rôles liés à un service Aurora, consultez Utilisation des rôles liés à un service pour Amazon Aurora.

Exemples de politiques basées sur l'identité pour Amazon Aurora

Par défaut, les jeux d'autorisations et les rôles ne sont pas autorisés à créer ou modifier des ressources Aurora. Ils ne peuvent pas non plus effectuer de tâches à l'aide de l' AWS API AWS Management Console AWS CLI, ou. Un administrateur doit créer des politiques IAM autorisant les jeux d'autorisations et les rôles à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. L'administrateur doit ensuite attacher ces politiques aux jeux d'autorisations et aux rôles qui ont besoin de ces autorisations.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques dans l'onglet JSON dans le Guide de l'utilisateur IAM.

Rubriques

Bonnes pratiques en matière de politiques

Les stratégies basées sur l’identité déterminent si une personne peut créer, consulter ou supprimer des ressources Amazon RDS dans votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l’identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d’informations, consultez politiques gérées par AWS ou politiques gérées par AWS pour les activités professionnelles dans le Guide de l’utilisateur IAM.
- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l’exécution d’une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d’informations sur l’utilisation d’IAM pour appliquer des autorisations, consultez politiques et autorisations dans IAM dans le Guide de l’utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l’accès : vous pouvez ajouter une condition à vos politiques afin de limiter l’accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service

AWS, tel que CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez Validation de politiques avec IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Sécurisation de l'accès aux API avec MFA dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez Bonnes pratiques de sécurité dans IAM dans le Guide de l'utilisateur IAM.

Utilisation de la console Aurora

Pour accéder à la console Amazon Aurora, vous devez disposer d'un ensemble minimal d'autorisations. Ces autorisations doivent vous permettre de répertorier et de consulter les informations relatives aux ressources Amazon Aurora présentes dans votre Compte AWS. Si vous créez une politique basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette politique.

Il n'est pas nécessaire d'accorder des autorisations de console minimales aux utilisateurs qui appellent uniquement l'API AWS CLI ou l' AWS API. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API que vous tentez d'effectuer.

Pour garantir que ces entités peuvent toujours utiliser la console Aurora, associez également la politique AWS gérée suivante aux entités.

```
AmazonRDSReadOnlyAccess
```

Pour plus d'informations, consultez Ajout d'autorisations à un utilisateur dans le Guide de l'utilisateur IAM.

Autorisations requises pour utiliser la console

Pour qu'un utilisateur puisse utiliser la console, il doit avoir un ensemble minimal d'autorisations. Ces autorisations permettent à l'utilisateur de décrire les ressources Amazon Aurora associées à son AWS compte et de fournir d'autres informations connexes, notamment des informations relatives à EC2 la sécurité et au réseau Amazon.

Si vous créez une politique IAM plus restrictive que les autorisations minimales requises, la console ne fonctionne pas comme prévu pour les utilisateurs dotés de cette politique IAM. Pour garantir que ces utilisateurs puissent continuer à utiliser la console, attachez également la stratégie gérée `AmazonRDSReadOnlyAccess` à l'utilisateur, comme décrit dans Gestion de l'accès à l'aide de politiques.

Vous n'avez pas besoin d'accorder d'autorisations minimales d'utilisation de la console aux utilisateurs qui effectuent des appels uniquement à l' AWS CLI ou à l'API Amazon RDS.

La politique suivante accorde un accès complet à toutes les ressources Amazon Aurora pour le AWS compte racine :

```
AmazonRDSFullAccess
```

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
```

```
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Politiques d’autorisation pour créer, modifier et supprimer des ressources dans Aurora

Les sections suivantes présentent des exemples de politiques d’autorisation qui accordent et restreignent l’accès aux ressources :

Autoriser un utilisateur à créer des instances de base de données dans un AWS compte

Voici un exemple de politique qui permet au compte doté de l'ID de `123456789012` créer des instances de base de données pour votre AWS compte. La stratégie exige que le nom de la nouvelle instance de base de données commence par `test`. La nouvelle instance de base de données doit également utiliser le moteur de base de données MySQL et la classe d’instance de base de données `db.t2.micro`. En outre, la nouvelle instance de base de données doit utiliser un groupe d’options et un groupe de paramètres de base de données commençant par `default`, et elle doit utiliser le groupe de sous-réseaux `default`.

JSON

```
{
   "Version":"2012-10-17",
```

```
   "Statement": [
      {
         "Sid": "AllowCreateDBInstanceOnly",
         "Effect": "Allow",
         "Action": [
            "rds:CreateDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:db:test*",
            "arn:aws:rds:*:123456789012:og:default*",
            "arn:aws:rds:*:123456789012:pg:default*",
            "arn:aws:rds:*:123456789012:subgrp:default"
         ],
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql",
               "rds:DatabaseClass": "db.t2.micro"
            }
         }
      }
   ]
}
```

La stratégie inclut une instruction unique spécifiant les autorisations suivantes pour l'utilisateur :

- La politique permet au compte de créer une instance de base de données à l'aide de l'opération Create DBInstance API (cela s'applique également à la create-db-instanceAWS CLIcommande et auAWS Management Console).
- L'élément `Resource` spécifie que l'utilisateur peut effectuer des actions sur et avec des ressources. Vous indiquez des ressources à l'aide d'un nom ARN (Amazon Resources Name). Cet ARN inclut le nom du service auquel appartient la ressource (`rds`), la AWS région (*indique n'importe quelle région dans cet exemple), le numéro de AWS compte (`123456789012`il s'agit du numéro de compte dans cet exemple) et le type de ressource. Pour plus d'informations sur la création ARNs, consultezAmazon Resource Names (ARN) dans Amazon RDS.

  L'élément `Resource` dans l'exemple spécifie les contraintes de stratégie suivantes sur les ressources de l'utilisateur :

  - L'identifiant d'instance de base de données de la nouvelle instance de base de données doit commencer par `test` (par exemple, `testCustomerData1`, `test-region2-data`).

- Le groupe d'options de la nouvelle instance de base de données doit commencer par `default`.
- Le groupe de paramètres de base de données de la nouvelle instance de base de données doit commencer par `default`.
- Le groupe de sous-réseaux de la nouvelle instance de base de données doit être le groupe de sous-réseaux `default`.

- L'élément `Condition` indique que le moteur de base de données doit être MySQL et la classe d'instance de base de données doit être `db.t2.micro`. L'élément `Condition` indique les conditions lorsqu'une stratégie doit entrer en vigueur. Vous pouvez ajouter des autorisations ou des restrictions supplémentaires à l'aide de l'élément `Condition`. Pour plus d'informations sur la spécification de conditions, consultez Clés de condition de politique pour Aurora. Cet exemple spécifie les conditions `rds:DatabaseEngine` et `rds:DatabaseClass`. Pour plus d'informations sur les valeurs de condition valides pour`rds:DatabaseEngine`, consultez la liste sous le `Engine` paramètre dans Create DBInstance. Pour plus d'informations sur les valeurs de conditions valides pour `rds:DatabaseClass`, consultez Moteurs de base de données pris en charge pour les classes d'instance de base de données.

La politique ne spécifie pas l'élément `Principal` car, dans une politique basée sur une identité, vous ne spécifiez pas le principal qui obtient l'autorisation. Quand vous attachez une politique à un utilisateur, l'utilisateur est le principal implicite. Lorsque vous attachez une politique d'autorisation à un rôle IAM, le principal identifié dans la politique d'approbation de ce rôle obtient les autorisations.

Pour afficher la liste des actions Aurora, consultez Actions définies par Amazon RDS dans Référence de l'autorisation de service.

Autoriser un utilisateur à effectuer une action Describe sur une ressource RDS

La politique d'autorisation suivante accorde des autorisations à un utilisateur lui permettant d'exécuter toutes les actions commençant par `Describe`. Ces actions affichent des informations sur une ressource RDS, telle qu'une instance de base de données. Le caractère générique (*) figurant dans l'élément `Resource` indique que les actions sont autorisées pour toutes les ressources Amazon Aurora détenues par le compte.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
```

```
      {
         "Sid": "AllowRDSDescribe",
         "Effect": "Allow",
         "Action": "rds:Describe*",
         "Resource": "*"
      }
   ]
}
```

Autoriser un utilisateur à créer une instance de base de données qui utilise le groupe de paramètres de base de données et le groupe de sous-réseau spécifiés

La politique d'autorisation suivante accorde des autorisations permettant à un utilisateur de créer uniquement une instance de base de données devant utiliser le groupe de paramètres de base de données `mydbpg` et le groupe de sous-réseau de base de données `mydbsubnetgroup`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": [
            "arn:aws:rds:*:*:pg:mydbpg",
            "arn:aws:rds:*:*:subgrp:mydbsubnetgroup"
         ]
      }
   ]
}
```

Accorder une autorisation pour des actions sur une ressource à l'aide d'une balise spécifique avec deux valeurs différentes

Vous pouvez utiliser des conditions dans votre stratégie basée sur l'identité pour contrôler l'accès aux ressources Aurora en fonction des balises. La politique suivante accorde l'autorisation d'exécuter

l'opération d'API `CreateDBSnapshot` sur les instances de base de données avec la balise `stage` définie sur `development` ou `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

La politique suivante accorde l'autorisation d'exécuter l'opération d'API `ModifyDBInstance` sur les instances de base de données avec la balise `stage` définie sur `development` ou `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
         ]
      },
      {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

Empêcher un utilisateur de supprimer une instance de base de données

La politique d’autorisation suivante accorde des autorisations empêchant un utilisateur de supprimer une instance de base de données spécifique. Par exemple, il est possible de refuser la capacité à supprimer vos instances de base de données de production à un utilisateur quelconque qui n’est pas un administrateur.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyDelete1",
         "Effect": "Deny",
         "Action": "rds:DeleteDBInstance",
         "Resource": "arn:aws:rds:us-west-2:123456789012:db:my-mysql-instance"
      }
   ]
}
```

Refuser tout accès à une ressource

Vous pouvez refuser explicitement l’accès à une ressource. Les politiques de refus ont priorité sur les politiques d’autorisation. La politique suivante refuse explicitement à un utilisateur la possibilité de gérer une ressource :

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Deny",
         "Action": "rds:*",
         "Resource": "arn:aws:rds:us-east-1:123456789012:db:mydb"
      }
   ]
}
```

Exemples de politiques : Utilisation des clés de condition

Les exemples suivants montrent comment vous pouvez utiliser des clés de condition dans les stratégies d'autorisation IAM Amazon Aurora.

Exemple 1 : Accorder l'autorisation de créer une instance de base de données qui utilise un moteur de base de données spécifique et n'est pas Multi-AZ

La politique suivante utilise une clé de condition RDS et autorise un utilisateur à créer seulement des instances de bases de données qui utilisent le moteur de base de données MySQL et n'utilisent pas la configuration Multi-AZ. L'élément `Condition` indique l'exigence que le moteur de base de données soit MySQL.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "AllowMySQLCreate",
         "Effect": "Allow",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseEngine": "mysql"
            },
            "Bool": {
               "rds:MultiAz": false
            }
         }
      }
   ]
}
```

Exemple 2 : Refuser explicitement l'autorisation de créer des instances de bases de données pour certaines classes d'instance de base de données et de créer des instances de bases de données qui utilisent les IOPS provisionnées

La stratégie suivante refuse explicitement l'autorisation de créer des instances de bases de données qui utilisent les classes d'instance de base de données `r3.8xlarge` et `m4.10xlarge`, lesquelles représentent les classes d'instances de base de données les plus grandes et les plus onéreuses. Cette politique empêche également les utilisateurs de créer des instances de bases de données qui utilisent les IOPS provisionnées, ce qui génère un coût additionnel.

Le refus explicite d'une autorisation a priorité sur toutes les autres autorisations accordées. Cela garantit que des identités n'obtiendront pas par erreur une autorisation que vous ne souhaitez pas accorder.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyLargeCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "StringEquals": {
               "rds:DatabaseClass": [
                  "db.r3.8xlarge",
                  "db.m4.10xlarge"
               ]
            }
         }
      },
      {
         "Sid": "DenyPIOPSCreate",
         "Effect": "Deny",
         "Action": "rds:CreateDBInstance",
         "Resource": "*",
         "Condition": {
            "NumericNotEquals": {
               "rds:Piops": "0"
```

```
            }
         }
      }
   ]
}
```

Exemple 3 : Limiter l'ensemble de clés et de valeurs de balise pouvant être utilisées pour baliser une ressource

La politique suivante utilise une clé de condition RDS et autorise l'ajout d'une balise avec la clé `stage` à une ressource avec les valeurs `test`, `qa` et `production`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTagEdits",
      "Effect": "Allow",
      "Action": [
        "rds:AddTagsToResource",
        "rds:RemoveTagsFromResource"
      ],
      "Resource": "arn:aws:rds:us-east-1:123456789012:db:db-123456",
      "Condition": {
        "StringEquals": {
          "rds:req-tag/stage": [
            "test",
            "qa",
            "production"
          ]
        }
      }
    }
  ]
}
```

Spécification de conditions : Utilisation de balises personnalisées

Amazon Aurora prend en charge la spécification de conditions dans une stratégie IAM à l'aide de balises personnalisées.

Par exemple, supposons que vous ajoutiez une balise nommée `environment` à vos instances de base de données avec des valeurs telles que `beta`, `staging`, `production`, etc. Dans ce cas, vous pouvez créer une stratégie qui limite certains utilisateurs aux instances de base de données fondées sur la valeur de balise `environment`.

> **Note**
>
> Les identifiants des balises personnalisées sont sensibles à la casse.

Le tableau suivant répertorie les identifiants des balises RDS que vous pouvez utiliser dans un élément `Condition`.

Identifiant de balise RDS	S'applique à
`db-tag`	Instances de base de données, y compris les réplicas en lecture
`snapshot-tag`	Instantanés de base de données
`ri-tag`	Instances de base de données réservées
`og-tag`	Groupes d'options DB
`pg-tag`	Groupes de paramètres DB
`subgrp-tag`	Groupes de sous-réseaux DB
`es-tag`	Abonnements aux événements
`cluster-tag`	Clusters DB
`cluster-pg-tag`	Groupes de paramètres de cluster DB
`cluster-snapshot-tag`	Instantanés de cluster DB

La syntaxe d'une condition de balise personnalisée est la suivante :

```
"Condition":{"StringEquals":{"rds:rds-tag-identifier/tag-name":
["value"]} }
```

Par exemple, l'élément `Condition` suivant s'applique aux instances de bases de données avec une balise nommée `environment` et la valeur de balise `production`.

```
"Condition":{"StringEquals":{"rds:db-tag/environment": ["production"]} }
```

Pour plus d'informations sur la création de balises, consultez Marquage des ressources Amazon Aurora et Amazon RDS.

Important

Si vous gérez l'accès à vos ressources RDS à l'aide du balisage, nous vous recommandons de sécuriser l'accès aux balises pour vos ressources RDS. Vous pouvez gérer l'accès aux balises en créant des stratégies pour les actions `AddTagsToResource` et `RemoveTagsFromResource`. Par exemple, la politique suivante refuse aux utilisateurs la capacité à ajouter ou supprimer des balises pour toutes les ressources. Vous pouvez alors créer des politiques pour autoriser des utilisateurs spécifiques à ajouter ou supprimer des balises.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyTagUpdates",
         "Effect":"Deny",
         "Action":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"*"
      }
   ]
}
```

Pour afficher la liste des actions Aurora, consultez Actions définies par Amazon RDS dans Référence de l'autorisation de service.

Exemples de politiques : Utilisation de balises personnalisées

Les exemples suivants montrent comment vous pouvez utiliser des balises personnalisées dans les stratégies d'autorisation IAM Amazon Aurora. Pour plus d'informations sur l'ajout de balises à une ressource Amazon Aurora, consultez Amazon Resource Names (ARN) dans Amazon RDS.

> **Note**
>
> Tous les exemples utilisent la région us-west-2 et contiennent un compte fictif. IDs

Exemple 1 : Accorder une autorisation pour des actions sur une ressource à l'aide d'une balise spécifique avec deux valeurs différentes

La politique suivante accorde l'autorisation d'exécuter l'opération d'API `CreateDBSnapshot` sur les instances de base de données avec la balise `stage` définie sur `development` ou `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowAnySnapshotName",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:snapshot:*"
      },
      {
         "Sid":"AllowDevTestToCreateSnapshot",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBSnapshot"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
```

```
            "StringEquals":{
                "rds:db-tag/stage":[
                  "development",
                  "test"
               ]
            }
         }
      }
   ]
}
```

La politique suivante accorde l'autorisation d'exécuter l'opération d'API `ModifyDBInstance` sur les instances de base de données avec la balise `stage` définie sur `development` ou `test`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowChangingParameterOptionSecurityGroups",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
            ],
          "Resource": [
            "arn:aws:rds:*:123456789012:pg:*",
            "arn:aws:rds:*:123456789012:secgrp:*",
            "arn:aws:rds:*:123456789012:og:*"
            ]
       },
       {
         "Sid":"AllowDevTestToModifyInstance",
         "Effect":"Allow",
         "Action":[
            "rds:ModifyDBInstance"
            ],
         "Resource":"arn:aws:rds:*:123456789012:db:*",
         "Condition":{
            "StringEquals":{
               "rds:db-tag/stage":[
```

```
                "development",
                "test"
                ]
            }
        }
     }
  ]
}
```

Exemple 2 : Refuser explicitement l'autorisation de créer une instance de base de données qui utilise les groupes de paramètres DB spécifiés

La politique suivante refuse explicitement l'autorisation de créer une instance de base de données qui utilise les groupes de paramètres DB avec des valeurs de balise spécifiques. Vous pouvez appliquer cette politique si vous avez besoin qu'un groupe de paramètres DB créé par le client soit toujours utilisé lors de la création des instances de bases de données. Notez que les stratégies qui utilisent `Deny` sont le plus souvent utilisées pour limiter un accès accordé par une stratégie plus large.

Le refus explicite d'une autorisation a priorité sur toutes les autres autorisations accordées. Cela garantit que des identités n'obtiendront pas par erreur une autorisation que vous ne souhaitez pas accorder.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"DenyProductionCreate",
         "Effect":"Deny",
         "Action":"rds:CreateDBInstance",
         "Resource":"arn:aws:rds:*:123456789012:pg:*",
         "Condition":{
            "StringEquals":{
               "rds:pg-tag/usage":"prod"
            }
         }
      }
   ]
```

```
}
```

Exemple 3 : Accorder une autorisation pour des actions sur une instance de base de données dont le nom d'instance a un nom d'utilisateur comme préfixe

La stratégie suivante accorde l'autorisation d'appeler une API quelconque (à l'exception de `AddTagsToResource` et de `RemoveTagsFromResource`) sur une instance de base de données dont le nom d'instance de base de données a comme préfixe le nom de l'utilisateur et a une balise nommée `stage` égale à `devo` ou qui n'a pas de balise nommée `stage`.

La ligne `Resource` dans la stratégie identifie une ressource par son Amazon Resource Name (ARN). Pour plus d'informations sur l'utilisation ARNs des ressources , consultezAmazon Resource Names (ARN) dans Amazon RDS.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowFullDevAccessNoTags",
         "Effect":"Allow",
         "NotAction":[
            "rds:AddTagsToResource",
            "rds:RemoveTagsFromResource"
         ],
         "Resource":"arn:aws:rds:*:123456789012:db:${aws:username}*",
         "Condition":{
            "StringEqualsIfExists":{
               "rds:db-tag/stage":"devo"
            }
         }
      }
   ]
}
```

Octroi d’autorisation de baliser les ressources Aurora lors de la création

Certaines actions RDS API vous permettent de spécifier des balises lorsque vous créez la ressource. Vous pouvez utiliser des balises de ressource pour implémenter le contrôle basé sur les attributs (ABAC). Pour plus d'informations, voir À quoi sert ABAC ? AWS et Contrôle de l'accès aux AWS ressources à l'aide de balises.

Pour permettre aux utilisateurs d’attribuer des balises aux ressources au moment de la création, ils doivent avoir les autorisations d’utiliser l’action qui crée la ressource (par exemple, `rds:CreateDBCluster`). Si les balises sont spécifiées dans l’action de création, RDS effectue des autorisations supplémentaires sur l’action `rds:AddTagsToResource` afin de vérifier si les utilisateurs disposent des autorisations nécessaires pour créer des balises. Par conséquent, les utilisateurs doivent également avoir des autorisations explicites d’utiliser l’action `rds:AddTagsToResource`.

Dans la définition de politique IAM de l’action `rds:AddTagsToResource`, vous pouvez utiliser la clé de condition `aws:RequestTag` pour exiger des balises dans une demande de balisage d’une ressource.

Par exemple, la politique suivante permet aux utilisateurs de créer des instances de base de données et d’appliquer des balises lors de la création d’une instances de base de données, mais uniquement avec des clés de balise spécifiques (`environment` ou `project`) :

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBInstance"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "rds:AddTagsToResource"
           ],
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/environment": ["production", "development"],
                    "aws:RequestTag/project": ["dataanalytics", "webapp"]
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": ["environment", "project"]
                }
            }
        }
    ]
}
```

Cette politique refuse toute demande de création d'instance de base de données qui inclut les balises `environment` ou `project`, ou qui ne spécifie aucune de ces balises. En outre, les utilisateurs doivent spécifier des valeurs pour les balises qui correspondent aux valeurs autorisées dans la politique.

La politique suivante permet aux utilisateurs de créer des clusters de bases de données et d'appliquer des balises pendant la création, à l'exception de la balise `environment=prod` :

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
               "rds:CreateDBCluster"
           ],
           "Resource": "*"
       },
       {
           "Effect": "Allow",
           "Action": [
               "rds:AddTagsToResource"
           ],
           "Resource": "*",
           "Condition": {
```

```
                "StringNotEquals": {
                    "aws:RequestTag/environment": "prod"
                }
            }
        }
    ]
}
```

Actions d’API RDS prises en charge pour le balisage lors de la création

Les actions d’API RDS suivantes prennent en charge le balisage lors de la création d’une ressource. Pour ces actions, vous pouvez spécifier des balises lorsque vous créez la ressource :

- `CreateBlueGreenDeployment`
- `CreateCustomDBEngineVersion`
- `CreateDBCluster`
- `CreateDBClusterEndpoint`
- `CreateDBClusterParameterGroup`
- `CreateDBClusterSnapshot`
- `CreateDBInstance`
- `CreateDBInstanceReadReplica`
- `CreateDBParameterGroup`
- `CreateDBProxy`
- `CreateDBProxyEndpoint`
- `CreateDBSecurityGroup`
- `CreateDBShardGroup`
- `CreateDBSnapshot`
- `CreateDBSubnetGroup`
- `CreateEventSubscription`
- `CreateGlobalCluster`
- `CreateIntegration`
- `CreateOptionGroup`
- `CreateTenantDatabase`

- `CopyDBClusterParameterGroup`
- `CopyDBClusterSnapshot`
- `CopyDBParameterGroup`
- `CopyDBSnapshot`
- `CopyOptionGroup`
- `RestoreDBClusterFromS3`
- `RestoreDBClusterFromSnapshot`
- `RestoreDBClusterToPointInTime`
- `RestoreDBInstanceFromDBSnapshot`
- `RestoreDBInstanceFromS3`
- `RestoreDBInstanceToPointInTime`
- `PurchaseReservedDBInstancesOffering`

Si vous utilisez l'API AWS CLI or pour créer une ressource avec des balises, le `Tags` paramètre est utilisé pour appliquer des balises aux ressources lors de la création.

Pour ces actions d’API, si le balisage échoue, la ressource n’est pas créée et la demande échoue avec une erreur. Ainsi, les ressources sont créées avec des balises ou elles ne sont pas créées du tout, ce qui empêche la création de ressources sans les balises prévues.

Politiques AWS gérées pour Amazon RDS

Pour ajouter des autorisations à des jeux d'autorisations et des rôles, il est plus facile d'utiliser des politiques gérées par AWS que d'écrire des politiques vous-même. Il faut du temps et de l'expertise pour créer des politiques gérées par le client IAM qui ne fournissent à votre équipe que les autorisations dont elle a besoin. Pour démarrer rapidement, vous pouvez utiliser nos politiques gérées par AWS. Ces politiques couvrent des cas d'utilisation courants et sont disponibles dans votre Compte AWS. Pour plus d'informations sur les politiques gérées par AWS, consultez Politiques gérées par AWSdans le Guide de l'utilisateur IAM.

Les Services AWS assurent la maintenance et la mise à jour des politiques gérées par AWS. Vous ne pouvez pas modifier les autorisations définies dans les politiques gérées par AWS. Les services ajoutent occasionnellement des autorisations à une politique gérée par AWS pour prendre en charge de nouvelles fonctionnalités. Ce type de mise à jour affecte toutes les identités (jeux d'autorisations et rôles) auxquelles la politique est attachée. Les services sont très susceptibles de mettre à jour une politique gérée par AWS quand une nouvelle fonctionnalité est lancée ou quand de nouvelles opérations sont disponibles. Les services ne suppriment pas les autorisations d'une politique gérée AWS. Ainsi, les mises à jour de politique n'interrompent vos autorisations existantes.

En outre, AWS prend en charge des politiques gérées pour des fonctions professionnelles couvrant plusieurs services. Par exemple, la politique `ReadOnlyAccess` gérée par AWS donne accès en lecture seule à tous les Services AWS et ressources. Quand un service lance une nouvelle fonctionnalité, AWS ajoute des autorisations en lecture seule pour les nouvelles opérations et ressources. Pour obtenir la liste des politiques de fonctions professionnelles et leurs descriptions, consultez la page politiques gérées par AWS pour les fonctions de tâche dans le Guide de l'utilisateur IAM.

Rubriques

- Politique gérée par AWS : AmazonRDSPreviewServiceRolePolicy
- Politique gérée par AWS : AmazonRDSBetaServiceRolePolicy

Politique AWS gérée : AmazonRDSReadOnlyAccess

Cette politique autorise l'accès en lecture seule à Amazon RDS par l'intermédiaire de la AWS Management Console.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `rds` : permet aux principaux de décrire les ressources Amazon RDS et de dresser la liste des balises pour les ressources Amazon RDS.
- `cloudwatch` : permet aux principaux d'obtenir des statistiques sur les métriques d'Amazon CloudWatch.
- `ec2` : permet aux principaux de décrire les zones de disponibilité et les ressources de réseaux.
- `logs` : permet aux principaux de décrire les flux de journaux CloudWatch Logs des groupes de journaux et d'obtenir des événements de journaux CloudWatch Logs.
- `devops-guru` : permet aux responsables de décrire les ressources couvertes par Amazon DevOps Guru, qui sont spécifiées soit par des noms de pile CloudFormation, soit par des identifications de ressources.

Pour plus d'informations sur cette politique, y compris le document de politique JSON, consultez AmazonRDSReadOnlyAccess dans le Guide de référence des politiques gérées par AWS.

Politique AWS gérée : AmazonRDSFullAccess

Cette politique fournit un accès complet à Amazon RDS par l'intermédiaire de la AWS Management Console.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `rds` : donne aux principaux un accès complet à Amazon RDS.
- `application-autoscaling` : permet aux principaux de décrire et de gérer les cibles et les politiques de scalabilité automatique des applications.

- `cloudwatch` : permet aux principaux d'obtenir les statistiques des métriques CloudWatch et de gérer les alarmes CloudWatch.
- `ec2` : permet aux principaux de décrire les zones de disponibilité et les ressources de réseaux.
- `logs` : permet aux principaux de décrire les flux de journaux CloudWatch Logs des groupes de journaux et d'obtenir des événements de journaux CloudWatch Logs.
- `outposts` : permet aux principaux d'obtenir des types d'instance AWS Outposts.
- `pi` : permet aux principaux d'obtenir les métriques de Performance Insights.
- `sns` : permet aux principaux de s'abonner à Amazon Simple Notification Service (Amazon SNS) et à ses rubriques, et de publier des messages Amazon SNS.
- `devops-guru` : permet aux responsables de décrire les ressources couvertes par Amazon DevOps Guru, qui sont spécifiées soit par des noms de pile CloudFormation, soit par des identifications de ressources.

Pour plus d'informations sur cette politique, y compris le document de politique JSON, consultez AmazonRDSFullAccess dans le Guide de référence des politiques gérées par AWS.

Politique AWS gérée : AmazonRDSDataFullAccess

Cette politique permet un accès complet pour utiliser l'API de données et l'éditeur de requêtes sur les clusters Aurora Serverless dans un Compte AWS spécifique. Cette politique permet au Compte AWS d'obtenir la valeur d'un secret à partir de AWS Secrets Manager.

Vous pouvez associer la politique `AmazonRDSDataFullAccess` à vos identités IAM.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `dbqms` : permet aux principaux d'accéder, de créer, de supprimer, de décrire et de mettre à jour des requêtes. Le service `dbqms` (Database Query Metadata Service, service de métadonnées de requête de base de données) est un service interne uniquement. La commande fournit vos requêtes récentes et sauvegardées pour l'éditeur de requêtes sur la AWS Management Console pour plusieurs Services AWS, y compris Amazon RDS.
- `rds-data` : permet aux principaux d'exécuter des instructions SQL sur les bases de données Aurora Serverless.
- `secretsmanager` : permet aux principaux d'obtenir la valeur d'un secret à partir de AWS Secrets Manager.

Pour plus d'informations sur cette politique, y compris le document de politique JSON, consultez AmazonRDSDataFullAccess dans le Guide de référence des politiques gérées par AWS.

Politique AWS gérée : AmazonRDSEnhancedMonitoringRole

Cette politique permet d'accéder aux journaux Amazon CloudWatch Logs pour la surveillance améliorée Amazon RDS.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `logs` : permet aux principaux de créer des groupes de journaux CloudWatch Logs et des politiques de rétention, et de créer et décrire les flux de journaux CloudWatch Logs des groupes de journaux. La commande permet également aux principaux de placer et d'extraire des événements de journal CloudWatch Logs.

Pour plus d'informations sur cette politique, y compris le document de politique JSON, consultez AmazonRDSEnhancedMonitoringRole dans le Guide de référence des politiques gérées par AWS.

Politique AWS gérée : AmazonRDSPerformanceInsightsReadOnly

Cette politique fournit un accès en lecture seule à l'analyse des performances d'Amazon RDS pour les instances Amazon de base de données RDS et les clusters de bases de données Amazon Aurora.

Cette politique inclut désormais `Sid` (ID d'instruction) comme identifiant pour l'instruction de la politique.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `rds` : permet aux principaux de décrire des instances de base de données Amazon RDS et des clusters de base de données Amazon Aurora.
- `pi` : permet aux principaux de faire des appels à l'API Analyse des performances d'Amazon RDS et d'accéder aux métriques de Performance Insights.

Pour plus d'informations sur cette politique, y compris le document de politique JSON, consultez AmazonRDSPerformanceInsightsReadOnly dans le Guide de référence des politiques gérées par AWS.

Politique gérée AWS: AmazonRDSPerformanceInsightsFullAccess

Cette politique fournit un accès complet à l'analyse des performances d'Amazon RDS pour les instances de base de données Amazon RDS et les clusters de bases de données Amazon Aurora.

Cette politique inclut désormais `Sid` (ID d'instruction) comme identifiant pour l'instruction de la politique.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `rds` : permet aux principaux de décrire des instances de base de données Amazon RDS et des clusters de bases de données Amazon Aurora.
- `pi` – Permet aux principaux d'appeler l'API Analyse des performances d'Amazon RDS et de créer, d'afficher et de supprimer des rapports d'analyse des performances.
- `cloudwatch` – Permet aux principaux de répertorier toutes les métriques Amazon CloudWatch et d'obtenir des données de métriques et des statistiques.

Pour plus d'informations sur cette politique, y compris le document de politique JSON, consultez AmazonRDSPerformanceInsightsFullAccess dans le Guide de référence des politiques gérées par AWS.

Politique AWS gérée : AmazonRDSDirectoryServiceAccess

Cette politique permet à Amazon RDS d'effectuer des appels vers Directory Service.

Détails de l'autorisation

Cette politique inclut l'autorisation suivante :

- `ds` : permet aux principaux de décrire les répertoires Directory Service et de contrôler l'autorisation aux répertoires Directory Service.

Pour plus d'informations sur cette politique, y compris le document de politique JSON, consultez AmazonRDSDirectoryServiceAccess dans le Guide de référence des politiques gérées par AWS.

Politique AWS gérée : AmazonRDSServiceRolePolicy

Vous ne pouvez pas attacher `AmazonRDSServiceRolePolicy` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon RDS d'effectuer des actions en votre nom. Pour plus d'informations, consultez Autorisations des rôles liés à un service pour Amazon Aurora.

Politique gérée par AWS : AmazonRDSPreviewServiceRolePolicy

Ne joignez pas `AmazonRDSPreviewServiceRolePolicy` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon RDS d'appeler les services AWS au nom de vos ressources de base de données RDS. Pour plus d'informations, consultez Rôles liés à un service pour Amazon RDS Preview.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `ec2` : permet aux principaux de décrire les zones de disponibilité et les ressources de réseaux.
- `secretsmanager` : permet aux principaux d'obtenir la valeur d'un secret à partir de AWS Secrets Manager.
- `cloudwatch`, `logs` - Permet à Amazon RD de charger les métriques et les journaux d'une instance de base de données sur CloudWatch via l'agent CloudWatch.

Pour plus d'informations sur cette politique, y compris le document de politique JSON, consultez AmazonRDSPreviewServiceRolePolicy dans le Guide de référence des politiques gérées par AWS.

Politique gérée par AWS : AmazonRDSBetaServiceRolePolicy

Ne joignez pas `AmazonRDSBetaServiceRolePolicy` à vos entités IAM. Cette politique est attachée à un rôle lié à un service qui permet à Amazon RDS d'appeler les services AWS au nom de vos ressources de base de données RDS. Pour plus d'informations, consultez Autorisations du rôle lié à un service pour Amazon RDS bêta.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes :

- `ec2` : permet à Amazon RDS d'effectuer des opérations de sauvegarde sur l'instance de base de données qui fournit des fonctionnalités de restauration ponctuelle.

- `secretsmanager` - Permet à Amazon RDS de gérer des secrets spécifique à une instance de base de données créés par Amazon RDS.
- `cloudwatch`, `logs` - Permet à Amazon RD de charger les métriques et les journaux d'une instance de base de données sur CloudWatch via l'agent CloudWatch.

Pour plus d'informations sur cette politique, y compris le document de politique JSON, consultez AmazonRDSBetaServiceRolePolicy dans le Guide de référence des politiques gérées par AWS.

Mises à jour Amazon RDS des politiques gérées par AWS

Affichez les détails des mises à jour des politiques gérées par AWS pour Amazon RDS depuis que ce service a commencé à assurer le suivi des modifications. Pour recevoir des alertes automatiques sur les modifications apportées à cette page, abonnez-vous au flux RSS de la page Document history (Historique des documents) d'Amazon RDS.

Modification	Description	Date
Politique gérée par AWS : AmazonRDSPreviewServiceRolePolicy – Mise à jour de la politique existante	Amazon RDS a supprimé l'autorisation `sns:Publish` de `AmazonRDSPreviewServiceRolePolicy` du rôle lié au service `AWSServiceRoleForRDSPreview` . Pour plus d'informations, consultez Politique gérée par AWS : AmazonRDSPreviewServiceRolePolicy.	7 août 2024
Politique gérée par AWS : AmazonRDSBetaServiceRolePolicy – Mise à jour de la politique existante	Amazon RDS a supprimé l'autorisation `sns:Publish` de `AmazonRDSBetaServiceRolePolicy` du rôle lié au service `AWSServiceRoleForRDSBeta` . Pour plus d'informations, consultez Politique gérée par AWS : AmazonRDSBetaServiceRolePolicy.	7 août 2024
Politique AWS gérée : AmazonRDSServiceRolePolicy – Mise à jour de la politique existante	Amazon RDS a supprimé l'autorisation `sns:Publish` de `AmazonRDSServiceRolePolicy` du rôle lié au service `AWSServiceRoleForRDS` . Pour plus	2 juillet 2024

Modification	Description	Date
	d’informations, consultez Politique AWS gérée : AmazonRDSServiceRolePolicy.	
Politiques AWS gérées pour Amazon RDS – Mise à jour de la politique existante	Amazon RDS a ajouté une nouvelle autorisation à `AmazonRDSCustomServiceRolePolicy` du rôle `AWSServiceRoleForRDSCustom` liés au service afin de permettre à RDS Custom for SQL Server de modifier le type d’instance de l’hôte de base de données sous-jacent. RDS a également ajouté l’autorisation `ec2:DescribeInstanceTypes` pour obtenir des informations sur le type d’instance pour l’hôte de base de données. Pour plus d’informations, consultez Politiques AWS gérées pour Amazon RDS.	8 avril 2024

Modification	Description	Date
Politiques AWS gérées pour Amazon RDS – Nouvelle politique	Amazon RDS a ajouté une nouvelle politique gérée nommée `AmazonRDSCustomInstanceProfileRolePolicy` pour permettre à RDS Custom d'effectuer des actions d'automatisation et des tâches de gestion de base de données via un profil d'instance EC2. Pour plus d'informations, consultez Politiques AWS gérées pour Amazon RDS.	27 février 2024
Autorisations des rôles liés à un service pour Amazon Aurora – Mise à jour d'une politique existante	Amazon RDS a ajouté de nouvelles ID d'instruction à la politique `AmazonRDSServiceRolePolicy` du rôle `AWSServiceRoleForRDS` lié au service. Pour plus d'informations, consultez Autorisations des rôles liés à un service pour Amazon Aurora.	19 janvier 2024

Modification	Description	Date
Politiques AWS gérées pour Amazon RDS – Mise à jour des politiques existantes	Les politiques gérées par `AmazonRDSPerformanceInsightsReadOnly` et `AmazonRDSPerformanceInsightsFullAccess` incluent désormais `Sid` (ID d'instruction) comme identifiant dans l'instruction de la politique. Pour plus d'informations, consultez Politique AWS gérée : AmazonRDSPerformanceInsightsReadOnly et Politique gérée AWS: AmazonRDSPerformanceInsightsFullAccess.	23 octobre 2023
Politiques AWS gérées pour Amazon RDS – Mise à jour de la politique existante	Amazon RDS a ajouté de nouvelles autorisations à la politique gérée `AmazonRDSFullAccess`. Les autorisations vous permettent de générer, d'afficher et de supprimer le rapport d'analyse des performances pendant une période donnée. Pour plus d'informations sur la configuration de stratégies d'accès pour l'analyse des performances, consultez Configuration des politiques d'accès pour Performance Insights	17 août 2023

Modification	Description	Date
Politiques AWS gérées pour Amazon RDS – Nouvelle politique et mise à jour de la politique existante	Amazon RDS a ajouté de nouvelles autorisations à la politique gérée `AmazonRDSPerformanceInsightsReadOnly` et une nouvelle politique gérée nommée `AmazonRDSPerformanceInsightsFullAccess`. Ces autorisations vous permettent d'analyser les informations de performances pour une période donnée, de consulter les résultats d'analyse ainsi que les recommandations, et de supprimer les rapports. Pour plus d'informations sur la configuration de stratégies d'accès pour l'analyse des performances, consultez Configuration des politiques d'accès pour Performance Insights	16 août 2023

Modification	Description	Date
Politiques AWS gérées pour Amazon RDS – Mise à jour d'une politique existante	Amazon RDS a ajouté un nouvel espace de noms Amazon CloudWatch `ListMetrics` pour `AmazonRDSFullAccess` et `AmazonRDSReadOnlyAccess` . Cet espace de nom est nécessaire à Amazon RDS pour répertorier des métriques spécifiques sur l'utilisation des ressources. Pour plus d'informations, consultez Présentation de la gestion des autorisations d'accès à vos ressources CloudWatch dans le Guide de l'utilisateur Amazon CloudWatch.	4 avril 2023

Modification	Description	Date
Autorisations des rôles liés à un service pour Amazon Aurora – Mise à jour d'une politique existante	Amazon RDS a ajouté de nouvelles autorisations à la politique `AmazonRDSServiceRolePolicy` du rôle `AWSServiceRoleForRDS` lié au service pour l'intégration avec AWS Secrets Manager. RDS nécessite une intégration à Secrets Manager pour gérer les mots de passe des utilisateurs principaux dans Secrets Manager. Le secret utilise une convention de dénomination réservée et restreint les mises à jour des clients. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager.	22 décembre 2022

Modification	Description	Date
Politiques AWS gérées pour Amazon RDS – Mise à jour des politiques existantes	Amazon RDS a ajouté une nouvelle autorisation aux politiques gérées `AmazonRDSFullAccess` et `AmazonRDSReadOnlyAccess` pour vous permettre d'activer Amazon DevOps Guru dans la console RDS. Cette autorisation est requise pour vérifier si DevOps Guru est activé. Pour plus d'informations, consultez Configuration des politiques d'accès IAM pour DevOps Guru for RDS.	19 décembre 2022

Modification	Description	Date
Autorisations des rôles liés à un service pour Amazon Aurora – Mise à jour d'une politique existante	Amazon RDS a ajouté un nouvel espace de noms Amazon CloudWatch à `AmazonRDSPreviewServiceRolePolicy` pour `PutMetricData` . Cet espace de nom est nécessaire à Amazon RDS pour publier des métriques sur l'utilisation des ressources. Pour plus d'informations, consultez Using condition keys to limit access to CloudWatch namespaces (Utilisation des clés de condition pour limiter l'accès aux espaces de noms CloudWatch) dans le Guide de l'utilisateur Amazon CloudWatch.	7 juin 2022

Modification	Description	Date
Autorisations des rôles liés à un service pour Amazon Aurora – Mise à jour d'une politique existante	Amazon RDS a ajouté un nouvel espace de noms Amazon CloudWatch à `AmazonRDSBetaServiceRolePolicy` pour `PutMetricData` . Cet espace de nom est nécessaire à Amazon RDS pour publier des métriques sur l'utilisation des ressources. Pour plus d'informations, consultez Using condition keys to limit access to CloudWatch namespaces (Utilisation des clés de condition pour limiter l'accès aux espaces de noms CloudWatch) dans le Guide de l'utilisateur Amazon CloudWatch.	7 juin 2022

Modification	Description	Date
Autorisations des rôles liés à un service pour Amazon Aurora – Mise à jour d'une politique existante	Amazon RDS a ajouté un nouvel espace de noms Amazon CloudWatch à `AWSServiceRoleForRDS` pour `PutMetricData` . Cet espace de nom est nécessaire à Amazon RDS pour publier des métriques sur l'utilisation des ressources. Pour plus d'informations, consultez Using condition keys to limit access to CloudWatch namespaces (Utilisation des clés de condition pour limiter l'accès aux espaces de noms CloudWatch) dans le Guide de l'utilisateur Amazon CloudWatch.	22 avril 2022

Modification	Description	Date
Politiques AWS gérées pour Amazon RDS – Nouvelle politique	Amazon RDS a ajouté une nouvelle politique gérée nommée `AmazonRDSPerformanceInsightsReadOnly` pour permettre à Amazon RDS d'appeler des services AWS au nom de vos instances de base de données. Pour plus d'informations sur la configuration de stratégies d'accès pour l'analyse des performances, consultez Configuration des politiques d'accès pour Performance Insights	10 mars 2022

Modification	Description	Date
Autorisations des rôles liés à un service pour Amazon Aurora – Mise à jour d'une politique existante	Amazon RDS a ajouté de nouveaux espaces de noms Amazon CloudWatch à `AWSServiceRoleForRDS` pour `PutMetricData` . Ces espaces de noms sont nécessaires pour qu'Amazon DocumentDB (compatible avec MongoDB) et Amazon Neptune puissent publier des métriques CloudWatch. Pour plus d'informations, consultez Using condition keys to limit access to CloudWatch namespaces (Utilisation des clés de condition pour limiter l'accès aux espaces de noms CloudWatch) dans le Guide de l'utilisateur Amazon CloudWatch.	4 mars 2022
Amazon RDS a commencé à assurer le suivi des modifications	Amazon RDS a commencé à assurer le suivi des modifications pour ses politiques gérées par AWS.	26 octobre 2021

Prévention des problèmes d'adjoint confus entre services

Le problème de l’adjoint confus est un problème de sécurité dans lequel une entité qui n’a pas l’autorisation d’effectuer une action peut contraindre une entité plus privilégiée à effectuer cette action. EnAWS, l'usurpation d'identité interservices peut entraîner un problème de confusion chez les adjoints.

L’usurpation d’identité entre services peut se produire lorsqu’un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé pour utiliser ses autorisations et agir sur les ressources d'un autre client, d'une manière dont il ne devrait pas avoir accès. Pour éviter cela, AWS fournit des outils qui peuvent vous aider à protéger vos données pour tous les services dont les responsables ont obtenu l'accès aux ressources de votre compte. Pour de plus amples informations, veuillez consulter Le problème du député confus dans le Guide de l’utilisateur IAM.

Afin de limiter les autorisations octroyées par Amazon RDS à un autre service pour une ressource spécifique, nous vous recommandons d'utiliser les clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount` dans les politiques de ressources.

Dans certains cas, la valeur `aws:SourceArn` ne contient pas l'ID du compte, par exemple lorsque vous utilisez l'Amazon Resource Name (ARN) pour un compartiment Amazon S3. Dans ces cas, veillez à utiliser les deux clés de contexte de condition globale pour limiter les autorisations. Dans certains cas, vous utilisez les deux clés de contexte de condition globale et la valeur `aws:SourceArn` contient l'ID du compte. Dans ces cas, assurez-vous que la valeur `aws:SourceAccount` et le compte dans le `aws:SourceArn` utilisent le même ID de compte lorsqu'ils sont utilisés dans la même instruction de politique. Utilisez `aws:SourceArn` si vous souhaitez qu'une seule ressource soit associée à l'accès entre services. Si vous souhaitez autoriser l'association d'une ressource du AWS compte spécifié à l'utilisation interservices, utilisez`aws:SourceAccount`.

Assurez-vous que la valeur de `aws:SourceArn` est un ARN d'un type de ressource Amazon RDS. Pour de plus amples informations, veuillez consulter Amazon Resource Names (ARN) dans Amazon RDS.

Le moyen le plus efficace de se protéger contre le problème de député confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l’ARN complet de la ressource. Dans certains cas, vous ne connaissez pas l'ARN complet de la ressource ou vous spécifiez plusieurs ressources. Dans ces cas, utilisez la clé de contexte de condition globale

`aws:SourceArn` avec des caractères génériques (*) pour les parties inconnues de l'ARN. Par exemple : `arn:aws:rds:*:123456789012:*`.

L'exemple suivant montre comment utiliser les clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount` pour dans Amazon RDS afin d'éviter le problème de l'adjoint confus.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "rds.amazonaws.com"
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:rds:us-east-1:123456789012:db:mydbinstance"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

Pour obtenir d'autres exemples de politiques qui utilisent les clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount`, veuillez consulter les sections suivantes :

- Octroi d’autorisations de publication de notifications dans une rubrique Amazon SNS
- Configuration de l’accès à un compartiment Amazon S3 (importation PostgreSQL)
- Configuration de l’accès à un compartiment Amazon S3 (exportation PostgreSQL)

Authentification de base de données IAM

Vous pouvez vous authentifier auprès de votre cluster d' de base de données à l'aide de l'authentification de base de données Gestion des identités et des accès AWS (IAM). L'authentification de base de données IAM fonctionne avec Aurora MySQL et Aurora PostgreSQL. Grâce à cette méthode d'authentification, vous n'avez plus besoin de mot de passe pour vous connecter à un cluster de bases de données. En revanche, un jeton d'authentification est nécessaire.

Un jeton d'authentification est une chaîne de caractères unique générée par Amazon Aurora sur demande. Les jetons d'authentification sont générés à l'aide de AWS la version 4 de Signature. Chaque jeton a une durée de vie de 15 minutes. Il n'est pas nécessaire de stocker les informations d'identification des utilisateurs dans la base de données, car l'authentification est gérée de manière externe avec IAM. Vous pouvez aussi toujours utiliser l'authentification de base de données standard. Le jeton est uniquement utilisé pour l'authentification et n'affecte pas la session une fois qu'il est établi.

L'authentification de base de données IAM offre les avantages suivants :

- Le trafic réseau à destination et en provenance de la base de données est chiffré à l'aide de Secure Socket Layer (SSL) ou de Transport Layer Security (TLS). Pour plus d'informations sur l'utilisation SSL/TLS d', consultezUtilisation SSL/TLS pour chiffrer une connexion à une de clusters.
- Vous pouvez utiliser IAM pour gérer de façon centralisée l'accès à vos ressources de base de données, au lieu de gérer l'accès de manière individuelle sur chaque cluster de bases de données.
- Pour les applications exécutées sur Amazon EC2, vous pouvez utiliser les informations d'identification de profil spécifiques à votre EC2 instance pour accéder à votre base de données au lieu d'un mot de passe, afin de renforcer la sécurité.

En règle générale, envisagez d'utiliser l'authentification de base de données IAM lorsque vos applications créent moins de 200 connexions par seconde, et que vous ne souhaitez pas gérer les noms d'utilisateur et les mots de passe directement dans le code de votre application.

Le pilote JDBC Amazon Web Services (AWS) prend en charge l'authentification de la base de données IAM. Pour plus d'informations, consultez la section Plug-in d'authentification AWS IAM dans le référentiel de pilotes JDBC Amazon Web Services (AWS). GitHub

Le pilote Python Amazon Web Services (AWS) prend en charge l'authentification de la base de données IAM. Pour plus d'informations, consultez la section Plug-in d'authentification AWS IAM dans le GitHubréférentiel de pilotes Python Amazon Web Services (AWS).

Consultez les rubriques suivantes pour apprendre à configurer IAM pour l'authentification de la base de données :

- Activation et désactivation de l'authentification de base de données IAM
- Création et utilisation d'une politique IAM pour l'accès à une base de données IAM
- Création d'un compte de base de données à l'aide de l'authentification IAM
- Connexion à votre cluster de base de données à l'aide de l'authentification IAM.

Disponibilité des régions et des versions

La disponibilité et la prise en charge des fonctions varient selon les versions spécifiques de chaque moteur de base de données Aurora, et selon les Régions AWS. Pour obtenir plus d'informations sur la disponibilité des versions et des régions avec Aurora et l'authentification de la base de données IAM, consultez Régions et moteurs de base de données Aurora pris en charge pour l'authentification de base de données IAM.

Pour Aurora MySQL, toutes les classes d'instance de base de données prises en charge prennent en charge l'authentification de base de données IAM, à l'exception de db.t2.small et db.t3.small. Pour plus d'informations sur les classes d'instance de base de données prises en charge, consultez Moteurs de base de données pris en charge pour les classes d'instance de base de données.

Support CLI et kit SDK

L'authentification de base de données IAM est disponible pour AWS CLIet pour les langues AWS SDKs spécifiques suivantes :

- AWS SDK pour .NET
- AWS SDK pour C++
- AWS SDK pour Go
- AWS SDK pour Java
- AWS SDK pour JavaScript
- AWS SDK pour PHP
- AWS SDK pour Python (Boto3)
- AWS SDK pour Ruby

Limites de l’authentification de base de données IAM

Les limitations suivantes s’appliquent lors de l’utilisation de l’authentification de base de données IAM :

- Actuellement, l’authentification de base de données IAM ne prend pas en charge toutes les clés de contexte de condition globale.

  Pour plus d’informations sur les clés de contexte de condition globale, consultez Clés de contexte de condition globales AWS dans le Guide de l’utilisateur IAM.
- Pour PostgreSQL, si le rôle IAM (`rds_iam`) est ajouté à un utilisateur (y compris à l’utilisateur principal RDS), l’authentification IAM a priorité sur l’authentification par mot de passe, de sorte que l’utilisateur doit se connecter en tant qu’utilisateur IAM.
- Vous ne pouvez pas utiliser un enregistrement DNS Route 53 personnalisé à la place du point de terminaison du cluster de bases de données pour générer le jeton d’authentification.
- CloudWatch et CloudTrail n'enregistrez pas l'authentification IAM. Ces services ne suivent pas les appels d’API `generate-db-auth-token` qui autorisent le rôle IAM à activer la connexion à la base de données.
- L’authentification de base de données IAM nécessite des ressources de calcul sur le cluster de bases de données. Pour garantir une connectivité fiable, votre base de données doit disposer de 300 à 1 000 Mio de mémoire supplémentaire. Pour connaître la mémoire nécessaire à votre charge de travail, comparez la colonne RES pour les processus RDS dans la liste des processus de surveillance améliorée avant et après l’activation de l’authentification de base de données IAM. Consultez Affichage des métriques du système d’exploitation dans la console RDS.

  Si vous utilisez une instance de classe à performances extensibles, évitez de manquer de mémoire en réduisant d’autant la mémoire utilisée par d’autres paramètres tels que les tampons et le cache.
- Pour Aurora MySQL, vous ne pouvez pas utiliser l'authentification par mot de passe pour un utilisateur de base de données que vous configurez avec l'authentification IAM.
- L’authentification de base de données IAM n’est prise en charge pour RDS on Outposts, quel que soit le moteur.

Recommandations pour l’authentification de base de données IAM

Nous recommandons les pratiques suivantes lors de l’utilisation de l’authentification de base de données IAM :

- Utilisez l’authentification de base de données IAM si votre application exige moins de 200 nouvelles connexions d’authentification de base de données IAM par seconde.

  Les moteurs de base de données qui fonctionnent avec Amazon Aurora n’imposent pas de limites de tentatives d’authentification par seconde. Néanmoins, lorsque vous utilisez l’authentification de base de données IAM, votre application doit générer un jeton d’authentification. Votre application emploie ensuite ce jeton pour la connexion au cluster de bases de données. Si vous dépassez la limite maximale de nouvelles connexions par seconde, le traitement supplémentaire d’authentification de base de données IAM peut entraîner une limitation de la connexion.

  Envisagez d’utiliser le regroupement de connexions dans vos applications pour limiter la création constante de connexions. Cela peut réduire les frais généraux liés à l’authentification de base de données IAM et permettre à vos applications de réutiliser les connexions existantes. Vous pouvez également envisager d’utiliser le proxy RDS pour ces cas d’utilisation. Le proxy RDS entraîne des coûts supplémentaires. Consultez Tarification de Proxy Amazon RDS.
- La taille d’un jeton d’authentification de base de données IAM dépend de nombreux facteurs, notamment du nombre de balises IAM, des politiques de service IAM, de la longueur des ARN, ainsi que d’autres propriétés IAM et de base de données. La taille minimale de ce jeton est généralement d’environ 1 Ko, mais elle peut être plus grande. Étant donné que ce jeton est utilisé comme mot de passe dans la chaîne de connexion à la base de données à l'aide de l'authentification IAM, vous devez vous assurer que les outils de votre pilote de base de données (par exemple, ODBC) and/or ne limitent ni ne tronquent ce jeton en raison de sa taille. Un jeton tronqué provoquera l’échec de la validation d’authentification effectuée par la base de données et IAM.
- Si vous utilisez des informations d’identification temporaires lors de la création d’un jeton d’authentification d’une base de données IAM, les informations d’identification temporaires doivent toujours être valides lorsque vous utilisez le jeton d’authentification d’une base de données IAM pour effectuer une demande de connexion.

Clés contextuelles de condition AWS globale non prises en charge

L'authentification de base de données IAM ne prend pas en charge le sous-ensemble suivant de clés AWS contextuelles de conditions globales.

- `aws:Referer`
- `aws:SourceIp`
- `aws:SourceVpc`

- `aws:SourceVpce`
- `aws:UserAgent`
- `aws:VpcSourceIp`

Pour plus d'informations, consultez Clés de contexte de condition globales AWS dans le Guide de l'utilisateur IAM.

Activation et désactivation de l'authentification de base de données IAM

Par défaut, l'authentification de base de données IAM est désactivée sur les et clusters de bases de données. Vous pouvez activer ou désactiver l'authentification de base de données IAM à l'aide de AWS Management Console AWS CLI, ou de l'API.

Vous pouvez activer l'authentification de base de données IAM lorsque vous effectuez une des actions suivantes :

- Pour créer un nouveau cluster de bases de données avec l'authentification de base de données IAM activée, consultez Création d'un cluster de bases de données Amazon Aurora.
- Pour modifier un cluster de bases de données afin d'activer l'authentification de base de données IAM, consultez Modification d'un cluster de bases de données Amazon Aurora.
- Pour restaurer un cluster de bases de données à partir d'un instantané avec l'authentification de base de données IAM activée, consultez Restauration à partir d'un instantané de cluster de bases de données.
- Pour restaurer un cluster de bases de données à un instant dans le passé avec l'authentification de base de données IAM activée, consultez Restauration d'un cluster de bases de données à une date définie.

Console

Chaque flux de travail de création ou de modification comporte une section Authentification de base de données dans laquelle vous pouvez activer ou désactiver l'authentification de base de données IAM. Dans cette section, choisissez Authentification de base de données par mot de passe et IAM pour activer l'authentification de base de données IAM.

Pour activer ou désactiver l’authentification de base de données IAM pour un cluster de bases de données

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la panneau de navigation, choisissez Databases (Bases de données).
3. Choisissez cluster de bases de données que vous souhaitez modifier.

   > **Note**
   >
   > Vous ne pouvez activer l’authentification IAM que si toutes les instances de base de données du cluster sont compatibles avec IAM. Consultez les exigences de compatibilité présentées dans Disponibilité des régions et des versions.

4. Sélectionnez Modify.
5. Dans la section Authentification de base de données, sélectionnez Mot de passe et authentification de données IAM pour activer l'authentification de base de données IAM. Choisissez Authentification par mot de passe ou Authentification par mot de passe et Kerberos pour désactiver l’authentification IAM.
6. Vous pouvez également choisir d'activer la publication des journaux d'authentification IAM DB dans CloudWatch Logs. Sous Exportations de journaux, choisissez l'option de iam-db-auth-error journal. La publication de vos CloudWatch journaux dans Logs consomme de l'espace de stockage et vous devez payer des frais pour ce stockage. Assurez-vous de supprimer tous les CloudWatch journaux dont vous n'avez plus besoin.
7. Choisissez Continuer.
8. Pour appliquer immédiatement les modifications, choisissez Immédiatement dans la section Planification des modifications.
9. Choisissez ou Modifier le cluster.

AWS CLI

Pour créer un nouveau cluster de base de données avec authentification IAM à l'aide de AWS CLI, utilisez la `create-db-cluster`commande. Spécifiez l’option `--enable-iam-database-authentication`.

Pour mettre à jour un cluster de bases de données existant de manière à activer ou non l’authentification IAM, utilisez la commande de l’ AWS CLI `modify-db-cluster`. Spécifiez

l'option `--enable-iam-database-authentication` ou `--no-enable-iam-database-authentication`, selon le cas.

> **Note**
>
> Vous ne pouvez activer l'authentification IAM que si toutes les instances de base de données du cluster sont compatibles avec IAM. Consultez les exigences de compatibilité présentées dans Disponibilité des régions et des versions.

Par défaut, Aurora procède à la modification pendant la fenêtre de maintenance suivante. Si vous souhaitez ignorer ceci et activer l'authentification de bases de données IAM dès que possible, utilisez le paramètre `--apply-immediately`.

Si vous restaurez un cluster d' de base de données, utilisez l'une des AWS CLI commandes suivantes :

- `restore-db-cluster-to-point-in-time`
- `restore-db-cluster-from-db-snapshot`

Le paramètre d'authentification de base de données IAM par défaut est celui de l'instantané source. Pour le modifier, spécifiez l'option `--enable-iam-database-authentication` ou `--no-enable-iam-database-authentication`, selon le cas.

API RDS

Pour créer une nouvelle instance de base de données avec authentification IAM par l'intermédiaire de l'API, utilisez l'opération d'API `CreateDBCluster`. Définissez le paramètre `EnableIAMDatabaseAuthentication` sur `true`.

Pour mettre à jour un cluster de bases de données existant de manière à activer l'authentification IAM, utilisez l'opération d'API `ModifyDBCluster`. Définissez le paramètre `EnableIAMDatabaseAuthentication` sur `true` pour activer l'authentification IAM ou sur `false` pour la désactiver.

> **Note**
>
> Vous ne pouvez activer l'authentification IAM que si toutes les instances de base de données du cluster sont compatibles avec IAM. Consultez les exigences de compatibilité présentées dans Disponibilité des régions et des versions.

Si vous restaurez un cluster ou une de base de données, utilisez l'une des opérations d'API suivantes :

- `RestoreDBClusterFromSnapshot`
- `RestoreDBClusterToPointInTime`

Le paramètre d'authentification de base de données IAM par défaut est celui de l'instantané source. Pour modifier ce paramètre, définissez le paramètre `EnableIAMDatabaseAuthentication` sur `true` pour activer l'authentification IAM ou sur `false` pour la désactiver.

Création et utilisation d'une politique IAM pour l'accès à une base de données IAM

Pour autoriser un utilisateur ou un rôle à se connecter à votre cluster de bases de données, vous devez créer une politique IAM. Vous attachez ensuite la politique à un jeu d'autorisations ou à un rôle.

> **Note**
>
> Pour en savoir plus sur les stratégies IAM, consultez Identity and Access Management pour Amazon Aurora.

L'exemple de politique suivant autorise un utilisateur à se connecter à un cluster de bases de données en utilisant l'authentification de base de données IAM.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:db-
ABCDEFGHIJKL01234/db_user"
            ]
        }
    ]
}
```

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:cluster-
ABCDEFGHIJKL01234/db_user"
            ]
        }
    ]
}
```

Important

Un utilisateur doté d'autorisations d'administrateur peut accéder aux clusters de bases de données sans autorisations explicites dans une politique IAM. Si vous souhaitez restreindre l'accès de l'administrateur aux et aux clusters de base de données, vous pouvez créer un rôle IAM avec les autorisations appropriées accordant moins de privilèges, puis les assigner à l'administrateur.

> **Note**
>
> Ne confondez pas le préfixe `rds-db:` avec d'autres préfixes d'opération d'API RDS; qui commencent par `rds:`. Vous utilisez le préfixe `rds-db:` et l'action `rds-db:connect` uniquement pour l'authentification de base de données IAM. Ils ne sont valides que dans ce contexte.

L'exemple de politique inclut une instruction unique avec les éléments suivants :

- `Effect` – Spécifiez `Allow` pour octroyer l'accès au cluster de base de données. Si vous n'autorisez pas explicitement l'accès, celui-ci est refusé par défaut.
- `Action` – Spécifiez `rds-db:connect` pour autoriser les connexions au cluster de base de données.
- `Resource` – Spécifiez un ARN (Amazon Resource Name) qui décrit un compte de base de données dans un cluster de base de données. Le format de l'ARN est le suivant.

```
arn:aws:rds-db:region:account-id:dbuser:DbClusterResourceId/db-user-name
```

  Dans ce format, remplacez les variables suivantes :

  - *`region`*est la AWS région du cluster d' de base de données. Dans l'exemple de politique, la AWS région est`us-east-2`.
  - *`account-id`*est le numéro de AWS compte du cluster d' de base de données. Dans l'exemple de stratégie, le numéro de compte est `1234567890`. L'utilisateur doit figurer dans le même compte que le compte du cluster de base de données.

    Pour bénéficier d'un accès intercompte, créez un rôle IAM avec la politique décrite ci-dessus dans le compte du cluster de base de données et autorisez votre autre compte à endosser ce rôle.
  - *`DbClusterResourceId`* correspond à l'identifiant du cluster de base de données. Cet identifiant est propre à une AWS région et ne change jamais. Dans cet exemple de stratégie, l'identifiant est `cluster-ABCDEFGHIJKL01234`.

    Pour trouver un ID de ressource de cluster d' de base de AWS Management Console données dans Amazon Aurora, choisissez le cluster d' de base de données pour en voir les détails.

Choisissez ensuite l'onglet Configuration. L'ID de ressource est indiqué dans la section Configuration.

Vous pouvez également utiliser la AWS CLI commande pour répertorier les identifiants et les ressources IDs de l'ensemble de votre cluster d' de base de données dans la AWS région actuelle, comme indiqué ci-dessous.

```
aws rds describe-db-clusters --query "DBClusters[*].
[DBClusterIdentifier,DbClusterResourceId]"
```

> **Note**
>
> Si vous vous connectez à une base de données via le proxy RDS, spécifiez l'ID de ressource de proxy, par exemple `prx-ABCDEFGHIJKL01234`. Pour plus d'informations sur l'utilisation de l'authentification de base de données IAM avec le proxy RDS, consultez Connexion à une base de données à l'aide de l'authentification IAM.

- *`db-user-name`* correspond au nom du compte de base de données à associer à l'authentification IAM. Dans cet exemple de stratégie, le compte de la base de données est `db_user`.

Vous pouvez en créer d'autres ARNs pour prendre en charge différents modèles d'accès. La stratégie suivante permet d'accéder à deux comptes de base de données différents dans un cluster de base de données.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
```

```
            "arn:aws:rds-db:us-east-2:123456789012:dbuser:db-ABCDEFGHIJKL01234/
jane_doe",
            "arn:aws:rds-db:us-east-2:123456789012:dbuser:db-ABCDEFGHIJKL01234/
mary_roe"
         ]
      }
   ]
}
```

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-
ABCDEFGHIJKL01234/jane_doe",
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-
ABCDEFGHIJKL01234/mary_roe"
         ]
      }
   ]
}
```

La politique suivante utilise le caractère « * » pour faire correspondre tous les clusters d' de base de données et les comptes de base de données pour un AWS compte et une AWS région spécifiques.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "rds-db:connect"
            ],
            "Resource": [
                "arn:aws:rds-db:us-east-2:111122223333:dbuser:*/*"
            ]
        }
    ]
}
```

La politique suivante correspond à tous les clusters d' de base de données pour un AWS compte et une AWS région particuliers. Néanmoins, cette stratégie n'octroie l'accès qu'aux et clusters de bases de données qui ont un compte de base de données `jane_doe`.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
             "rds-db:connect"
         ],
         "Resource": [
             "arn:aws:rds-db:us-east-2:123456789012:dbuser:*/jane_doe"
         ]
      }
   ]
}
```

L'utilisateur ou le rôle a uniquement accès aux bases de données auxquelles l'utilisateur de base de données a accès. Supposons par exemple que votre cluster de base de donnés possède une base de données nommée dev et une autre base de données nommée test. Si l'utilisateur de base de données `jane_doe` a uniquement accès à dev, tous les rôles ou utilisateurs qui accèdent à ce cluster de bases de données avec l'utilisateur `jane_doe` ont aussi uniquement accès à dev. Cette

restriction d'accès s'applique également aux autres objets de bases de données, tels que les tables, les vues, etc.

Un administrateur doit créer des politiques IAM autorisant les entités à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. L'administrateur doit ensuite attacher ces politiques aux jeux d'autorisations ou aux rôles qui ont besoin de ces autorisations. Pour obtenir des exemples de stratégies, consultez la section Exemples de politiques basées sur l'identité pour Amazon Aurora.

Attacher une politique IAM à un jeu d'autorisations ou à un rôle

Après avoir créé une politique IAM pour permettre l'authentification d'une base de données, il convient d'attacher la politique à un jeu d'autorisations ou à un rôle. Pour accéder à un didacticiel sur ce sujet, veuillez consulter Créer et attacher votre première politique gérée par le client dans le Guide de l'utilisateur IAM.

Tandis que vous parcourez ce didacticiel, vous pouvez utiliser un exemple de politique illustré dans cette section comme point de départ afin de le personnaliser en fonction de vos besoins. À la fin de ce didacticiel, vous obtenez un jeu d'autorisations avec une politique attachée qui peut utiliser l'action `rds-db:connect`.

> **Note**
>
> Vous pouvez mapper plusieurs jeux d'autorisations ou rôles au même compte d'utilisateur de base de données. Supposons par exemple que votre politique IAM a spécifié l'ARN de ressource suivant.
>
> ```
> arn:aws:rds-db:us-east-2:123456789012:dbuser:cluster-12ABC34DEFG5HIJ6KLMNOP78QR/
> jane_doe
> ```
>
> Si vous attachez la politique à Jane, Bob et Diego, chacun de ces utilisateurs peut se connecter au cluster de bases de données en utilisant le compte de base de données `jane_doe`.

Création d’un compte de base de données à l’aide de l’authentification IAM

Avec l’authentification de base de données IAM, vous n’avez pas besoin d’associer de mots de passe de base de données aux comptes d’utilisateurs que vous créez. Si vous supprimez un utilisateur qui est mappé à un compte de base de données, vous devez également supprimer le compte de base de données avec l’instruction `DROP USER`.

> **Note**
>
> Le nom d’utilisateur utilisé pour l’authentification IAM doit correspondre à la casse du nom d’utilisateur dans la base de données.

Rubriques

- Utilisation de l’authentification IAM avec Aurora MySQL
- Utilisation de l’authentification IAM avec Aurora PostgreSQL

Utilisation de l’authentification IAM avec Aurora MySQL

Avec Aurora MySQL, l’authentification est gérée par `AWSAuthenticationPlugin`, un plugin fourni par AWS qui fonctionne de manière transparente avec IAM pour authentifier vos utilisateurs. Connectez-vous au cluster de bases de données en tant qu’utilisateur principal ou autre utilisateur qui peut créer des utilisateurs et accorder des privilèges. Après vous être connecté, exécutez l’instruction `CREATE USER`, comme indiqué dans l’exemple suivant.

```
CREATE USER 'jane_doe' IDENTIFIED WITH AWSAuthenticationPlugin AS 'RDS';
```

La clause `IDENTIFIED WITH` permet à Aurora MySQL d’utiliser `AWSAuthenticationPlugin` pour authentifier le compte de base de données (`jane_doe`). La clause `AS 'RDS'` fait référence à la méthode d’authentification. Assurez-vous que le nom d’utilisateur de base de données spécifié est identique à une ressource dans la politique IAM pour l’accès à la base de données IAM. Pour plus d’informations, consultez Création et utilisation d'une politique IAM pour l'accès à une base de données IAM.

> **Note**
>
> Si vous voyez le message suivant, cela signifie que le plugin fourni par AWS n'est pas disponible pour le cluster de bases de données.
> `ERROR 1524 (HY000): Plugin 'AWSAuthenticationPlugin' is not loaded`
> Pour remédier à cette erreur, vérifiez si vous utilisez une configuration prise en charge et si vous avez activé l'authentification de base de données IAM sur votre cluster de bases de données. Pour plus d'informations, consultez Disponibilité des régions et des versions et Activation et désactivation de l'authentification de base de données IAM.

Après avoir créé un compte à l'aide de `AWSAuthenticationPlugin`, vous pouvez le gérer de la même manière que les autres comptes de base de données. Vous pouvez par exemple modifier les privilèges de compte avec `GRANT` et `REVOKE`, ou changer divers attributs de compte avec l'instruction `ALTER USER`.

Le trafic réseau de base de données est chiffré à l'aide de SSL/TLS lors de l'utilisation d'IAM. Pour autoriser les connexions SSL, modifiez le compte d'utilisateur à l'aide de la commande suivante.

```
ALTER USER 'jane_doe'@'%' REQUIRE SSL;
```

Utilisation de l'authentification IAM avec Aurora PostgreSQL

Pour utiliser l'authentification IAM avec Aurora PostgreSQL, connectez-vous au cluster de bases de données en tant qu'utilisateur principal ou autre utilisateur qui peut créer des utilisateurs et accorder des privilèges. Après vous être connecté, créez des utilisateurs de base de données, puis accordez-leur le rôle `rds_iam`, comme indiqué dans l'exemple suivant.

```
CREATE USER db_userx;
GRANT rds_iam TO db_userx;
```

Assurez-vous que le nom d'utilisateur de base de données spécifié est identique à une ressource dans la politique IAM pour l'accès à la base de données IAM. Pour plus d'informations, consultez Création et utilisation d'une politique IAM pour l'accès à une base de données IAM. Vous devez accorder le rôle `rds_iam` pour utiliser l'authentification IAM. Vous pouvez également utiliser des appartenances imbriquées ou des octrois indirects du rôle.

Notez qu'un utilisateur de base de données PostgreSQL peut utiliser l'authentification IAM ou Kerberos, mais pas les deux, si bien que cet utilisateur ne peut pas avoir le rôle `rds_ad`. Cela s'applique également aux adhésions imbriquées. Pour plus d'informations, consultez Étape 7 : Créer des utilisateurs PostgreSQL pour vos principaux Kerberos .

Connexion à votre cluster de base de données à l'aide de l'authentification IAM.

Avec l'authentification de base de données IAM, vous utilisez un jeton d'identification lors de la connexion à votre cluster de base de données. Un jeton d'authentification constitue une chaîne de caractères unique qui remplace un mot de passe. Après avoir été créé, un jeton d'authentification est valable pendant 15 minutes avant d'expirer. Si vous tentez de vous connecter alors que le jeton expiré, la demande de connexion est rejetée.

Chaque jeton d'authentification doit être accompagné d'une signature valide, en utilisant AWS Signature Version 4. (Pour plus d'informations, consultez Processus de signature Signature version 4 dans la Références générales AWS.) L'AWS CLI et un kit SDK AWS tel que AWS SDK pour Java ou AWS SDK pour Python (Boto3) peuvent signer automatiquement chaque jeton que vous créez.

Vous pouvez utiliser un jeton d'authentification lorsque vous vous connectez à Amazon Aurora à partir d'un autre service AWS tel que AWS Lambda. L'utilisation d'un jeton vous évite d'avoir à placer un mot de passe dans votre code. Vous pouvez aussi utiliser un kit SDK AWS pour créer et signer un jeton d'authentification par programmation.

Après avoir obtenu un jeton d'authentification IAM signé, vous pouvez vous connecter à un cluster de bases de données Aurora. Utilisez les liens ci-dessous pour savoir comment procéder avec un outil de ligne de commande ou un kit SDK AWS, comme le AWS SDK pour Java ou AWS SDK pour Python (Boto3).

Pour plus d'informations, consultez les billets de blog suivants :

- Utilisation de l'authentification IAM pour se connecter avec SQL Workbench/J à Aurora MySQL ou Amazon RDS for MySQL
- Utilisation de l'authentification IAM pour se connecter à PgAdmin Amazon Aurora PostgreSQL ou Amazon RDS for PostgreSQL

Prérequis

Les conditions préalables à la connexion à votre cluster de base de données à l'aide de l'authentification IAM sont les suivantes :

- Activation et désactivation de l’authentification de base de données IAM
- Création et utilisation d'une politique IAM pour l'accès à une base de données IAM
- Création d’un compte de base de données à l’aide de l’authentification IAM

Rubriques

- Connexion à votre cluster de bases de données à l’aide de l’authentification IAM avec les pilotes AWS
- Connexion à votre cluster de base de données à l'aide de l'authentification IAM à partir de la ligne de commande : AWS CLI et client mysql
- Connexion à votre cluster de base de données à l'aide de l'authentification IAM à partir de la ligne de commande : AWS CLI et client psql
- Connexion à votre cluster de base de données à l'aide de l'authentification IAM et de AWS SDK pour .NET
- Connexion à votre cluster de base de données à l'aide de l'authentification IAM et de AWS SDK pour Go
- Connexion à votre cluster de bases de données à l’aide de l’authentification IAM et de AWS SDK pour Java
- Connexion à votre cluster de base de données à l'aide de l'authentification IAM et de AWS SDK pour Python (Boto3)

Connexion à votre cluster de bases de données à l’aide de l’authentification IAM avec les pilotes AWS

La suite de pilotes AWS a été conçue pour accélérer les temps de bascule et de basculement, ainsi que pour l’authentification avec AWS Secrets Manager, Gestion des identités et des accès AWS (IAM) et l’identité fédérée. Les pilotes AWS s’appuient sur la surveillance de l’état du cluster de bases de données et sur la connaissance de la topologie du cluster pour déterminer le nouvel enregistreur. Cette approche réduit les temps de bascule et de basculement à moins de 10 secondes, contre des dizaines de secondes pour les pilotes open source.

Pour plus d’informations sur les pilotes AWS, consultez le pilote correspondant au langage utilisé pour votre cluster de bases de données Aurora MySQL ou Aurora PostgreSQL.

Connexion à votre cluster de base de données à l'aide de l'authentification IAM à partir de la ligne de commande : AWS CLI et client mysql

À partir de la ligne de commande, vous pouvez vous connecter à une cluster de bases de données Aurora avec l'AWS CLI et l'outil de ligne de commande `mysql` en procédant comme suit.

Prérequis

Les conditions préalables à la connexion à votre cluster de base de données à l'aide de l'authentification IAM sont les suivantes :

- Activation et désactivation de l’authentification de base de données IAM
- Création et utilisation d'une politique IAM pour l'accès à une base de données IAM
- Création d’un compte de base de données à l’aide de l’authentification IAM

> **Note**
>
> Pour plus d'informations sur la connexion à votre base de données à l'aide de SQL Workbench/J avec authentification IAM, lisez le billet de blog Utilisation de l'authentification IAM pour se connecter avec SQL Workbench/J à Aurora MySQL ou Amazon RDS for MySQL.

Rubriques

- Création d'un jeton d'authentification IAM
- Connexion à votre cluster de base de données

Création d'un jeton d'authentification IAM

L'exemple suivant illustre comment obtenir un jeton d'identification signé à l'aide d AWS CLI.

```
aws rds generate-db-auth-token \
   --hostname rdsmysql.123456789012.us-west-2.rds.amazonaws.com \
   --port 3306 \
   --region us-west-2 \
   --username jane_doe
```

Dans cet exemple, les paramètres sont les suivants :

- `--hostname` – Le nom d'hôte du cluster de base de données auquel vous souhaitez accéder.
- `--port` – Le numéro du port utilisé lors de la connexion au cluster d' de base de données.
- `--region` – La région AWS où l'cluster de base de données s'exécute.
- `--username` – Le compte de base de données auquel vous souhaitez accéder.

Les premiers caractères du jeton ressemblent à l'exemple suivant.

```
rdsmysql.123456789012.us-west-2.rds.amazonaws.com:3306/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> Vous ne pouvez pas utiliser un enregistrement DNS Route 53 personnalisé ni un point de terminaison personnalisé Aurora à la place du point de terminaison du cluster de bases de données pour générer le jeton d'authentification.

Connexion à votre cluster de base de données

Le format général de connexion est illustré ci-dessous.

```
mysql --host=hostName --port=portNumber --ssl-ca=full_path_to_ssl_certificate --enable-
cleartext-plugin --user=userName --password=authToken
```

Les paramètres sont les suivants :

- `--host` – Le nom d'hôte du cluster de base de données auquel vous souhaitez accéder.
- `--port` – Le numéro du port utilisé lors de la connexion au cluster d' de base de données.
- `--ssl-ca` – Le chemin d'accès complet vers le fichier de certificat SSL contenant la clé publique.

  Pour de plus amples informations, consultez Connexions TLS aux clusters de bases de données Aurora MySQL.

  Pour télécharger un certificat SSL, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.
- `--enable-cleartext-plugin` – Une valeur qui spécifie que `AWSAuthenticationPlugin` doit être utilisé pour cette connexion.

Si vous utilisez un client MariaDB, l'option `--enable-cleartext-plugin` n'est pas requise.

- `--user` – Le compte de base de données auquel vous souhaitez accéder.
- `--password` – Un jeton d'authentification IAM signé.

Le jeton d'authentification est composé de plusieurs centaines de caractères. Il peut être encombrant sur la ligne de commande. Pour contourner ce problème, vous pouvez enregistrer le jeton dans une variable d'environnement, puis utiliser cette variable pour la connexion. L'exemple suivant illustre une manière de contourner ce problème. Dans cet exemple, */sample_dir/* est le chemin d'accès complet au fichier de certificat SSL contenant la clé publique.

```
RDSHOST="mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
TOKEN="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 3306 --region us-
west-2 --username jane_doe )"

mysql --host=$RDSHOST --port=3306 --ssl-ca=/sample_dir/global-bundle.pem --enable-
cleartext-plugin --user=jane_doe --password=$TOKEN
```

Lorsque vous vous connectez avec `AWSAuthenticationPlugin`, la connexion est sécurisée par SSL. Pour le vérifier, tapez la commande suivante à l'invite de commande `mysql>`.

```
show status like 'Ssl%';
```

Les lignes suivantes de l'affichage obtenu fournissent plus de détails.

```
+---------------+-------------+
| Variable_name | Value

                                                                    |
+---------------+-------------+
| ...           | ...
| Ssl_cipher    | AES256-SHA

                                                                    |
| ...           | ...
| Ssl_version   | TLSv1.1

                                                                    |
| ...           | ...
```

```
+----------------------------+
```

Si vous souhaitez vous connecter à un cluster de bases de données via un proxy, consultez Connexion à une base de données à l'aide de l'authentification IAM.

Connexion à votre cluster de base de données à l'aide de l'authentification IAM à partir de la ligne de commande : AWS CLI et client psql

À partir de la ligne de commande, vous pouvez vous connecter à une cluster de base de données Aurora PostgreSQL avec AWS CLIl'outil de ligne de commande psql comme décrit ci-après.

Prérequis

Les conditions préalables à la connexion à votre cluster de base de données à l'aide de l'authentification IAM sont les suivantes :

- Activation et désactivation de l’authentification de base de données IAM
- Création et utilisation d'une politique IAM pour l'accès à une base de données IAM
- Création d’un compte de base de données à l’aide de l’authentification IAM

> **Note**
>
> Pour plus d'informations sur la connexion à votre base de données à l'aide de pgAdmin avec authentification IAM, consultez le billet de blog Utilisation de l'authentification IAM pour se connecter à PgAdmin Amazon Aurora PostgreSQL ou Amazon RDS for PostgreSQL

Rubriques

- Création d'un jeton d'authentification IAM
- Connexion à une un cluster Aurora PostgreSQL

Création d'un jeton d'authentification IAM

Le jeton d'authentification se compose de plusieurs centaines de caractères ; il peut donc être complexe à manipuler sur la ligne de commande. Pour contourner ce problème, vous pouvez enregistrer le jeton dans une variable d'environnement, puis utiliser cette variable pour la connexion. L'exemple de code suivant montre comment utiliser l'AWS CLI pour obtenir un jeton d'authentification

signé à l'aide de la commande `generate-db-auth-token` et le stocker dans une variable d'environnement `PGPASSWORD`.

```
export RDSHOST="mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"
```

Dans cet exemple, les paramètres de la commande `generate-db-auth-token` sont les suivants :

- `--hostname` – Nom d'hôte de l'du cluster (point de terminaison du cluster) de base de données auquel vous souhaitez accéder.
- `--port` – Le numéro du port utilisé lors de la connexion au cluster d' de base de données.
- `--region` – La région AWS où l'cluster de base de données s'exécute.
- `--username` – Le compte de base de données auquel vous souhaitez accéder.

Les premiers caractères du jeton généré ressemblent à l'exemple suivant.

```
mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com:5432/?
Action=connect&DBUser=jane_doe&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Expires=900...
```

> **Note**
>
> Vous ne pouvez pas utiliser un enregistrement DNS Route◦53 personnalisé◦ni un point de terminaison personnalisé Aurora◦à la place du point de terminaison du cluster de bases de données◦pour générer le jeton d'authentification.

Connexion à une un cluster Aurora PostgreSQL

Le format général pour utiliser psql pour la connexion est illustré ci-dessous.

```
psql "host=hostName port=portNumber sslmode=verify-full
 sslrootcert=full_path_to_ssl_certificate dbname=DBName user=userName
 password=authToken"
```

Les paramètres sont les suivants :

- `host` – Nom d'hôte de l'du cluster (point de terminaison du cluster) de base de données auquel vous souhaitez accéder.
- `port` – Le numéro du port utilisé lors de la connexion au cluster d' de base de données.
- `sslmode` – Le mode SSL à utiliser.

  Lorsque vous utilisez `sslmode=verify-full`, la connexion SSL vérifie le point de terminaison de l'cluster de base de données par rapport au point de terminaison dans le certificat SSL.
- `sslrootcert` – Le chemin d'accès complet vers le fichier de certificat SSL contenant la clé publique.

  Pour de plus amples informations, consultez Sécurisation des données Aurora PostgreSQL avec SSL/TLS.

  Pour télécharger un certificat SSL, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.
- `dbname` – La base de données à laquelle vous souhaitez accéder.
- `user` – Le compte de base de données auquel vous souhaitez accéder.
- `password` – Un jeton d'authentification IAM signé.

> **Note**
>
> Vous ne pouvez pas utiliser un enregistrement DNS Route 53 personnalisé ni un point de terminaison personnalisé Aurora à la place du point de terminaison du cluster de bases de données pour générer le jeton d'authentification.

L'exemple suivant montre l'utilisation de psql pour se connecter. Dans cet exemple, psql utilise la variable d'environnement `RDSHOST` pour l'hôte et la variable d'environnement `PGPASSWORD` pour le jeton généré. Par ailleurs, *`/sample_dir/`* est le chemin d'accès complet au fichier de certificat SSL contenant la clé publique.

```
export RDSHOST="mypostgres-cluster.cluster-123456789012.us-west-2.rds.amazonaws.com"
export PGPASSWORD="$(aws rds generate-db-auth-token --hostname $RDSHOST --port 5432 --
region us-west-2 --username jane_doe )"

psql "host=$RDSHOST port=5432 sslmode=verify-full sslrootcert=/sample_dir/global-
bundle.pem dbname=DBName user=jane_doe password=$PGPASSWORD"
```

Si vous souhaitez vous connecter à un cluster de bases de données via un proxy, consultez Connexion à une base de données à l'aide de l'authentification IAM.

Connexion à votre cluster de base de données à l'aide de l'authentification IAM et de AWS SDK pour .NET

Vous pouvez vous connecter à un cluster de bases de données Aurora MySQL ou Aurora PostgreSQL avec l'AWS SDK pour .NET, comme décrit ci-après.

Prérequis

Les conditions préalables à la connexion à votre cluster de base de données à l'aide de l'authentification IAM sont les suivantes :

- Activation et désactivation de l’authentification de base de données IAM
- Création et utilisation d'une politique IAM pour l'accès à une base de données IAM
- Création d’un compte de base de données à l’aide de l’authentification IAM

Exemples

Les exemples de code suivants montre comment générer un jeton d'authentification, puis comment l'utiliser pour se connecter à un cluster de base de données.

Pour exécuter cet exemple de code, vous avez besoin de AWS SDK pour .NET, disponible sur le site AWS. Les paquets `AWSSDK.CORE` et `AWSSDK.RDS` sont requis. Pour vous connecter à un(e) cluster de base de données, utilisez le connecteur de base de données .NET pour le moteur de base de données, tel que MySQLConnector pour MariaDB ou MySQL, ou Npgsql pour PostgreSQL.

Ce code se connecte à un cluster de base de données Aurora MySQL. Modifiez la valeur des variables suivantes selon les besoins :

- `server` – Le point de terminaison de cluster de base de données à laquelle vous souhaitez accéder.
- `user` – Le compte de base de données auquel vous souhaitez accéder.
- `database` – La base de données à laquelle vous souhaitez accéder.
- `port` – Le numéro du port utilisé lors de la connexion au cluster d' de base de données.
- `SslMode` – Le mode SSL à utiliser.

Lorsque vous utilisez `SslMode=Required`, la connexion SSL vérifie le point de terminaison de l'cluster de base de données par rapport au point de terminaison dans le certificat SSL.

- `SslCa` – Le chemin d'accès complet au certificat SSL pour Amazon Aurora

  Pour télécharger un certificat, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.

> **Note**
>
> Vous ne pouvez pas utiliser un enregistrement DNS Route 53 personnalisé ni un point de terminaison personnalisé Aurora à la place du point de terminaison du cluster de bases de données pour générer le jeton d'authentification.

```
using System;
using System.Data;
using MySql.Data;
using MySql.Data.MySqlClient;
using Amazon;

namespace ubuntu
{
  class Program
  {
    static void Main(string[] args)
    {
      var pwd =
 Amazon.RDS.Util.RDSAuthTokenGenerator.GenerateAuthToken(RegionEndpoint.USEast1,
 "mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com", 3306, "jane_doe");
      // for debug only Console.Write("{0}\n", pwd);  //this verifies the token is
 generated

      MySqlConnection conn = new
 MySqlConnection($"server=mysqlcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com;user=jane_doe;database=mydB;port=3306;password={pwd};SslMode=Required;
      conn.Open();

      // Define a query
      MySqlCommand sampleCommand = new MySqlCommand("SHOW DATABASES;", conn);
```

```
      // Execute a query
      MySqlDataReader mysqlDataRdr = sampleCommand.ExecuteReader();

      // Read all rows and output the first column in each row
      while (mysqlDataRdr.Read())
        Console.WriteLine(mysqlDataRdr[0]);

      mysqlDataRdr.Close();
      // Close connection
      conn.Close();
    }
  }
}
```

Ce code se connecte à un cluster de base de données Aurora PostgreSQL.

Modifiez la valeur des variables suivantes selon les besoins :

- `Server` – Le point de terminaison de cluster de base de données à laquelle vous souhaitez accéder.
- `User ID` – Le compte de base de données auquel vous souhaitez accéder.
- `Database` – La base de données à laquelle vous souhaitez accéder.
- `Port` – Le numéro du port utilisé lors de la connexion au cluster d' de base de données.
- `SSL Mode` – Le mode SSL à utiliser.

  Lorsque vous utilisez `SSL Mode=Required`, la connexion SSL vérifie le point de terminaison de l'cluster de base de données par rapport au point de terminaison dans le certificat SSL.
- `Root Certificate` – Le chemin d'accès complet au certificat SSL pour Amazon Aurora

  Pour télécharger un certificat, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.

> **Note**
>
> Vous ne pouvez pas utiliser un enregistrement DNS Route 53 personnalisé ni un point de terminaison personnalisé Aurora à la place du point de terminaison du cluster de bases de données pour générer le jeton d'authentification.

```
using System;
using Npgsql;
using Amazon.RDS.Util;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            var pwd =
 RDSAuthTokenGenerator.GenerateAuthToken("postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com", 5432, "jane_doe");
// for debug only Console.Write("{0}\n", pwd);  //this verifies the token is generated

            NpgsqlConnection conn = new
 NpgsqlConnection($"Server=postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com;User Id=jane_doe;Password={pwd};Database=mydb;SSL
 Mode=Require;Root Certificate=full_path_to_ssl_certificate");
            conn.Open();

            // Define a query
                   NpgsqlCommand cmd = new NpgsqlCommand("select count(*) FROM
 pg_user", conn);

            // Execute a query
            NpgsqlDataReader dr = cmd.ExecuteReader();

            // Read all rows and output the first column in each row
            while (dr.Read())
                Console.Write("{0}\n", dr[0]);

            // Close connection
            conn.Close();
        }
    }
}
```

Si vous souhaitez vous connecter à un cluster de bases de données via un proxy, consultez Connexion à une base de données à l'aide de l'authentification IAM.

Connexion à votre cluster de base de données à l'aide de l'authentification IAM et de AWS SDK pour Go

Vous pouvez vous connecter à un cluster de bases de données Aurora MySQL ou Aurora PostgreSQL avec l'AWS SDK pour Go, comme décrit ci-après.

Prérequis

Les conditions préalables à la connexion à votre cluster de base de données à l'aide de l'authentification IAM sont les suivantes :

- Activation et désactivation de l’authentification de base de données IAM
- Création et utilisation d'une politique IAM pour l'accès à une base de données IAM
- Création d’un compte de base de données à l’aide de l’authentification IAM

Exemples

Pour exécuter cet exemple de code, vous avez besoin de AWS SDK pour Go, disponible sur le site AWS.

Modifiez la valeur des variables suivantes selon les besoins :

- `dbName` – La base de données à laquelle vous souhaitez accéder.
- `dbUser` – Le compte de base de données auquel vous souhaitez accéder.
- `dbHost` – Le point de terminaison de cluster de base de données à laquelle vous souhaitez accéder.

> **Note**
>
> Vous ne pouvez pas utiliser un enregistrement DNS Route 53 personnalisé ni un point de terminaison personnalisé Aurora à la place du point de terminaison du cluster de bases de données pour générer le jeton d'authentification.

- `dbPort` – Le numéro du port utilisé lors de la connexion au cluster d' de base de données.
- `region` – La région AWS où l'cluster de base de données s'exécute.

En outre, assurez-vous que les bibliothèques importées dans l'exemple de code existent sur votre système.

> **Important**
>
> Les exemples de cette section utilisent le code suivant pour fournir des informations d'identification qui accèdent à une base de données à partir d'un environnement local :
> `creds := credentials.NewEnvCredentials()`
> Si vous accédez à une base de données à partir d'un service AWS, tel que Amazon EC2 ou Amazon ECS, vous pouvez remplacer le code par le code suivant :
> `sess := session.Must(session.NewSession())`
> `creds := sess.Config.Credentials`
> Si vous effectuez cette modification, assurez-vous d'ajouter l'importation suivante :
> `"github.com/aws/aws-sdk-go/aws/session"`

Rubriques

- Connexion à l'aide de l'authentification IAM et de AWS SDK pour Go V2
- Connexion à l'aide de l'authentification IAM et de AWS SDK pour Go V1.

Connexion à l'aide de l'authentification IAM et de AWS SDK pour Go V2

Vous pouvez vous connecter à un cluster de base de données à l'aide de l'authentification IAM et de AWS SDK pour Go V2.

Les exemples de code suivants montre comment générer un jeton d'authentification, puis comment l'utiliser pour se connecter à un cluster de base de données.

Ce code se connecte à un cluster de base de données Aurora MySQL.

```
package main

import (
     "context"
     "database/sql"
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/go-sql-driver/mysql"
)
```

```
func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "mysqlcluster.cluster-123456789012.us-
east-1.rds.amazonaws.com"
     var dbPort int = 3306
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authenticationToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Ce code se connecte à un cluster de base de données Aurora PostgreSQL.

```
package main

import (
     "context"
     "database/sql"
```

```
     "fmt"

     "github.com/aws/aws-sdk-go-v2/config"
     "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
     _ "github.com/lib/pq"
)

func main() {

     var dbName string = "DatabaseName"
     var dbUser string = "DatabaseUser"
     var dbHost string = "postgresmycluster.cluster-123456789012.us-
east-1.rds.amazonaws.com"
     var dbPort int = 5432
     var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
     var region string = "us-east-1"

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
     panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
     context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
     panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authenticationToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Si vous souhaitez vous connecter à un cluster de bases de données via un proxy, consultez Connexion à une base de données à l'aide de l'authentification IAM.

Connexion à l'aide de l'authentification IAM et de AWS SDK pour Go V1.

Vous pouvez vous connecter à un cluster de base de données à l'aide de l'authentification IAM et de AWS SDK pour Go V1

Les exemples de code suivants montre comment générer un jeton d'authentification, puis comment l'utiliser pour se connecter à un cluster de base de données.

Ce code se connecte à un cluster de base de données Aurora MySQL.

```
package main

import (
    "database/sql"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go/aws/credentials"
    "github.com/aws/aws-sdk-go/service/rds/rdsutils"
    _ "github.com/go-sql-driver/mysql"
)

func main() {
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 3306
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authToken, dbEndpoint, dbName,
    )
```

```
    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Ce code se connecte à un cluster de base de données Aurora PostgreSQL.

```
package main

import (
 "database/sql"
 "fmt"

 "github.com/aws/aws-sdk-go/aws/credentials"
 "github.com/aws/aws-sdk-go/service/rds/rdsutils"
 _ "github.com/lib/pq"
)

func main() {
    dbName := "app"
    dbUser := "jane_doe"
    dbHost := "postgresmycluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
    dbPort := 5432
    dbEndpoint := fmt.Sprintf("%s:%d", dbHost, dbPort)
    region := "us-east-1"

    creds := credentials.NewEnvCredentials()
    authToken, err := rdsutils.BuildAuthToken(dbEndpoint, region, dbUser, creds)
    if err != nil {
        panic(err)
    }

    dsn := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s",
        dbHost, dbPort, dbUser, authToken, dbName,
    )

    db, err := sql.Open("postgres", dsn)
```

```
    if err != nil {
        panic(err)
    }

    err = db.Ping()
    if err != nil {
        panic(err)
    }
}
```

Si vous souhaitez vous connecter à un cluster de bases de données via un proxy, consultez Connexion à une base de données à l'aide de l'authentification IAM.

Connexion à votre cluster de bases de données à l’aide de l’authentification IAM et de AWS SDK pour Java

Vous pouvez vous connecter à un cluster de bases de données Aurora MySQL ou Aurora PostgreSQL avec l’AWS SDK pour Java, comme décrit ci-après.

Prérequis

Les conditions préalables à la connexion à votre cluster de base de données à l’aide de l’authentification IAM sont les suivantes :

- Activation et désactivation de l’authentification de base de données IAM
- Création et utilisation d'une politique IAM pour l'accès à une base de données IAM
- Création d’un compte de base de données à l’aide de l’authentification IAM
- Configuration du kit SDK AWS pour Java

Pour des exemples d’utilisation du kit SDK pour Java 2.x, consultez Exemples Amazon RDS utilisant le kit SDK pour Java 2.x. Vous pouvez également utiliser l’AWSAdvanced JDBC Wrapper, consultez la documentation AWSAdvanced JDBC Wrapper.

Rubriques

- Création d’un jeton d’authentification IAM
- Construction manuelle d’un jeton d’authentification IAM
- Connexion à votre cluster de bases de données

Création d'un jeton d'authentification IAM

Si vous écrivez des programmes à l'aide de AWS SDK pour Java, vous pouvez obtenir un jeton d'authentification signé au moyen de la classe `RdsIamAuthTokenGenerator`. L'usage de cette classe exige que vous fournissiez les informations d'identification AWS. Pour ce faire, vous devez créer une instance de la classe `DefaultAWSCredentialsProviderChain`. `DefaultAWSCredentialsProviderChain` utilise la première clé d'accès AWS et la clé secrète trouvée dans la chaîne de fournisseur d'informations d'identification par défaut. Pour plus d'informations sur les clés d'accès AWS, consultez Gestion des clés d'accès pour les utilisateurs IAM.

> **Note**
>
> Vous ne pouvez pas utiliser un enregistrement DNS Route 53 personnalisé ni un point de terminaison personnalisé Aurora à la place du point de terminaison du cluster de bases de données pour générer le jeton d'authentification.

Après avoir créé une instance de `RdsIamAuthTokenGenerator`, vous pouvez appeler la méthode `getAuthToken` pour obtenir un jeton signé. Fournissez la région AWS, le nom d'hôte, le numéro de port et le nom d'utilisateur. L'exemple de code suivant montre comment procéder.

```
package com.amazonaws.codesamples;

import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;

public class GenerateRDSAuthToken {

    public static void main(String[] args) {

     String region = "us-west-2";
     String hostname = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
     String port = "3306";
     String username = "jane_doe";

     System.out.println(generateAuthToken(region, hostname, port, username));
    }
```

```
    static String generateAuthToken(String region, String hostName, String port, String
 username) {

     RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
      .credentials(new DefaultAWSCredentialsProviderChain())
      .region(region)
      .build();

     String authToken = generator.getAuthToken(
      GetIamAuthTokenRequest.builder()
      .hostname(hostName)
      .port(Integer.parseInt(port))
      .userName(username)
      .build());

     return authToken;
    }

}
```

Construction manuelle d'un jeton d'authentification IAM

Dans Java, la manière la plus facile de créer un jeton d'authentification est d'utiliser `RdsIamAuthTokenGenerator`. Cette classe créé un jeton d'authentification pour vous, puis le signe à l'aide d'AWS Signature Version 4. Pour plus d'informations, consultez Processus de signature Signature Version 4 dans le Références générales AWS.

Vous pouvez néanmoins aussi construire et signer un jeton d'authentification manuellement, comme indiqué dans l'exemple de code suivant.

```
package com.amazonaws.codesamples;

import com.amazonaws.SdkClientException;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.SigningAlgorithm;
import com.amazonaws.util.BinaryUtils;
import org.apache.commons.lang3.StringUtils;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.Charset;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
```

```
import java.util.Date;
import java.util.SortedMap;
import java.util.TreeMap;

import static com.amazonaws.auth.internal.SignerConstants.AWS4_TERMINATOR;
import static com.amazonaws.util.StringUtils.UTF8;

public class CreateRDSAuthTokenManually {
    public static String httpMethod = "GET";
    public static String action = "connect";
    public static String canonicalURIParameter = "/";
    public static SortedMap<String, String> canonicalQueryParameters = new TreeMap();
    public static String payload = StringUtils.EMPTY;
    public static String signedHeader = "host";
    public static String algorithm = "AWS4-HMAC-SHA256";
    public static String serviceName = "rds-db";
    public static String requestWithoutSignature;

    public static void main(String[] args) throws Exception {

        String region = "us-west-2";
        String instanceName = "rdsmysql.123456789012.us-west-2.rds.amazonaws.com";
        String port = "3306";
        String username = "jane_doe";

        Date now = new Date();
        String date = new SimpleDateFormat("yyyyMMdd").format(now);
        String dateTimeStamp = new
 SimpleDateFormat("yyyyMMdd'T'HHmmss'Z'").format(now);
       DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
    String awsAccessKey = creds.getCredentials().getAWSAccessKeyId();
    String awsSecretKey = creds.getCredentials().getAWSSecretKey();
       String expiryMinutes = "900";

       System.out.println("Step 1:  Create a canonical request:");
       String canonicalString = createCanonicalString(username, awsAccessKey, date,
 dateTimeStamp, region, expiryMinutes, instanceName, port);
       System.out.println(canonicalString);
       System.out.println();

       System.out.println("Step 2:  Create a string to sign:");
       String stringToSign = createStringToSign(dateTimeStamp, canonicalString,
 awsAccessKey, date, region);
```

```
        System.out.println(stringToSign);
        System.out.println();

        System.out.println("Step 3:  Calculate the signature:");
        String signature = BinaryUtils.toHex(calculateSignature(stringToSign,
newSigningKey(awsSecretKey, date, region, serviceName)));
        System.out.println(signature);
        System.out.println();

        System.out.println("Step 4:  Add the signing info to the request");

        System.out.println(appendSignature(signature));
        System.out.println();

    }

    //Step 1: Create a canonical request date should be in format YYYYMMDD and dateTime
should be in format YYYYMMDDTHHMMSSZ
    public static String createCanonicalString(String user, String accessKey, String
date, String dateTime, String region, String expiryPeriod, String hostName, String
port) throws Exception {
        canonicalQueryParameters.put("Action", action);
        canonicalQueryParameters.put("DBUser", user);
        canonicalQueryParameters.put("X-Amz-Algorithm", "AWS4-HMAC-SHA256");
        canonicalQueryParameters.put("X-Amz-Credential", accessKey + "%2F" + date +
"%2F" + region + "%2F" + serviceName + "%2Faws4_request");
        canonicalQueryParameters.put("X-Amz-Date", dateTime);
        canonicalQueryParameters.put("X-Amz-Expires", expiryPeriod);
        canonicalQueryParameters.put("X-Amz-SignedHeaders", signedHeader);
        String canonicalQueryString = "";
        while(!canonicalQueryParameters.isEmpty()) {
            String currentQueryParameter = canonicalQueryParameters.firstKey();
            String currentQueryParameterValue =
canonicalQueryParameters.remove(currentQueryParameter);
            canonicalQueryString = canonicalQueryString + currentQueryParameter + "=" +
currentQueryParameterValue;
            if (!currentQueryParameter.equals("X-Amz-SignedHeaders")) {
                canonicalQueryString += "&";
            }
        }
        String canonicalHeaders = "host:" + hostName + ":" + port + '\n';
        requestWithoutSignature = hostName + ":" + port + "/?" + canonicalQueryString;

        String hashedPayload = BinaryUtils.toHex(hash(payload));
```

```
        return httpMethod + '\n' + canonicalURIParameter + '\n' + canonicalQueryString
 + '\n' + canonicalHeaders + '\n' + signedHeader + '\n' + hashedPayload;

    }

    //Step 2: Create a string to sign using sig v4
    public static String createStringToSign(String dateTime, String canonicalRequest,
 String accessKey, String date, String region) throws Exception {
        String credentialScope = date + "/" + region + "/" + serviceName + "/
aws4_request";
        return algorithm + '\n' + dateTime + '\n' + credentialScope + '\n' +
 BinaryUtils.toHex(hash(canonicalRequest));

    }

    //Step 3: Calculate signature
    /**
     * Step 3 of the &AWS; Signature version 4 calculation. It involves deriving
     * the signing key and computing the signature. Refer to
     * http://docs.aws.amazon
     * .com/general/latest/gr/sigv4-calculate-signature.html
     */
    public static byte[] calculateSignature(String stringToSign,
                                            byte[] signingKey) {
        return sign(stringToSign.getBytes(Charset.forName("UTF-8")), signingKey,
                SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(byte[] data, byte[] key,
                          SigningAlgorithm algorithm) throws SdkClientException {
        try {
            Mac mac = algorithm.getMac();
            mac.init(new SecretKeySpec(key, algorithm.toString()));
            return mac.doFinal(data);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    public static byte[] newSigningKey(String secretKey,
                                   String dateStamp, String regionName, String
 serviceName) {
```

```
        byte[] kSecret = ("AWS4" + secretKey).getBytes(Charset.forName("UTF-8"));
        byte[] kDate = sign(dateStamp, kSecret, SigningAlgorithm.HmacSHA256);
        byte[] kRegion = sign(regionName, kDate, SigningAlgorithm.HmacSHA256);
        byte[] kService = sign(serviceName, kRegion,
                SigningAlgorithm.HmacSHA256);
        return sign(AWS4_TERMINATOR, kService, SigningAlgorithm.HmacSHA256);
    }

    public static byte[] sign(String stringData, byte[] key,
                      SigningAlgorithm algorithm) throws SdkClientException {
        try {
            byte[] data = stringData.getBytes(UTF8);
            return sign(data, key, algorithm);
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to calculate a request signature: "
                            + e.getMessage(), e);
        }
    }

    //Step 4: append the signature
    public static String appendSignature(String signature) {
        return requestWithoutSignature + "&X-Amz-Signature=" + signature;
    }

    public static byte[] hash(String s) throws Exception {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(s.getBytes(UTF8));
            return md.digest();
        } catch (Exception e) {
            throw new SdkClientException(
                    "Unable to compute hash while signing request: "
                            + e.getMessage(), e);
        }
    }
}
```

Connexion à votre cluster de bases de données

L'exemple de code suivant montre comment créer un jeton d'authentification, puis comment l'utiliser pour se connecter à un cluster exécutant Aurora MySQL.

Pour exécuter cet exemple de code, vous avez besoin de AWS SDK pour Java, disponible sur le site AWS. En outre, vous avez besoin des éléments suivants :

- MySQL Connector/J. Cet exemple de code a été testé avec `mysql-connector-java-5.1.33-bin.jar`.
- Certificat intermédiaire pour Amazon Aurora qui est spécifique à une région AWS. (Pour plus d'informations, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.) À l'exécution, le chargeur de classe recherche le certificat dans le même annuaire que celui de cet exemple de code Java, afin de pouvoir le trouver.
- Modifiez la valeur des variables suivantes selon les besoins :
  - `RDS_INSTANCE_HOSTNAME` – Le nom d'hôte de l'cluster de bases de données auquel vous souhaitez accéder.
  - `RDS_INSTANCE_PORT` – Le numéro du port utilisé pour la connexion à votre cluster de bases de données PostgreSQL.
  - `REGION_NAME` – La région AWS où l'cluster de bases de données s'exécute.
  - `DB_USER` – Le compte de base de données auquel vous souhaitez accéder.
  - `SSL_CERTIFICATE` – Un sertificat SSL pour Amazon Aurora qui est spécifique à une région AWS.

    Pour télécharger un certificat pour votre région AWS, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters. Placez le certificat SSL dans le même annuaire que ce fichier de programme Java, afin que le chargeur de classe puisse le trouver à l'exécution.

Cet exemple de code obtient les informations d'identification AWS à partir de la chaîne de fournisseur d'informations d'identification par défaut.

> **Note**
>
> Spécifiez un mot de passe pour `DEFAULT_KEY_STORE_PASSWORD` différent de celui indiqué ici, en tant que bonne pratique de sécurité.

```
package com.amazonaws.samples;

import com.amazonaws.services.rds.auth.RdsIamAuthTokenGenerator;
import com.amazonaws.services.rds.auth.GetIamAuthTokenRequest;
```

```
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.auth.DefaultAWSCredentialsProviderChain;
import com.amazonaws.auth.AWSStaticCredentialsProvider;

import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.security.KeyStore;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

import java.net.URL;

public class IAMDatabaseAuthenticationTester {
    //&AWS; Credentials of the IAM user with policy enabling IAM Database Authenticated
 access to the db by the db user.
    private static final DefaultAWSCredentialsProviderChain creds = new
 DefaultAWSCredentialsProviderChain();
    private static final String AWS_ACCESS_KEY =
 creds.getCredentials().getAWSAccessKeyId();
    private static final String AWS_SECRET_KEY =
 creds.getCredentials().getAWSSecretKey();

    //Configuration parameters for the generation of the IAM Database Authentication
 token
    private static final String RDS_INSTANCE_HOSTNAME = "rdsmysql.123456789012.us-
west-2.rds.amazonaws.com";
    private static final int RDS_INSTANCE_PORT = 3306;
    private static final String REGION_NAME = "us-west-2";
    private static final String DB_USER = "jane_doe";
    private static final String JDBC_URL = "jdbc:mysql://" + RDS_INSTANCE_HOSTNAME +
 ":" + RDS_INSTANCE_PORT;

    private static final String SSL_CERTIFICATE = "rds-ca-2019-us-west-2.pem";

    private static final String KEY_STORE_TYPE = "JKS";
    private static final String KEY_STORE_PROVIDER = "SUN";
```

```
    private static final String KEY_STORE_FILE_PREFIX = "sys-connect-via-ssl-test-
cacerts";
    private static final String KEY_STORE_FILE_SUFFIX = ".jks";
    private static final String DEFAULT_KEY_STORE_PASSWORD = "changeit";

    public static void main(String[] args) throws Exception {
        //get the connection
        Connection connection = getDBConnectionUsingIam();

        //verify the connection is successful
        Statement stmt= connection.createStatement();
        ResultSet rs=stmt.executeQuery("SELECT 'Success!' FROM DUAL;");
        while (rs.next()) {
             String id = rs.getString(1);
            System.out.println(id); //Should print "Success!"
        }

        //close the connection
        stmt.close();
        connection.close();

        clearSslProperties();

    }

    /**
     * This method returns a connection to the db instance authenticated using IAM
 Database Authentication
     * @return
     * @throws Exception
     */
    private static Connection getDBConnectionUsingIam() throws Exception {
        setSslProperties();
        return DriverManager.getConnection(JDBC_URL, setMySqlConnectionProperties());
    }

    /**
     * This method sets the mysql connection properties which includes the IAM Database
 Authentication token
     * as the password. It also specifies that SSL verification is required.
     * @return
     */
    private static Properties setMySqlConnectionProperties() {
        Properties mysqlConnectionProperties = new Properties();
```

```
        mysqlConnectionProperties.setProperty("verifyServerCertificate","true");
        mysqlConnectionProperties.setProperty("useSSL", "true");
        mysqlConnectionProperties.setProperty("user",DB_USER);
        mysqlConnectionProperties.setProperty("password",generateAuthToken());
        return mysqlConnectionProperties;
    }

    /**
     * This method generates the IAM Auth Token.
     * An example IAM Auth Token would look like follows:
     * btusi123.cmz7kenwo2ye.rds.cn-north-1.amazonaws.com.cn:3306/?
Action=connect&DBUser=iamtestuser&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Date=20171003T010726Z&X-Amz-SignedHeaders=host&X-Amz-Expires=899&X-Amz-
Credential=AKIAPFXHGVDI5RNFO4AQ%2F20171003%2Fcn-north-1%2Frds-db%2Faws4_request&X-Amz-
Signature=f9f45ef96c1f770cdad11a53e33ffa4c3730bc03fdee820cfdf1322eed15483b
     * @return
     */
    private static String generateAuthToken() {
        BasicAWSCredentials awsCredentials = new BasicAWSCredentials(AWS_ACCESS_KEY,
 AWS_SECRET_KEY);

        RdsIamAuthTokenGenerator generator = RdsIamAuthTokenGenerator.builder()
                .credentials(new
 AWSStaticCredentialsProvider(awsCredentials)).region(REGION_NAME).build();
        return generator.getAuthToken(GetIamAuthTokenRequest.builder()

 .hostname(RDS_INSTANCE_HOSTNAME).port(RDS_INSTANCE_PORT).userName(DB_USER).build());
    }

    /**
     * This method sets the SSL properties which specify the key store file, its type
 and password:
     * @throws Exception
     */
    private static void setSslProperties() throws Exception {
        System.setProperty("javax.net.ssl.trustStore", createKeyStoreFile());
        System.setProperty("javax.net.ssl.trustStoreType", KEY_STORE_TYPE);
        System.setProperty("javax.net.ssl.trustStorePassword",
 DEFAULT_KEY_STORE_PASSWORD);
    }

    /**
     * This method returns the path of the Key Store File needed for the SSL
 verification during the IAM Database Authentication to
```

```
     * the db instance.
     * @return
     * @throws Exception
     */
    private static String createKeyStoreFile() throws Exception {
        return createKeyStoreFile(createCertificate()).getPath();
    }

    /**
     *  This method generates the SSL certificate
     * @return
     * @throws Exception
     */
    private static X509Certificate createCertificate() throws Exception {
        CertificateFactory certFactory = CertificateFactory.getInstance("X.509");
        URL url = new File(SSL_CERTIFICATE).toURI().toURL();
        if (url == null) {
            throw new Exception();
        }
        try (InputStream certInputStream = url.openStream()) {
            return (X509Certificate) certFactory.generateCertificate(certInputStream);
        }
    }

    /**
     * This method creates the Key Store File
     * @param rootX509Certificate - the SSL certificate to be stored in the KeyStore
     * @return
     * @throws Exception
     */
    private static File createKeyStoreFile(X509Certificate rootX509Certificate) throws
Exception {
        File keyStoreFile = File.createTempFile(KEY_STORE_FILE_PREFIX,
KEY_STORE_FILE_SUFFIX);
        try (FileOutputStream fos = new FileOutputStream(keyStoreFile.getPath())) {
            KeyStore ks = KeyStore.getInstance(KEY_STORE_TYPE, KEY_STORE_PROVIDER);
            ks.load(null);
            ks.setCertificateEntry("rootCaCertificate", rootX509Certificate);
            ks.store(fos, DEFAULT_KEY_STORE_PASSWORD.toCharArray());
        }
        return keyStoreFile;
    }

    /**
```

```
     * This method clears the SSL properties.
     * @throws Exception
     */
    private static void clearSslProperties() throws Exception {
           System.clearProperty("javax.net.ssl.trustStore");
           System.clearProperty("javax.net.ssl.trustStoreType");
           System.clearProperty("javax.net.ssl.trustStorePassword");
    }

}
```

Si vous souhaitez vous connecter à un cluster de bases de données via un proxy, consultez Connexion à une base de données à l'aide de l'authentification IAM.

Connexion à votre cluster de base de données à l'aide de l'authentification IAM et de AWS SDK pour Python (Boto3)

Vous pouvez vous connecter à un cluster de bases de données Aurora MySQL ou Aurora PostgreSQL avec l'AWS SDK pour Python (Boto3), comme décrit ci-après.

Prérequis

Les conditions préalables à la connexion à votre cluster de base de données à l'aide de l'authentification IAM sont les suivantes :

- Activation et désactivation de l’authentification de base de données IAM
- Création et utilisation d'une politique IAM pour l'accès à une base de données IAM
- Création d’un compte de base de données à l’aide de l’authentification IAM

En outre, assurez-vous que les bibliothèques importées dans l'exemple de code existent sur votre système.

Exemples

Les exemples de code utilisent des profils pour les informations d'identification partagées. Pour plus d'informations sur les informations d'identification spécifiant, veuillez consulter Informations d'identification dans la documentation AWS SDK pour Python (Boto3).

Les exemples de code suivants montre comment générer un jeton d'authentification, puis comment l'utiliser pour se connecter à un cluster de base de données.

Pour exécuter cet exemple de code, vous avez besoin de AWS SDK pour Python (Boto3), disponible sur le site AWS.

Modifiez la valeur des variables suivantes selon les besoins :

- `ENDPOINT` – Le point de terminaison de cluster de base de données à laquelle vous souhaitez accéder.
- `PORT` – Le numéro du port utilisé lors de la connexion au cluster d' de base de données.
- `USER` – Le compte de base de données auquel vous souhaitez accéder.
- `REGION` – La région AWS où l'cluster de base de données s'exécute.
- `DBNAME` – La base de données à laquelle vous souhaitez accéder.
- `SSLCERTIFICATE` – Le chemin d'accès complet au certificat SSL pour Amazon Aurora

  Pour `ssl_ca`, spécifiez un certificat SSL. Pour télécharger un certificat SSL, consultez Utilisation SSL/TLS pour chiffrer une connexion à une de clusters.

> **Note**
>
> Vous ne pouvez pas utiliser un enregistrement DNS Route 53 personnalisé ni un point de terminaison personnalisé Aurora à la place du point de terminaison du cluster de bases de données pour générer le jeton d'authentification.

Ce code se connecte à un cluster de base de données Aurora MySQL.

Avant d'exécuter ce code, installez le pilote PyMySQL en suivant les instructions fournies dans Python Package Index.

```
import pymysql
import sys
import boto3
import os

ENDPOINT="mysqlcluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
PORT="3306"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"
```

```
os.environ['LIBMYSQL_ENABLE_CLEARTEXT_PLUGIN'] = '1'

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='default')
client = session.client('rds')

token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn =
 pymysql.connect(auth_plugin_map={'mysql_clear_password':None},host=ENDPOINT,
 user=USER, password=token, port=PORT, database=DBNAME, ssl_ca='SSLCERTIFICATE',
 ssl_verify_identity=True, ssl_verify_cert=True)
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

Ce code se connecte à un cluster de base de données Aurora PostgreSQL.

Avant d'exécuter ce code, installez `psycopg2` en suivant les instructions de la documentation de Psycopg.

```
import psycopg2
import sys
import boto3
import os

ENDPOINT="postgresmycluster.cluster-123456789012.us-east-1.rds.amazonaws.com"
PORT="5432"
USER="jane_doe"
REGION="us-east-1"
DBNAME="mydb"

#gets the credentials from .aws/credentials
session = boto3.Session(profile_name='RDSCreds')
client = session.client('rds')
```

```
token = client.generate_db_auth_token(DBHostname=ENDPOINT, Port=PORT, DBUsername=USER,
 Region=REGION)

try:
    conn = psycopg2.connect(host=ENDPOINT, port=PORT, database=DBNAME, user=USER,
 password=token, sslrootcert="SSLCERTIFICATE")
    cur = conn.cursor()
    cur.execute("""SELECT now()""")
    query_results = cur.fetchall()
    print(query_results)
except Exception as e:
    print("Database connection failed due to {}".format(e))
```

Si vous souhaitez vous connecter à un cluster de bases de données via un proxy, consultez Connexion à une base de données à l'aide de l'authentification IAM.

Résolution des problèmes liés à l'authentification de base de données IAM

Vous trouverez ci-dessous des idées de dépannage pour certains problèmes courants liés à l'authentification de base de données IAM, ainsi que des informations sur les journaux CloudWatch pour l'authentification de base de données IAM.

Exportation des journaux d'erreurs d'authentification de base de données IAM vers CloudWatch Logs

Les journaux des erreurs d'authentification de base de données IAM sont stockés sur l'hôte de la base de données, et vous pouvez les exporter vers votre compte CloudWatch Logs. Utilisez les journaux et les méthodes de correction de cette page pour résoudre les problèmes liés à l'authentification de base de données IAM.

Vous pouvez activer les exportations de journaux vers CloudWatch Logs à partir de la console AWS CLI et de l'API RDS. Pour des instructions sur l'utilisation de la console, consultez Publication des journaux de base de données dans Amazon CloudWatch Logs.

Pour exporter vos journaux d'erreurs d'authentification de base de données IAM vers CloudWatch Logs lors de la création d'un cluster de bases de données à partir du AWS CLI, utilisez la commande suivante :

```
aws rds create-db-cluster --db-cluster-identifier mydbinstance \
--region us-east-1 \
--engine postgres \
```

```
--engine-version 16 \
--master-username master \
--master-user-password password \
--publicly-accessible \
--enable-iam-database-authentication \
--enable-cloudwatch-logs-exports=iam-db-auth-error
```

Pour exporter vos journaux d'erreurs d'authentification de base de données IAM vers CloudWatch Logs lors de la modification d'un cluster de bases de données à partir du AWS CLI, utilisez la commande suivante :

```
aws rds modify-db-cluster --db-cluster-identifier mydbcluster \
--region us-east-1 \
--cloudwatch-logs-export-configuration '{"EnableLogTypes":["iam-db-auth-error"]}'
```

Pour vérifier si votre cluster de bases de données exporte les journaux d'authentification de base de données IAM vers CloudWatch Logs, vérifiez si le paramètre `EnabledCloudwatchLogsExports` est défini sur `iam-db-auth-error` dans la sortie de la commande `describe-db-instances`.

```
aws rds describe-db-cluster --region us-east-1 --db-cluster-identifier mydbcluster
            ...

             "EnabledCloudwatchLogsExports": [
                "iam-db-auth-error"
            ],
            ...
```

Métriques CloudWatch pour l'authentification de base de données IAM

Amazon Aurora fournit des statistiques en temps quasi réel concernant l'authentification de base de données IAM sur votre compte Amazon CloudWatch. Le tableau suivant répertorie les métriques d'authentification de base de données IAM disponibles avec CloudWatch :

Métrique	Description
`IamDbAuthConnectionRequests`	Nombre total de demandes de connexion effectuées avec l'authentification de base de données IAM.

Métrique	Description
`IamDbAuthConnectionSuccess`	Nombre total de demandes d'authentification de base de données IAM réussies.
`IamDbAuthConnectionFailure`	Nombre total de demandes d'authentification de base de données IAM ayant échoué.
`IamDbAuthConnectionFailureInvalidToken`	Nombre total de demandes d'authentification de base de données IAM ayant échoué à cause d'un jeton non valide.
`IamDbAuthConnectionFailureInsufficientPermissions`	Nombre total de demandes d'authentification de base de données IAM ayant échoué en raison de politiques ou d'autorisations incorrectes.
`IamDbAuthConnectionFailureThrottling`	Nombre total de demandes d'authentification de base de données IAM ayant échoué en raison de la limitation de l'authentification de base de données IAM.
`IamDbAuthConnectionFailureServerError`	Nombre total de demandes d'authentification de base de données IAM ayant échoué en raison d'une erreur interne du serveur dans la fonctionnalité d'authentification de base de données IAM.

Problèmes courants et solutions correspondantes

Vous pouvez rencontrer les problèmes suivants lors de l'utilisation de l'authentification de base de données IAM. Suivez les étapes de correction indiquées dans le tableau pour résoudre les problèmes :

Erreur	Métrique(s)	Cause	Solution
`[ERROR] Failed to authenticate the connection request for user db_user because the provided token is malformed or otherwise invalid. (Status Code: 400, Error Code: InvalidToken)`	`IamDbAuthConnectionFailure` `IamDbAuthConnectionFailureInvalidToken`	Le jeton d'authentification de base de données IAM figurant dans la demande de connexion n'est pas un jeton SigV4a valide ou il n'est pas formaté correctement.	Vérifiez votre stratégie de génération de jetons dans votre application. Dans certains cas, assurez-vous de transmettre le jeton avec un formatage valide. Le fait de tronquer le jeton (ou un formatage de chaîne incorrect) le rend non valide.
`[ERROR] Failed to authenticate the connection request for user db_user because the token age is longer than 15 minutes. (Status Code: 400, Error Code:ExpiredToken)`	`IamDbAuthConnectionFailure` `IamDbAuthConnectionFailureInvalidToken`	Le jeton d'authentification de base de données IAM a expiré. Les jetons ne sont valides que pendant 15 minutes.	Vérifiez votre logique de mise en cache et/ou de réutilisation des jetons dans votre application. Vous ne devez pas réutiliser des jetons datant de plus de 15 minutes.
`[ERROR] Failed to authorize the connection request for user db_user because the IAM policy assumed by the caller 'arn:aws:sts::1234`	`IamDbAuthConnectionFailure` `IamDbAuthConnectionFailureInsufficientPermissions`	Cette erreur peut être due aux raisons suivantes : • La politique IAM adoptée par l'application n'autorise pas	Vérifiez que vous assumez le rôle et/ou la politique IAM dans votre application. Assurez-vous de suivre la même politique pour générer le jeton que pour vous

Erreur	Métrique(s)	Cause	Solution
`56789012: assumed-role/ <RoleName>/ <RoleSession>' is not authorized to perform `rds-db:connect` on the DB instance. (Status Code: 403, Error Code:NotAuthorized)`		l'action `rds-db:connect` . • Vous assumez le rôle ou la politique incorrects pour que *db_user* se connecte à la base de données. • Vous adoptez la bonne politique pour *db_user*, mais vous ne vous connectez pas à la bonne base de données.	connecter à la base de données.
`[ERROR] Failed to authorize the connection request for user` *db_user* `due to IAM DB authentication throttling. (Status Code: 429, Error Code: ThrottlingException)`	`IamDbAuthConnectionFailure` `IamDbAuthConnectionFailureThrottling`	Vous envoyez trop de demandes de connexion à votre base de données en peu de temps. La limite de limitation de l'authentification IAM DB est de 200 connexions par seconde.	Réduisez le taux d'établissement de nouvelles connexions grâce à l'authentification IAM. Envisagez d'implémenter le regroupement de connexions à l'aide du proxy RDS afin de réutiliser les connexions établies dans votre application.

Erreur	Métrique(s)	Cause	Solution
`[ERROR] Failed to authorize the connection request for user db_user due to an internal IAM DB authentication error. (Status Code: 500, Error Code: InternalError)`	`IamDbAuthConnectionFailure` `IamDbAuthConnectionFailureThrottling`	Une erreur interne s'est produite lors de l'autorisation de la connexion à la base de données avec l'authentification de base de données IAM.	Contactez https://aws.amazon.com/premiumsupport/ pour enquêter sur le problème.

Résolution des problèmes liés à Identity and Access Amazon Aurora

Utilisez les informations suivantes pour identifier et résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec Aurora et IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans Aurora
- Je ne suis pas autorisé à exécuter iam:PassRole
- Je souhaite autoriser des personnes extérieures à mon compte AWS, à accéder à mes ressources Aurora.

Je ne suis pas autorisé à effectuer une action dans Aurora

Si la AWS Management Console indique que vous n'êtes pas autorisé à exécuter une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de connexion.

L'exemple d'erreur suivant se produit lorsque l'utilisateur `mateojackson` tente d'utiliser la console pour afficher des informations détaillées concernant un *`widget`*, mais ne dispose pas d'autorisations `rds:GetWidget`.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 rds:GetWidget on resource: my-example-widget
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *my-example-widget* à l'aide de l'action `rds:GetWidget`.

Je ne suis pas autorisé à exécuter iam:PassRole

Si vous recevez un message d'erreur selon lequel vous n'êtes pas autorisé à exécuter l'action `iam:PassRole`, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni vos informations de connexion. Demandez à cette personne de mettre à jour vos stratégies pour vous permettre de transmettre un rôle à Aurora.

Certains services AWS vous permettent de transmettre un rôle existant à ce service, au lieu de créer un nouveau rôle de service ou rôle lié à un service. Pour ce faire, un utilisateur doit disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur nommé `marymajor` essaie d'utiliser la console pour exécuter une action dans Aurora. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction du service. Mary ne dispose pas des autorisations nécessaires pour transférer le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dans ce cas, Mary demande à son administrateur de mettre à jour ses politiques pour lui permettre d'exécuter l'action `iam:PassRole`.

Je souhaite autoriser des personnes extérieures à mon compte AWS, à accéder à mes ressources Aurora.

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour en savoir plus, consultez les éléments suivants :

- Pour savoir si Aurora prend en charge ces fonctionnalités, consultez Comment Amazon Aurora fonctionne avec IAM.
- Pour savoir comment octroyer l'accès à vos ressources à des comptes AWS dont vous êtes propriétaire, veuillez consulter la section Fournir l'accès à un utilisateur IAM dans un autre compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment octroyer accès à vos ressources à des comptes AWS tiers, veuillez consulter Fournir l'accès aux comptes AWS appartenant à des tiers dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir un accès par le biais de la fédération d'identité, veuillez consulter Fournir un accès à des utilisateurs authentifiés en externe (fédération d'identité) dans le Guide de l'utilisateur IAM.
- Pour découvrir quelle est la différence entre l'utilisation des rôles et l'utilisation des politiques basées sur les ressources pour l'accès entre comptes, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.

Journalisation et surveillance dans Amazon Aurora

La surveillance est un aspect important du maintien de la fiabilité, de la disponibilité et des performances d' Amazon Aurora et de vos solutions AWS. Vous devez recueillir les données de surveillance de tous les composants de votre solution AWS, de manière à pouvoir déboguer plus facilement un éventuel échec multipoint. AWS fournit plusieurs outils pour surveiller vos ressources Amazon Aurora et réagir à des incidents potentiels :

Alarmes Amazon CloudWatch

À l'aide d'alarmes Amazon CloudWatch, vous surveillez une métrique unique sur une période donnée que vous spécifiez. Si la métrique dépasse un seuil donné, une notification est envoyée à une rubrique Amazon SNS ou à une stratégie AWS Auto Scaling. Les alarmes CloudWatch n'invoquent pas une action uniquement parce qu'elles se trouvent dans un état particulier. L'état doit avoir changé et avoir été conservé pendant un nombre de périodes spécifié.

AWS CloudTrailJournaux

CloudTrail fournit un registre des actions entreprises par un utilisateur, un rôle ou un service AWS dans Amazon Aurora. CloudTrail capture tous les appels d'API pour Amazon Aurora en tant qu'événements, y compris les appels de la console et les appels de code à des opérations d'API Amazon RDS. Les informations collectées par CloudTrail, vous permettent de déterminer quelle demande a été envoyée à Amazon Aurora, l'adresse IP source à partir de laquelle la demande

a été effectuée, qui a effectué la demande, quand, ainsi que d'autres informations. Pour plus d'informations, consultez Surveillance des appels d'API Amazon Aurora dansAWS CloudTrail.

Surveillance améliorée

Amazon Aurora fournit des métriques en temps réel pour le système d'exploitation sur lequel votre cluster de bases de données s'exécute. Vous pouvez afficher les métriques pour votre cluster de bases de données à l'aide de la console, ou utiliser la sortie JSON de surveillance améliorée d'Amazon CloudWatch Logs dans le système de surveillance de votre choix. Pour plus d'informations, consultez Surveillance des métriques du système d'exploitation à l'aide de la Surveillance améliorée.

Analyse des performances d'Amazon RDS

Performance Insights complète les fonctions de surveillance existantes sur Amazon Aurora. Ce service illustre les performances de votre base de données et facilite votre analyse des problèmes qui les impactent. Grâce au tableau de bord de Performance Insights, vous pouvez visualiser la charge de la base de données et la filtrer par attentes, instructions SQL, hôtes ou utilisateurs. Pour plus d'informations, consultez Surveillance de la charge de la base de données avec Performance Insights sur .

Journaux de base de données

Vous pouvez afficher, télécharger et consulter les journaux de base de données à l'aide d'AWS Management Console, de l'AWS CLI ou de l'API RDS. Pour plus d'informations, consultez Surveillance des fichiers journaux Amazon Aurora.

Recommandations Amazon Aurora

Amazon Aurora fournit des recommandations automatisées pour les ressources de base de données. Ces recommandations offrent des conseils quand aux bonnes pratiques en analysant la configuration du cluster de bases de données, son utilisation et les données relatives à ses performances. Pour plus d'informations, consultez Recommandations d'Amazon Aurora.

Notification d'événement Amazon Aurora

Amazon Aurora utilise Amazon Simple Notification Service (Amazon SNS) pour fournir une notification lorsqu'un événement Amazon Aurora se produit. Ces notifications peuvent être faites sous n'importe quelle forme prise en charge par Amazon SNS pour une région AWS, telle qu'un e-mail, un SMS ou un appel à un point de terminaison HTTP. Pour plus d'informations, consultez Utiliser la notification d'événements d'Amazon RDS.

AWS Trusted Advisor

Trusted Advisor tire profit des bonnes pratiques acquises à travers la satisfaction de centaines de milliers de clients AWS. Trusted Advisor examine votre environnement AWS, puis effectue des recommandations lorsqu'il est possible de faire des économies, d'améliorer la disponibilité et les performances du système, ou de remédier à des failles de sécurité. Tous les clients AWS ont accès à cinq contrôles Trusted Advisor. Les clients avec un plan de support Business ou Enterprise peuvent afficher tous les contrôles Trusted Advisor.

Trusted Advisor dispose des contrôles de sécurité suivants liés à Amazon Aurora :

- Instances de base de données Amazon Aurora inactives
- Risque lié à l'accès aux groupes de sécurité Amazon Aurora
- Sauvegardes Amazon Aurora
- Multi-AZ Amazon Aurora
- Aurora Accessibilité d'instance de base de données

Pour plus d'informations sur ces vérifications, consultez Bonnes pratiques Trusted Advisor (Checks).

Flux d'activité de base de données.

Pour les flux d'activité de base de données protègent vos bases de données contre les menaces internes en contrôlant l'accès des DBA aux flux d'activité de base de données. Par conséquent, la collecte, la transmission, le stockage et les traitements des flux d'activité de base de données qui en découlent sont inaccessibles pour les DBA qui gèrent la base de données. Les flux d'activité de base de données peuvent vous permettre de fournir des protections à votre base de données et de satisfaire les exigences en matière de conformité et de règlementation. Pour plus d'informations, consultez Surveillance d'Amazon Aurora à l'aide des flux d'activité de base de données.

Pour plus d'informations concernant la surveillance Aurora, consultez Surveillance des métriques d'un cluster de bases de données Amazon Aurora .

Validation de la conformité pour Amazon Aurora

Des auditeurs tiers évaluent la sécurité et la conformité d'Amazon Aurora dans le cadre de plusieurs programmes de conformité AWS. Il s'agit notamment des certifications SOC, PCI, FedRAMP, HIPAA et d'autres.

Pour une liste des AWS services concernés par des programmes de conformité spécifiques, voir AWSServices concernés par programme de conformité. Pour obtenir des informations générales, veuillez consulter Programmes de conformité d'AWS.

Vous pouvez télécharger des rapports d'audit tiers à l'aide deAWS Artifact. Pour plus d'informations, consultez la section Téléchargement de rapports dans AWS Artifact.

Lorsque vous utilisez Aurora, votre responsabilité en matière de conformité est déterminée par la sensibilité de vos données, les objectifs de conformité de votre organisation et les lois et réglementations applicables. AWSfournit les ressources suivantes pour faciliter la mise en conformité :

- Guides de démarrage rapide sur la sécurité et la conformité : ces guides de déploiement abordent les considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de base axés sur la sécurité et la conformité sur. AWS
- Architecture axée sur la sécurité et la conformité HIPAA sur Amazon Web Services : ce livre blanc décrit comment les entreprises peuvent créer des applications AWS conformes à la loi HIPAA.
- AWSressources de conformité : collection de classeurs et de guides susceptibles de s'appliquer à votre secteur d'activité et à votre région.
- AWS Config— Ce AWS service évalue dans quelle mesure les configurations de vos ressources sont conformes aux pratiques internes, aux directives du secteur et aux réglementations.
- AWS Security Hub CSPM— Cela Service AWS fournit une vue complète de votre état de sécurité interneAWS. Security Hub CSPM utilise des contrôles de sécurité pour évaluer vos AWS ressources et vérifier votre conformité aux normes et aux meilleures pratiques du secteur de la sécurité. Pour obtenir la liste des services et des contrôles pris en charge, consultez la référence des contrôles Security Hub CSPM.

Résilience dans Amazon Aurora

L'infrastructure mondiale d'AWS s'articule autours de régions et de zones de disponibilité AWS.AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, reliées par un réseau à latence faible, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur les régions et les zones de disponibilité AWS, veuillez consulter Infrastructure mondiale AWS.

Outre l'infrastructure globale d'AWS, Aurora offrent différentes fonctions qui contribuent à satisfaire vos besoins en matière de résilience et de sauvegarde de données.

Sauvegarde et restauration

Aurora sauvegarde automatiquement votre volume de cluster et conserve les données de restauration pendant la totalité de la période de rétention des sauvegardes. Les sauvegardes Aurora étant continues et incrémentielles, vous pouvez rapidement opérer une restauration à un point quelconque de la période de rétention des sauvegardes. Aucun impact sur les performances ou interruption du service de base de données ne se produit lors de l'écriture des données de sauvegarde. Vous pouvez spécifier une période de rétention des sauvegardes, comprise entre 1 et 35 jours, lorsque vous créez ou modifiez un cluster de base de données.

Si vous souhaitez conserver une sauvegarde au-delà de la période de rétention, vous pouvez aussi prendre un instantané des données dans votre volume de cluster. Aurora conserve les données des restaurations incrémentielles pendant la totalité de la période de rétention des sauvegardes. Par conséquent, vous avez uniquement besoin de créer un instantané pour les données que vous souhaitez garder au-delà de la période de rétention des sauvegardes. Vous pouvez créer un nouveau cluster de base de données à partir de l'instantané.

Vous pouvez récupérer vos données en créant un cluster de bases de données Aurora à partir des données de sauvegarde qu'Aurora conserve ou d'un instantané de cluster de bases de données que vous avez enregistré. Vous pouvez créer rapidement une nouvelle copie d'un cluster de bases de données à partir des données de sauvegarde à un point quelconque de la période de rétention des sauvegardes. La nature continue et incrémentielle des sauvegardes Aurora pendant la période

de rétention signifie que vous n'avez pas besoin de prendre fréquemment des instantanés de vos données pour pouvoir améliorer les temps de restauration.

Pour plus d'informations, consultez Sauvegarde et restauration d'un cluster de bases de données Amazon Aurora.

Réplication

Les réplicas Aurora sont les points de terminaison indépendants d'un cluster de bases de données Aurora, utilisés de préférence pour le dimensionnement des opérations de lecture et l'augmentation de la disponibilité. Le nombre de réplicas Aurora pouvant être distribués entre les zones de disponibilité que couvre un cluster de bases de données au sein d'une région AWS est limité à 15. Le volume de cluster de bases de données est composé de plusieurs copies des données du cluster de bases de données. Cependant, les données du volume de cluster sont représentées comme un seul volume logique à l'instance principale en écriture et aux réplicas Aurora du cluster de bases de données. En d'autres termes, si l'instance principale est défaillante, un réplica Aurora peut être promu comme l'instance de base de données principale.

Aurora prend aussi en charge les options de réplication qui sont spécifiques à Aurora MySQL et Aurora PostgreSQL.

Pour plus d'informations, consultez Réplication avec Amazon Aurora.

Basculement

Aurora stocke les copies des données d'un cluster de base de données dans plusieurs zones de disponibilité d'une même région AWS. Le stockage se produit indépendamment du fait que les instances de base de données du cluster de base de données recouvrent ou pas plusieurs zones de disponibilité. Lorsque vous créez des réplicas Aurora sur plusieurs zones de disponibilité, Aurora les alloue et les gère automatiquement de manière synchrone. L'instance de base de données principale est répliquée de manière synchrone entre les zones de disponibilité sur des réplicas Aurora de façon à assurer une redondance des données, à éliminer les blocages d'I/O et à réduire les pics de latence pendant les sauvegardes du système. L'exécution d'un cluster de base de données avec la haute disponibilité peut améliorer la disponibilité pendant la maintenance planifiée du système et contribuer à protéger vos bases de données contre toute défaillance ou perturbation d'une zone de disponibilité.

Pour de plus amples informations, consultez Haute disponibilité pour Amazon Aurora.

Sécurité de l'infrastructure dans Amazon Aurora

En tant que service géré, Amazon Relational Database Service est protégé AWS par la sécurité du réseau mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section Sécurité du AWS cloud. Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section Protection de l'infrastructure dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Amazon RDS via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Ses suites de chiffrement PFS (Perfect Forward Secrecy) comme DHE (Ephemeral Diffie-Hellman) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, Aurora offre des fonctions pour contribuer à prendre en charge la sécurité de l'infrastructure.

Groupes de sécurité

Les groupes de sécurité contrôlent l'accès dont dispose le trafic entrant et sortant d'un cluster de base de données. Par défaut, l'accès au réseau est désactivé sur un cluster de base de données. Vous pouvez spécifier des règles dans un groupe de sécurité qui autorisent l’accès depuis une plage d’adresses IP, un port ou un groupe de sécurité. Une fois les règles de trafic entrant configurées, les mêmes règles s'appliquent à tou(te)s les clusters de base de données qui sont associé(e)s à ce groupe de sécurité.

Pour de plus amples informations, veuillez consulter Contrôle d’accès par groupe de sécurité.

Accessible publiquement

Lorsque vous lancez une instance de base de données à l'intérieur d'un VPC basé sur le service Amazon VPC, vous pouvez activer ou désactiver l'accessibilité publique pour cette instance de base de données. Pour définir si l'instance de base de données que vous créez comporte un nom DNS qui se résout en une adresse IP publique, vous utilisez le paramètre Public accessibility (Accessibilité publique). Ce paramètre vous permet de définir s'il existe un accès publique à l'instance de base

de données. Vous pouvez modifier une instance de base de données pour activer ou désactiver l'accessibilité publique en modifiant le paramètre Public accessibility (Accessibilité publique).

Pour de plus amples informations, veuillez consulter Masquer un cluster de bases de données dans un VPC depuis Internet.

> **Note**
>
> Si votre instance de base de données se trouve dans un VPC mais n'est pas accessible au public, vous pouvez également utiliser une connexion AWS Site-to-Site VPN ou une Direct Connect connexion pour y accéder depuis un réseau privé. Pour de plus amples informations, veuillez consulter Confidentialité du trafic inter-réseau.

API Amazon RDS et points de terminaison d'un VPC d'interface (AWS PrivateLink)

Vous pouvez établir une connexion privée entre votre VPC et vos points de terminaison d'API Amazon RDS en créant un point de terminaison de VPC d'interface. Les points de terminaison d'interface sont alimentés par AWS PrivateLink.

AWS PrivateLink vous permet d'accéder en privé aux opérations de l'API Amazon RDS sans passerelle Internet, appareil NAT, connexion VPN ou Direct Connect connexion. Les instances de base de données de votre VPC n'ont pas besoin d'adresses IP publiques pour communiquer avec les points de terminaison d'API Amazon RDS for lancer, modifier ou mettre fin à des instances et des clusters de base de données. Vos instances de base de données n'ont pas non plus besoin d'adresses IP publiques pour utiliser une des opérations d'API RDS disponibles. Le trafic entre votre VPC et Amazon RDS ne quitte pas le réseau Amazon.

Chaque point de terminaison d'interface est représenté par une ou plusieurs interfaces réseau Elastic dans vos sous-réseaux. Pour plus d'informations sur les interfaces réseau élastiques, consultez la section relative aux interfaces réseau élastiques dans le guide de EC2 l'utilisateur Amazon.

Pour plus d'informations sur les points de terminaison d'un VPC, consultez Points de terminaison de VPC d'interface (AWS PrivateLink) dans le Guide de l'utilisateur Amazon VPC. Pour plus d'informations sur les opérations d'API RDS, consultez Référence d'API Amazon RDS.

Vous n'avez pas besoin d'un point de terminaison VPC d'interface pour vous connecter à un(e) cluster de base de données. Pour de plus amples informations, veuillez consulter Scénarios d’accès à un cluster de bases de données d’un VPC.

Considérations relatives aux points de terminaison d'un VPC

Avant de configurer un point de terminaison d'un VPC d'interface pour les points de terminaison d'API Amazon RDS, assurez-vous de vérifier les propriétés et limitations du point de terminaison d'interface dans le Amazon VPC Guide de l'utilisateur.

Toutes les opérations d'API RDS pertinentes pour gestion de ressources Amazon Aurora sont disponibles à partir de votre VPC à l'aide d' AWS PrivateLink.

Les politiques de point de terminaison d'un VPC sont prises en charge pour les points de terminaison de l'API RDS. Par défaut, l'accès complet aux opérations de l'API RDS est autorisé via le point de

terminaison. Pour plus d'informations, veuillez consulter Contrôle de l'accès aux services avec points de terminaison d'un VPC dans le Amazon VPC Guide de l'utilisateur.

Disponibilité

L'API Amazon RDS prend actuellement en charge les points de terminaison VPC dans les régions suivantes : AWS

- USA Est (Ohio)
- USA Est (Virginie du Nord)
- USA Ouest (Californie du Nord)
- USA Ouest (Oregon)
- Afrique (Le Cap)
- Asie-Pacifique (Hong Kong)
- Asie-Pacifique (Mumbai)
- Asie-Pacifique (Nouvelle-Zélande)
- Asie-Pacifique (Osaka)
- Asia Pacific (Seoul)
- Asie-Pacifique (Singapour)
- Asie-Pacifique (Sydney)
- Asie-Pacifique (Taipei)
- Asie-Pacifique (Thaïlande)
- Asie-Pacifique (Tokyo)
- Canada (Centre)
- Canada Ouest (Calgary)
- Chine (Pékin)
- China (Ningxia)
- Europe (Francfort)
- Europe (Zurich)
- Europe (Irlande)
- Europe (Londres)
- Europe (Paris)

- Europe (Stockholm)
- Europe (Milan)
- Israël (Tel Aviv)
- Mexique (Centre)
- Middle East (Bahrain)
- Amérique du Sud (São Paulo)
- AWS GovCloud (USA Est)
- AWS GovCloud (US-Ouest)

Création d'un point de terminaison de VPC d'interface pour l'API Amazon RDS

Vous pouvez créer un point de terminaison VPC pour l'API Amazon RDS à l'aide de la console Amazon VPC ou du (). AWS Command Line Interface AWS CLI Pour plus d'informations, consultez Création d'un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Créez un point de terminaison de VPC pour l'API Amazon RDS à l'aide du nom de service `com.amazonaws.region.rds`.

À l'exception des AWS régions de Chine, si vous activez le DNS privé pour le point de terminaison, vous pouvez envoyer des demandes d'API à Amazon RDS avec le point de terminaison VPC en utilisant son nom DNS par défaut pour AWS la région, par exemple. `rds.us-east-1.amazonaws.com` Pour les AWS régions de Chine (Pékin) et de Chine (Ningxia), vous pouvez effectuer des demandes d'API avec le point de terminaison VPC `rds-api.cn-north-1.amazonaws.com.cn` en utilisant `rds-api.cn-northwest-1.amazonaws.com.cn` et, respectivement.

Pour plus d'informations, consultez Accès à un service via un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Création d'une politique de point de terminaison de VPC pour l'API Amazon RDS

Vous pouvez attacher une politique de point de terminaison à votre point de terminaison de VPC qui contrôle l'accès à l'API Amazon RDS. La politique spécifie les informations suivantes :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, veuillez consulter Contrôle de l'accès aux services avec points de terminaison d'un VPC dans le Amazon VPC Guide de l'utilisateur.

Exemple : politique de point de terminaison de VPC pour les actions de l'API Amazon RDS

Voici un exemple de politique de point de terminaison pour l'API Amazon RDS. Lorsqu'elle est attachée à un point de terminaison, cette politique accorde l'accès aux actions de l'API Amazon RDS répertoriées pour tous les principaux sur toutes les ressources.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "rds:CreateDBInstance",
            "rds:ModifyDBInstance",
            "rds:CreateDBSnapshot"
         ],
         "Resource":"*"
      }
   ]
}
```

Exemple : politique de point de terminaison VPC qui refuse tout accès depuis un compte spécifié AWS

La politique de point de terminaison VPC suivante refuse au AWS compte `123456789012` tout accès aux ressources utilisant le point de terminaison. La politique autorise toutes les actions provenant d'autres comptes.

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
```

```
      "Resource": "*",
      "Principal": "*"
    },
    {
      "Action": "*",
      "Effect": "Deny",
      "Resource": "*",
      "Principal": { "AWS": [ "123456789012" ] }
     }
   ]
}
```

Bonnes pratiques de sécurité pour Amazon Aurora

Utilisez des comptes Gestion des identités et des accès AWS (IAM) pour contrôler l'accès aux opérations de l'API Amazon RDS, en particulier aux opérations qui créent, modifient ou suppriment des ressources . Les ressources de ce type incluent les clusters de bases de données, les groupes de sécurité et les groupes de paramètres. Utilisez également IAM pour contrôler les actions qui effectuent des tâches administratives courantes telles que la sauvegarde et la restauration de clusters de bases de données.

- Créez un utilisateur pour chaque personne qui gère les ressources Amazon Aurora, y compris vous-même. N'utilisez pas les informations d'identification AWS root pour gérer les ressources Amazon Aurora.
- Accordez à chaque utilisateur un ensemble minimum d’autorisations requises pour exécuter ses tâches.
- Utilisez des groupes IAM pour gérer efficacement des autorisations pour plusieurs utilisateurs.
- Effectuer une rotation régulière des informations d’identification IAM.
- Configurez AWS Secrets Manager pour alterner automatiquement les secrets pour Amazon Aurora. Pour plus d’informations, consultez Rotation de vos secrets AWS Secrets Manager dans le Guide de l’utilisateur AWS Secrets Manager. Vous pouvez également récupérer les informations d'identification par AWS Secrets Manager programmation. Pour plus d’informations, consultez Récupération de la valeur du secret dans le Guide de l’utilisateur AWS Secrets Manager.

Pour plus d’informations sur la sécurité dans Amazon Aurora, consultez Sécurité dans Amazon Aurora. Pour plus d’informations sur IAM, consultez Gestion des identités et des accès AWS. Pour plus d’informations sur les bonnes pratiques IAM, consultez Bonnes pratiques IAM.

AWS Security Hub CSPMutilise des contrôles de sécurité pour évaluer les configurations des ressources et les normes de sécurité afin de vous aider à vous conformer aux différents cadres de conformité. Pour plus d'informations sur l'utilisation de Security Hub CSPM pour évaluer les ressources RDS, consultez les contrôles Amazon Relational Database Service dans le guide de l'utilisateur. AWS Security Hub

Vous pouvez surveiller votre utilisation de RDS en ce qui concerne les meilleures pratiques de sécurité à l'aide de Security Hub CSPM. Pour plus d'informations, voir Qu'est-ce que c'est AWS Security Hub CSPM ? .

Utilisez l'API AWS Management ConsoleAWS CLI, la ou l'API RDS pour modifier le mot de passe de votre utilisateur principal. Si vous utilisez un autre outil, comme SQL client, pour modifier le mot de passe de l’utilisateur principal, cela pourrait finir par la révocation involontaire des privilèges de l’utilisateur.

Amazon GuardDuty est un service de surveillance continue de la sécurité qui analyse et traite diverses sources de données, y compris l'activité de connexion à Amazon RDS. Il utilise les flux de renseignements sur les menaces et le machine learning pour identifier les activités inattendues, potentiellement non autorisées, de comportement de connexion suspect et malveillantes dans votre environnement AWS.

Lorsqu'Amazon GuardDuty RDS Protection détecte une tentative de connexion potentiellement suspecte ou anormale indiquant une menace pour votre base de données, GuardDuty génère un nouveau résultat contenant des informations sur la base de données potentiellement compromise. Pour plus d’informations, consultez Surveillance des menaces avec Amazon GuardDuty RDS Protection pour Amazon Aurora.

Contrôle d’accès par groupe de sécurité

Les groupes de sécurité du VPC contrôlent l’accès dont dispose le trafic entrant et sortant d’ un cluster.de bases de données. Par défaut, l’accès au réseau est désactivé pour un cluster de bases de données. Vous pouvez spécifier des règles dans un groupe de sécurité qui autorisent l’accès depuis une plage d’adresses IP, un port ou un groupe de sécurité. Une fois les règles de trafic entrant configurées, les mêmes règles s’appliquent à tous les clusters de bases de données qui sont associés à ce groupe de sécurité. Vous pouvez spécifier jusqu’à 20 règles dans un groupe de sécurité.

Présentation des groupes de sécurité VPC

Chaque règle de groupe de sécurité VPC permet à une source spécifique d'accéder à un cluster de bases de données dans un VPC associé à ce groupe de sécurité VPC. Cette source peut être une plage d'adresses (par exemple, 203.0.113.0/24) ou un autre groupe de sécurité VPC. En spécifiant un groupe de sécurité VPC en tant que source, vous autorisez le trafic entrant provenant de toutes les instances (généralement les serveurs d'application) qui utilisent le groupe de sécurité VPC source. Les groupes de sécurité du VPC peuvent avoir des règles qui régissent à la fois le trafic entrant et sortant. Cependant, les règles de trafic sortant ne s'appliquent généralement pas aux clusters de bases de données. Les règles de trafic sortant ne s'appliquent que si le cluster de bases de données fait office de client. Vous devez utiliser l'API Amazon EC2 ou l'option Security Group (Groupe de sécurité) de la console VPC pour créer des groupes de sécurité VPC.

Lorsque vous créez des règles pour votre groupe de sécurité VPC pour permettre d'accéder aux clusters dans votre VPC, vous devez spécifier un port pour chaque plage d'adresses à laquelle la règle autorise l'accès. Par exemple, si vous souhaitez activer l'accès Secure Shell (SSH) pour les instances du VPC, créez une règle autorisant l'accès au port TCP 22 pour la plage d'adresses spécifiée.

Vous pouvez configurer plusieurs groupes de sécurité VPC qui permettent d'accéder à des ports différents pour différentes instances dans votre VPC. Par exemple, vous pouvez créer un groupe de sécurité VPC qui autorise l'accès au port TCP 80 pour les serveurs Web de votre VPC. Vous pouvez ensuite créer un autre groupe de sécurité VPC qui autorise l'accès au port TCP 3306 pour les instances de base de données Aurora MySQL de votre VPC.

> **Note**
>
> Dans un cluster de bases de données Aurora, le groupe de sécurité VPC associé au cluster de bases de données est également associé à toutes les instances de base de données du cluster de bases de données. Si vous modifiez le groupe de sécurité VPC associé au cluster de base de données ou à une instance de base de données, la modification est automatiquement appliquée à toutes les instances de base de données du cluster de base de données.

Pour plus d'informations sur les groupes de sécurité VPC, consultez Groupes de sécurité dans le Guide de l'utilisateur Amazon Virtual Private Cloud.

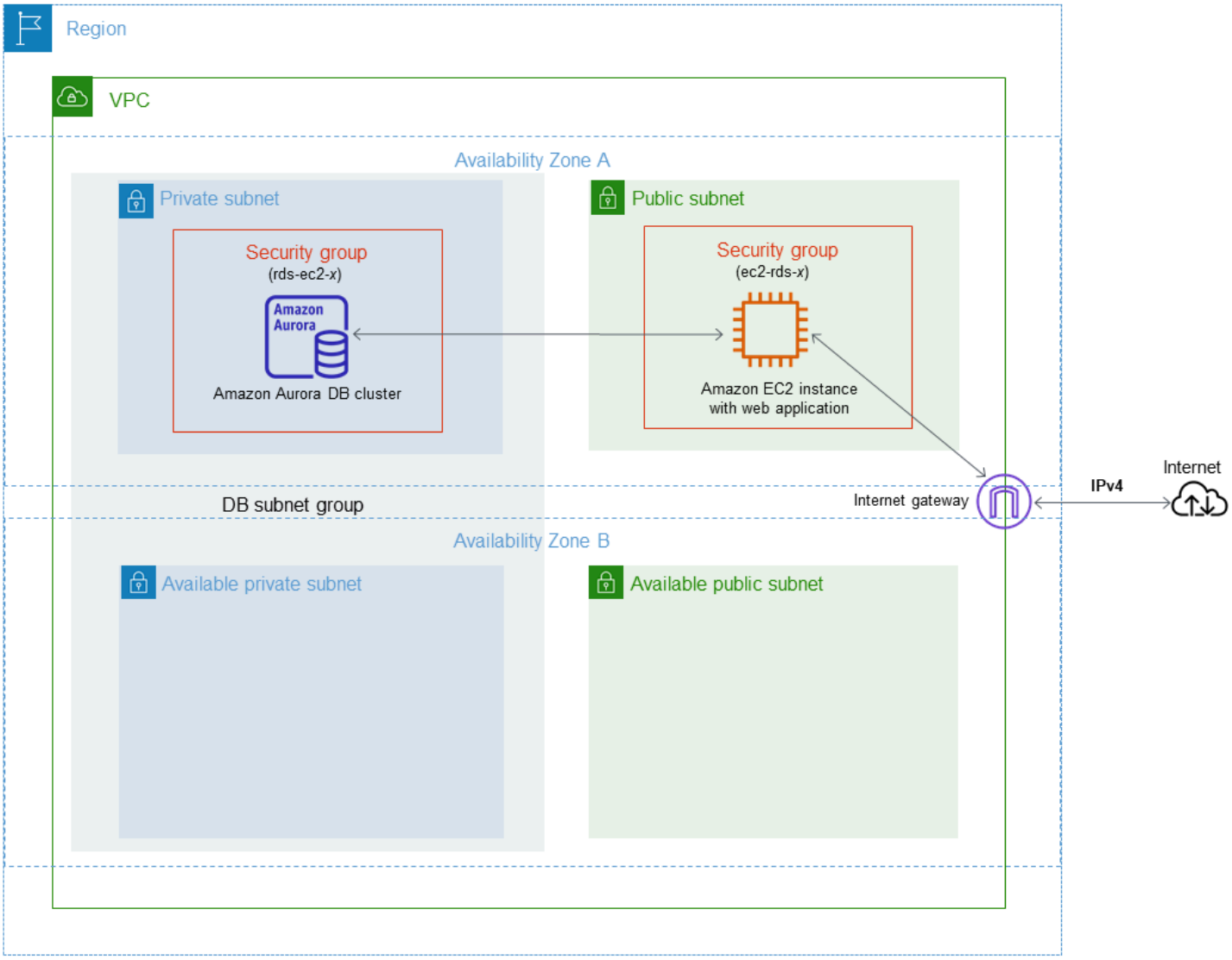
> **Note**
>
> Si votre cluster de bases de données se trouve dans un VPC mais n'est pas accessible publiquement, vous pouvez également utiliser une connexion AWS Site-to-Site VPN ou une connexion Direct Connect pour y accéder à partir d'un réseau privé. Pour plus d'informations, consultez Confidentialité du trafic inter-réseau.

Scénario de groupes de sécurité

Une utilisation courante d'un cluster de bases de données dans un VPC consiste à partager les données avec un serveur d'application qui s'exécute dans une instance Amazon EC2 dans le même VPC et auquel accède une application cliente située hors du VPC. Dans ce scénario, vous utilisez les pages RDS et VPC sur la AWS Management Console ou les opérations d'API RDS et EC2 pour créer les instances et les groupes de sécurité nécessaires :

1. Créez un groupe de sécurité VPC (par exemple, `sg-0123ec2example`) et définissez des règles entrantes qui utilisent les adresses IP de l'application cliente comme source. Ce groupe de sécurité autorise votre application cliente à se connecter aux instances EC2 dans un VPC qui utilise ce groupe de sécurité.
2. Créez une instance EC2 pour l'application et ajoutez l'instance EC2 au groupe de sécurité VPC (`sg-0123ec2example`) que vous avez créé à l'étape précédente.
3. Créez un second groupe de sécurité VPC (par exemple, `sg-6789rdsexample`) et créez une nouvelle règle en spécifiant le groupe de sécurité VPC que vous avez créé à l'étape 1 (`sg-0123ec2example`) en tant que source.
4. Créez un cluster de bases de données et ajoutez le cluster de bases de données au groupe de sécurité VPC (`sg-6789rdsexample`) que vous avez créé à l'étape précédente. Lorsque vous créez le cluster de bases de données, utilisez le même numéro de port que celui spécifié pour la règle du groupe de sécurité VPC (`sg-6789rdsexample`) que vous avez créée à l'étape 3.

Le schéma suivant illustre ce scénario.



Pour des instructions détaillées sur la configuration d'un VPC pour ce scénario, consultez Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement). Pour plus d'informations sur l'utilisation d'un VPC, consultez Amazon VPC et Amazon Aurora.

Création d'un groupe de sécurité VPC

Vous pouvez créer un groupe de sécurité VPC pour une instance de base de données à l'aide de la console VPC. Pour plus d'informations sur la création d'un groupe de sécurité, consultez Créer un groupe de sécurité qui autorise l'accès au cluster de bases de données dans le VPC et Groupes de sécurité dans le Guide de l'utilisateur Amazon Virtual Private Cloud.

Association d'un groupe de sécurité à un cluster de bases de données

Vous pouvez associer un groupe de sécurité à un cluster de bases de données à l'aide de l'option Modifier le cluster sur la console RDS, de l'API `ModifyDBCluster` Amazon RDS ou de la commande `modify-db-cluster` de l'AWS CLI.

L'exemple de commande CLI suivant associe un groupe VPC spécifique et supprime les groupes de sécurité de base de données d'un cluster de bases de données

```
aws rds modify-db-cluster --db-cluster-identifier dbName --vpc-security-group-ids sg-ID
```

Pour plus d'informations sur la modification d'un cluster de bases de données, consultez Modification d'un cluster de bases de données Amazon Aurora.

Privilèges du compte utilisateur principal

Lorsque vous créez un nouveau cluster de bases de données, l'utilisateur principal par défaut que vous utilisez obtient certains privilèges pour ce cluster de bases de données. Vous ne pouvez pas changer le nom de l'utilisateur principal après la création du cluster de bases de données.

> **Important**
>
> Nous vous recommandons vivement de ne pas avoir recours au rôle d'utilisateur principal directement dans vos applications. Au lieu de cela, respectez la bonne pratique qui consiste à avoir recours à un utilisateur de base de données doté des privilèges minimum requis pour votre application.

> **Note**
>
> Si vous supprimez par mégarde les autorisations de l'utilisateur principal, vous pouvez les restaurer en modifiant le cluster de bases de données et en définissant un nouveau mot de passe d'utilisateur principal. Pour plus d'informations sur la modification d'un cluster de bases de données, consultez Modification d'un cluster de bases de données Amazon Aurora.

Le tableau suivant montre les privilèges et les rôles de base de données que l'utilisateur principal obtient pour chacun des moteurs de base de données.

Moteur de base de données	Privilège système	Rôle de base de données
Aurora MySQL	Version 2 : `ALTER,ALTER ROUTINE, CREATE,CREATE ROUTINE,CREATE TEMPORARY TABLES,CREATE USER,CREATE VIEW,DELETE,DROP, EVENT,EXECUTE,GRANT OPTION,INDEX,INSERT, LOAD FROM S3,LOCK TABLES, PROCESS,REFERENCES , RELOAD,REPLICATI ON CLIENT , REPLICATION SLAVE ,SELECT, SELECT INTO S3,SHOW DATABASES , SHOW VIEW,TRIGGER, UPDATE`	—
	Version 3 : `ALTER,APPLICATION_PASSWORD_ADMIN , ALTER ROUTINE,CONNECTION_ADMIN , CREATE,CREATE ROLE,CREATE ROUTINE,CREATE TEMPORARY TABLES, CREATE USER,CREATE VIEW, DELETE,DROP,DROP ROLE, EVENT,EXECUTE,INDEX, INSERT,LOCK TABLES, PROCESS,REFERENCES , RELOAD,REPLICATION CLIENT , REPLICATI ON SLAVE ,ROLE_ADMIN , SET_USER_ ID ,SELECT,SHOW DATABASES ,SHOW VIEW,TRIGGER, UPDATE,XA_RECOVER_ADMIN`  À partir d'Aurora MySQL version 3.04.0, l'utilisateur principal obtient également le privilège `SHOW_ROUT INE` .  À partir d'Aurora MySQL version 3.09.0, l'utilisateur principal obtient également les privilèges `FLUSH_OPT`	`rds_superuser_role` Pour plus d'informations sur rds_superuser_role, consultez Modèle de privilège basé sur les rôles.

Moteur de base de données	Privilège système	Rôle de base de données
	`IMIZER_COSTS` , `FLUSH_STATUS` , `FLUSH_TABLES` et `FLUSH_USER_RESOURCES` .	
Aurora PostgreSQL	`LOGIN`,`NOSUPERUSER` , `INHERIT`,`CREATEDB`, `CREATEROLE` ,`NOREPLICATION` ,`VALID UNTIL 'infinity'`	`RDS_SUPERUSER` Pour plus d'informations sur RDS_SUPERUSER, consultez Comprendre les rôles et les autorisations PostgreSQL.

Utilisation des rôles liés à un service pour Amazon Aurora

Amazon Aurora utilise des rôles liés à un service pour Gestion des identités et des accès AWS (IAM) Un rôle lié à un service est un type unique de rôle IAM lié directement à Amazon Aurora. Les rôles liés à un service sont prédéfinis par Amazon Aurora et comprennent toutes les autorisations dont le service a besoin pour appeler d'autres services AWS en votre nom.

Un rôle lié à un service simplifie l'utilisation d'Amazon Aurora, car vous n'avez pas besoin d'ajouter manuellement les autorisations requises. Amazon Aurora définit les autorisations de ses rôles liés à un service et, sauf définition contraire, seul Amazon Aurora peut endosser ses rôles. Les autorisations définies comprennent la politique d'approbation et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer les rôles uniquement après la suppression préalable de leurs ressources connexes. Vos ressources Amazon Aurora sont ainsi protégées, car vous ne pouvez pas involontairement supprimer l'autorisation d'accéder aux ressources.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés à un service, consultez AWS services that work with IAM (Services AWS qui fonctionnent avec IAM) et recherchez les services avec un Yes (Oui) dans la colonne Service-Linked Role (Rôle lié à un service). Choisissez un Yes (oui) ayant un lien permettant de consulter les détails du rôle pour ce service.

Autorisations des rôles liés à un service pour Amazon Aurora

Amazon Aurora utilise le rôle lié à un service nommé AWSServiceRoleForRDS for permettre à Amazon RDS d'appeler des services AWS pour le compte de vos clusters de bases de données.

Le rôle lié à un service AWSServiceRoleForRDS approuve les services suivants pour endosser le rôle :

- `rds.amazonaws.com`

Ce rôle lié à un service est associé à une politique appelée `AmazonRDSServiceRolePolicy` qui lui accorde l'autorisation d'opérer dans votre compte.

Pour plus d'informations sur cette politique, y compris le document de politique JSON, consultez AmazonRDSServiceRolePolicy dans le Guide de référence des politiques gérées par AWS.

> **Note**
>
> Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, groupe ou rôle) de créer, modifier ou supprimer un rôle lié à un service. Si vous rencontrez le message d'erreur suivant :
> Impossible de créer la ressource. Vérifiez que vous détenez l'autorisation de créer un rôle lié au service. Dans le cas contraire, attendez et réessayez ultérieurement.
> Vérifiez que les autorisations suivantes sont activées :
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/rds.amazonaws.com/
> AWSServiceRoleForRDS",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon Aurora

Vous n'avez pas besoin de créer manuellement un rôle lié à un service. Lorsque vous créez un cluster de bases de données, Amazon Aurora crée le rôle lié à un service pour vous.

> **Important**
>
> Si vous utilisiez le service Amazon Aurora avant le 1er décembre 2017, date à laquelle il a commencé à prendre en charge les rôles liés à un service, Amazon Aurora a créé le rôle AWSServiceRoleForRDS dans votre compte. Pour en savoir plus, consultez Un nouveau rôle est apparu dans mon compte AWS.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez une un cluster de bases de données, Amazon Aurora crée de nouveau le rôle lié à un service pour vous.

Modification d'un rôle lié à un service pour Amazon Aurora

Amazon Aurora ne vous permet pas de modifier le rôle lié à un service AWSServiceRoleForRDS. Une fois que vous avez créé un rôle lié à un service, vous ne pouvez pas modifier le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Suppression d'un rôle lié à un service pour Amazon Aurora

Si vous n'avez plus besoin d'utiliser une fonction ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Cependant, vous devez supprimer toutes vos et clusters de bases de données avant de pouvoir supprimer le rôle lié à un service.

Nettoyage d'un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez d'abord vérifier qu'aucune session n'est active pour le rôle et supprimer toutes les ressources utilisées par le rôle.

Pour vérifier si une session est active pour le rôle lié à un service dans la console IAM

1. Connectez-vous à la AWS Management Console, puis ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le panneau de navigation de la console IAM, choisissez Rôles. Ensuite, sélectionnez le nom (pas la case à cocher) du rôle AWSServiceRoleForRDS.
3. Sur la page Récapitulatif du rôle sélectionné, choisissez l'onglet Dernier accès.
4. Dans l'onglet Dernier accès, consultez l'activité récente pour le rôle lié à un service.

> **Note**
>
> Si vous ignorez si Amazon Aurora utilise le rôle AWSServiceRoleForRDS, vous pouvez essayer de supprimer le rôle. Si le service utilise le rôle, la suppression échoue et vous avez accès aux régions AWS dans lesquelles le rôle est utilisé. Si le rôle est utilisé, vous

devez attendre que la session se termine avant de pouvoir le supprimer. Vous ne pouvez pas révoquer la session d'un rôle lié à un service.

Si vous souhaitez supprimer le rôle AWSServiceRoleForRDS, vous devez commencer par supprimer toutes vos clusters de bases de données.

Suppression de tous vos clusters

Utilisez l'une des procédures suivantes pour supprimer un seul cluster. Répétez la procédure pour chacun de vos clusters.

Pour supprimer un cluster (console)

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans la liste Bases de données, choisissez le cluster que vous souhaitez supprimer.
3. Pour Cluster Actions (Actions de cluster), choisissez Delete (Supprimer).
4. Sélectionnez Delete.

Pour supprimer un cluster (CLI)

Consultez `delete-db-cluster` dans la Référence de commande AWS CLI.

Pour supprimer un cluster (API)

Consultez `DeleteDBCluster` dans le Amazon RDS API Reference.

Vous pouvez utiliser la console IAM, la CLI IAM ou l'API IAM pour supprimer le rôle lié à un service AWSServiceRoleForRDS. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Autorisations du rôle lié à un service pour Amazon RDS bêta

Amazon Aurora utilise le rôle lié à un service nommé `AWSServiceRoleForRDSBeta` pour permettre à Amazon Aurora d'appeler des services AWS au nom de vos ressources de bases de données RDS.

Le rôle lié à un service AWSServiceRoleForRDSBeta approuve les services suivants pour endosser le rôle :

- `rds.amazonaws.com`

Ce rôle lié à un service est associé à une politique appelée `AmazonRDSBetaServiceRolePolicy` qui lui accorde l'autorisation d'opérer dans votre compte. Pour plus d'informations, consultez Politique gérée par AWS : AmazonRDSBetaServiceRolePolicy.

> **Note**
>
> Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, groupe ou rôle) de créer, modifier ou supprimer un rôle lié à un service. Si vous rencontrez le message d'erreur suivant :
> Impossible de créer la ressource. Vérifiez que vous détenez l'autorisation de créer un rôle lié au service. Dans le cas contraire, attendez et réessayez ultérieurement.
> Vérifiez que les autorisations suivantes sont activées :
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSBetaServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Rôles liés à un service pour Amazon RDS Preview

Amazon Aurora utilise le rôle lié à un service nommé `AWSServiceRoleForRDSPreview` pour permettre à Amazon Aurora d'appeler des services AWS au nom de vos ressources de bases de données RDS.

Le rôle lié à un service AWSServiceRoleForRDSPreview approuve les services suivants pour assumer le rôle :

- `rds.amazonaws.com`

Ce rôle lié à un service est associé à une politique appelée `AmazonRDSPreviewServiceRolePolicy` qui lui accorde l'autorisation d'opérer dans votre compte. Pour plus d'informations, consultez Politique gérée par AWS : AmazonRDSPreviewServiceRolePolicy.

> **Note**
>
> Vous devez configurer les autorisations de manière à permettre à une entité IAM (comme un utilisateur, groupe ou rôle) de créer, modifier ou supprimer un rôle lié à un service. Si vous rencontrez le message d'erreur suivant :
> Impossible de créer la ressource. Vérifiez que vous détenez l'autorisation de créer un rôle lié au service. Dans le cas contraire, attendez et réessayez ultérieurement.
> Vérifiez que les autorisations suivantes sont activées :
>
> ```
> {
>     "Action": "iam:CreateServiceLinkedRole",
>     "Effect": "Allow",
>     "Resource": "arn:aws:iam::*:role/aws-service-role/custom.rds.amazonaws.com/
> AmazonRDSPreviewServiceRolePolicy",
>     "Condition": {
>         "StringLike": {
>             "iam:AWSServiceName":"custom.rds.amazonaws.com"
>         }
>     }
> }
> ```
>
> Pour plus d'informations, consultez Autorisations de rôles liés à un service dans le Guide de l'utilisateur IAM.

Amazon VPC et Amazon Aurora

Amazon Virtual Private Cloud (Amazon VPC) vous permet de lancer des ressources AWS, telles que des clusters de base de données Aurora, dans un cloud privé virtuel (VPC).

Lorsque vous utilisez un VPC, vous disposez d'un contrôle total sur l'environnement de réseau virtuel. Vous pouvez choisir votre propre plage d'adresses IP, créer des sous-réseaux et configurer le routage et les listes de contrôle d'accès. Il n'y a pas de frais supplémentaires pour exécuter votre cluster de bases de données dans un VPC.

Les comptes disposent d'un VPC par défaut. Tou(te)s les nouveaux clusters de base de données sont créées dans le VPC par défaut, à moins que vous ne spécifiez une autre option.

Rubriques

Vous trouverez ci-dessous une discussion sur la fonctionnalité VPC pertinente pour les clusters de base de données Amazon Aurora. Pour plus d'informations sur Amazon VPC, consultez le Guide de mise en route Amazon VPC et le Guide de l'utilisateur Amazon VPC.

Utilisation d'un cluster de bases de données dans un VPC

Votre cluster de bases de données se trouve dans un cloud privé virtuel (VPC). Un VPC est un réseau virtuel isolé logiquement des autres réseaux virtuels du cloud. AWS Amazon VPC vous permet de lancer des AWS ressources, telles qu'un cluster d'instances de base de données Amazon Aurora ou une Amazon, dans un VPC. Le VPC peut être un VPC par défaut fourni avec votre compte ou un VPC que vous créez. Ils VPCs sont tous associés à votre AWS compte.

Votre VPC par défaut a trois sous-réseaux que vous pouvez utiliser pour isoler les ressources à l'intérieur du VPC. Le VPC par défaut possède aussi une passerelle Internet qui peut être utilisée pour fournir l'accès aux ressources à l'intérieur du VPC depuis l'extérieur du VPC.

Pour obtenir une liste des scénarios impliquant des clusters de bases de données Amazon Aurora dans un VPC et , consultez Scénarios d'accès à un cluster de bases de données d'un VPC.

Rubriques

Dans les tutoriels suivants, vous apprendrez à créer un VPC que vous pouvez utiliser pour un scénario commun Amazon Aurora :

Utilisation d’un cluster de bases de données dans un VPC

Voici quelques conseils d’utilisation d’un cluster de bases de données dans un VPC :

- Votre VPC doit avoir au moins deux sous-réseaux. Ces sous-réseaux doivent se trouver dans deux zones de disponibilité différentes dans l' Région AWS endroit où vous souhaitez déployer votre cluster d' de base de données. Un sous-réseau est un segment de la plage d’adresses IP d’un VPC que vous pouvez spécifier et que vous pouvez utiliser pour regrouper des clusters de bases de données en fonction de vos besoins en matière de sécurité et de fonctionnement.
- Si vous voulez que votre cluster de bases de données dans le VPC soit publiquement accessible, assurez-vous d’activer les attributs VPC DNS hostnames (Noms d’hôtes DNS) et DNS resolution (Résolution DNS).
- Votre VPC doit disposer d’un groupe de sous-réseau de base de données que vous créez. Vous créez un groupe de sous-réseaux de base de données en spécifiant les sous-réseaux que vous avez créés. Amazon Aurora choisit un sous-réseau et une adresse IP dans ce sous-réseau pour les associer avec l’instance de base de données principale dans votre cluster de bases de données. L’instance de base de données principale utilise la zone de disponibilité contenant le sous-réseau.
- Votre VPC doit avoir un groupe de sécurité VPC qui autorise l’accès au cluster de bases de données.

Pour plus d’informations, consultez Scénarios d’accès à un cluster de bases de données d’un VPC.

- Les blocs CIDR de chacun de vos sous-réseaux doivent être assez grands pour accueillir les adresses IP de rechange utilisées par Amazon Aurora pendant les activités de maintenance, y compris le basculement et le dimensionnement du calcul. Par exemple, une plage telle que 10.0.0.0/24 et 10.0.1.0/24 est généralement suffisante.
- Un VPC peut avoir un attribut instance tenancy (location d’instance) ayant la valeur par défaut ou dédiée. Tous les paramètres par défaut VPCs ont l'attribut de location d'instance défini sur défaut, et un VPC par défaut peut prendre en charge n'importe quelle classe d'instance de base de données.

  Si vous choisissez de placer votre cluster d' de base de données dans un VPC dédié où l'attribut de location d'instance est défini sur dedicated, la classe d'instance de votre cluster d'instance de base de données doit être l'un des types d' EC2 dédiés Amazon approuvés. Par exemple, l'instance EC2 dédiée r5.large correspond à la classe d'instance de base de données db.r5.large. Pour plus d’informations sur la location d’instance dans un VPC, consultez Instances dédiées dans le Guide de l’utilisateur Amazon Elastic Compute Cloud.

  Pour plus d'informations sur les types d'instances qui peuvent figurer dans une instance dédiée, consultez la section Instances EC2 dédiées Amazon sur la page de EC2 tarification Amazon.

  > **Note**
  >
  > Lorsque vous définissez l’attribut de location d’instance sur dédié pour un cluster de bases de données, cela ne garantit pas que le cluster de bases de données fonctionnera sur un hôte dédié.

Contrôle du chiffrement VPC

Les contrôles de chiffrement VPC vous permettent de les appliquer encryption-in-transit à l'ensemble du trafic réseau au sein de votre. VPCs Utilisez le contrôle du chiffrement pour répondre aux exigences de conformité réglementaire en vous assurant que seul le matériel Nitro compatible avec le chiffrement peut être fourni dans un environnement désigné. VPCs Le contrôle du chiffrement permet également de détecter les problèmes de compatibilité au moment de la demande d'API plutôt que lors du provisionnement. Vos charges de travail existantes continuent de fonctionner et seules les nouvelles demandes incompatibles sont bloquées.

Définissez vos contrôles de chiffrement VPC en configurant le mode de contrôle VPC de manière à :

- désactivé (par défaut)
- moniteur
- appliqué

Pour vérifier le mode de contrôle actuel de votre VPC, utilisez la AWS Management Console commande ou la commande API ou DescribeVpcsCLI.

Si votre VPC applique le chiffrement, vous ne pouvez fournir que des de base de données de clusters de base de données basées sur Nitro qui prennent en charge le chiffrement en transit dans ce VPC. Pour plus d'informations, voir,Types de classes d'instance de base de données. Pour plus d'informations sur les instances Nitro, consultez la section Instances créées sur le système AWS Nitro dans le guide de EC2 l'utilisateur Amazon.

> **Note**
>
> Si vous essayez de provisionner des de données incompatibles dans un VPC basé sur le chiffrement, Aurora RDS renvoie une exception. `VpcEncryptionControlViolationException`

Aurora ServerlessPour MySQL et PostreSQL, le contrôle du chiffrement nécessite la version 3 ou une version ultérieure de la plate-forme.

Utilisation de groupes de sous-réseaux DB

Les sous-réseaux sont des segments d'une plage d'adresses IP d'un VPC que vous définissez pour regrouper vos ressources en fonction de vos besoins de sécurité et de fonctionnement. Un groupe de sous-réseaux de base de données est une collection de sous-réseaux (généralement privés) que vous créez dans un VPC et que vous spécifiez alors pour vos clusters de base de données. En utilisant un groupe de sous-réseaux de base de données, vous pouvez spécifier un VPC particulier lors de la création de clusters d' de base de données à l'aide de l'API ou AWS CLI RDS. Si vous utilisez la console, vous pouvez choisir le VPC et les groupes de sous-réseaux que vous voulez utiliser.

Chaque groupe de sous-réseaux de base de données doit avoir des sous-réseaux dans au moins deux zones de disponibilité d'une Région AWS donnée. Lorsque vous créez un cluster de bases

de données dans un VPC, vous choisissez un groupe de sous-réseau de base de données pour celui-ci. Dans le groupe de sous-réseaux de base de données, Amazon Aurora choisit un sous-réseau et une adresse IP dans ce sous-réseau pour les employer avec l'instance de base de données principale dans votre cluster de bases de données. La base de données utilise la zone de disponibilité contenant le sous-réseau. Aurora attribue toujours une adresse IP à partir d'un sous-réseau disposant d'un espace d'adresse IP libre.

Les sous-réseaux d'un groupe de sous-réseaux de base de données sont publics ou privés. Les sous-réseaux sont publics ou privés, selon la configuration que vous définissez pour leurs listes de contrôle d'accès réseau (réseau ACLs) et leurs tables de routage. Pour qu'un cluster de bases de données soit accessible au public, tous les sous-réseaux de son groupe de sous-réseaux de base de données doivent être publics. Si un sous-réseau associé à un cluster de bases de données accessible au public passe de public à privé, cela peut affecter la disponibilité du cluster de bases de données.

Pour créer un groupe de sous-réseaux de base de données prenant en charge le mode double pile, assurez-vous qu'un bloc CIDR du protocole Internet version 6 (IPv6) est associé à chaque sous-réseau que vous ajoutez au groupe de sous-réseaux de base de données. Pour plus d'informations, consultez Adressage IP Amazon Aurora la section « Migration vers » IPv6 dans le guide de l'utilisateur Amazon VPC.

Lorsque Amazon Aurora crée un cluster de bases de données dans un VPC, il attribue une interface réseau à votre cluster de bases de données en utilisant une adresse IP de votre groupe de sous-réseau de base de données. Toutefois, nous vous recommandons vivement d'utiliser le nom du système de nom de domaine (DNS) pour vous connecter à votre cluster de bases de données. Nous le recommandons, car l'adresse IP sous-jacente change pendant le basculement.

> **Note**
>
> Pour chaque cluster de bases de données que vous exécutez dans un VPC, assurez-vous de réserver au moins une adresse dans chaque sous-réseau du groupe de sous-réseaux de base de données qui sera utilisée par Amazon Aurora pour les actions de récupération.

Sous-réseaux partagés

Vous pouvez créer une instance de base de données dans un VPC partagé.

Voici quelques points à prendre en compte lors de l'utilisation du partage VPCs :

- Vous pouvez déplacer un cluster de bases de données d’un sous-réseau VPC partagé vers un sous-réseau VPC non partagé et vice-versa.
- Les participants à un VPC partagé doivent créer un groupe de sécurité dans le VPC pour pouvoir créer un cluster de bases de données.
- Les propriétaires et les participants d’un VPC partagé peuvent accéder à la base de données à l’aide de requêtes SQL. Toutefois, seul le créateur d’une ressource peut effectuer des appels d’API sur cette ressource.

Adressage IP Amazon Aurora

Les adresses IP permettent aux ressources de votre VPC de communiquer entre elles et avec les ressources sur Internet. Amazon Aurora prend en charge les deux protocoles ainsi que les protocoles IPv4 d' IPv6 adressage. Par défaut, , Amazon Aurora et Amazon VPC utilisent IPv4 le protocole d'adressage. Vous ne pouvez pas désactiver ce comportement. Lorsque vous créez un VPC, assurez-vous de spécifier un bloc IPv4 CIDR (une plage d'adresses privées IPv4). Vous pouvez éventuellement attribuer un bloc IPv6 CIDR à votre VPC et à vos sous-réseaux, et IPv6 attribuer les adresses de ce bloc aux clusters d' de base de données de votre sous-réseau.

Support du IPv6 protocole augmente le nombre d'adresses IP prises en charge. En utilisant le IPv6 protocole, vous vous assurez de disposer de suffisamment d'adresses disponibles pour la future croissance d'Internet. Les ressources et IPv6 adresses RDS nouvelles et existantes peuvent être utilisées IPv4 au sein de votre VPC. La configuration, la sécurisation et la traduction du trafic réseau entre les deux protocoles utilisés dans les différentes parties d’une application peuvent entraîner une surcharge opérationnelle. Vous pouvez standardiser le IPv6 protocole des ressources Amazon RDS afin de simplifier la configuration de votre réseau.

Rubriques

- IPv4 adresses
- IPv6 adresses
- Mode double pile

IPv4 adresses

Lorsque vous créez un VPC, vous devez spécifier une plage d' IPv4 adresses pour le VPC sous la forme d'un bloc CIDR, tel que. `10.0.0.0/16` Un groupe de sous-réseau de base de données

définit la plage d’adresses IP de ce bloc CIDR qu’un cluster de bases de données peut utiliser. Ces adresses IP peuvent être privées ou publiques.

Une IPv4 adresse privée est une adresse IP qui n'est pas accessible via Internet. Vous pouvez utiliser IPv4 des adresses privées pour la communication entre votre cluster d' de base de données et d'autres ressources, telles que EC2 les instances Amazon, dans le même VPC. Chaque cluster de bases de données dispose d’une adresse IP privée pour la communication dans le VPC.

Une adresse IP publique est une IPv4 adresse accessible depuis Internet. Vous pouvez utiliser des adresses publiques pour la communication entre votre cluster de bases de données et des ressources sur Internet, comme un client SQL. Vous contrôlez si votre cluster de bases de données reçoit une adresse IP publique.

Pour un didacticiel expliquant comment créer un VPC avec uniquement des IPv4 adresses privées que vous pouvez utiliser pour un scénario Amazon , consultez. Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement)

IPv6 adresses

Vous pouvez éventuellement associer un bloc IPv6 CIDR à votre VPC et à vos sous-réseaux, et IPv6 attribuer des adresses de ce bloc aux ressources de votre VPC. Chaque IPv6 adresse est unique au monde.

Le bloc IPv6 CIDR pour votre VPC est automatiquement attribué à partir du pool IPv6 d'adresses d'Amazon. Vous ne pouvez pas choisir la plage vous-même.

Lorsque vous vous connectez à une IPv6 adresse, assurez-vous que les conditions suivantes sont remplies :

- Le client est configuré de telle sorte que le trafic entre le client et la base de données IPv6 soit autorisé.
- Les groupes de sécurité RDS utilisés par l'instance de base de données sont correctement configurés afin que le trafic client-base de données IPv6 soit autorisé.
- La pile du système d'exploitation client autorise le trafic sur l' IPv6 adresse, et les pilotes et bibliothèques du système d'exploitation sont configurés pour choisir le point de terminaison d'instance de base de données par défaut correct (IPv4 ou non IPv6).

Pour plus d'informations IPv6, consultez la section Adressage IP dans le guide de l'utilisateur Amazon VPC.

Mode double pile

Lorsqu'un cluster d' de base de données peut communiquer à la fois via le protocole IPv4 et le protocole d'IPv6 adressage, il fonctionne en mode double pile. Ainsi, les ressources peuvent communiquer avec le cluster d' de base de IPv4 données IPv6 via ou les deux. RDS désactive l'accès à Internet Gateway pour les IPv6 points de terminaison des instances de base de données privées en mode double pile. RDS fait cela pour garantir que vos IPv6 points de terminaison sont privés et ne sont accessibles que depuis votre VPC.

Rubriques

- Mode double pile et groupes de sous-réseaux de base de données
- Utilisation d’instances de base de données en mode double pile
- Modification de IPv4 clusters d' de base de données réservés uniquement pour utiliser le mode double pile
- Disponibilité de clusters de bases de données en réseau à double pile
- Limitations pour les clusters de base de données en réseau à double pile

Pour un didacticiel expliquant comment créer un VPC à la fois avec IPv4 des IPv6 adresses que vous pouvez utiliser pour un scénario Amazon , consultez. Tutoriel : Créer un VPC à utiliser avec un cluster de bases de données (mode double-pile)

Mode double pile et groupes de sous-réseaux de base de données

Pour utiliser le mode double pile, assurez-vous qu'un bloc IPv6 CIDR est associé à chaque sous-réseau du groupe de sous-réseaux de base de données que vous associez au cluster d' de base de données. Vous pouvez créer un nouveau groupe de sous-réseau de base de données ou modifier un groupe de sous-réseau de base de données existant pour répondre à cette exigence. Une fois qu’un cluster de bases de données est en mode double pile, les clients peuvent s’y connecter normalement. Assurez-vous que les pare-feux de sécurité du client et les groupes de sécurité des instances de base de données RDS sont correctement configurés pour autoriser le transfert du trafic. IPv6 Pour se connecter, les clients utilisent le point de terminaison de l’instance principale du cluster de bases de données. Les applications client peuvent spécifier quel protocole est préféré lors de la connexion à une base de données. En mode double pile, le cluster d' de base de données détecte le protocole réseau préféré du client, soit IPv6, IPv4 soit, et utilise ce protocole pour la connexion.

Si un groupe de sous-réseaux de base de données cesse de prendre en charge le mode double pile en raison de la suppression d’un sous-réseau ou d’une dissociation CIDR, il existe un risque

d’incompatibilité de l’état du réseau pour les instances de base de données associées au groupe de sous-réseaux de base de données. De même, vous ne pouvez pas utiliser le groupe de sous-réseau de base de données lorsque vous créez un cluster de bases de données en mode double pile.

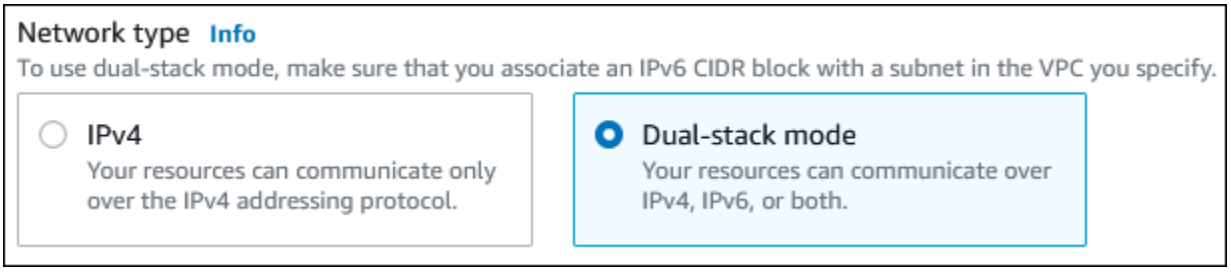
Pour déterminer si un groupe de sous-réseaux de base de données prend en charge le mode double pile à l'aide du AWS Management Console, consultez le type de réseau sur la page de détails du groupe de sous-réseaux de base de données. Pour déterminer si un groupe de sous-réseaux de base de données prend en charge le mode double pile à l'aide de AWS CLI, exécutez la describe-db-subnet-groupscommande et visualisez `SupportedNetworkTypes` la sortie.

Les réplicas en lecture sont traités comme des instances de base de données indépendantes et peuvent avoir un type de réseau différent de celui de l’instance de base de données principale. Si vous modifiez le type de réseau de l’instance de base de données principale d’un réplica en lecture, le réplica en lecture n’est pas affecté. Lorsque vous restaurez une instance de base de données, vous pouvez la restaurer sur tout type de réseau pris en charge.

Utilisation d’instances de base de données en mode double pile

Lorsque vous créez ou modifiez un cluster d' de base de données, vous pouvez spécifier le mode double pile pour permettre à vos ressources de communiquer avec votre cluster d' de base de données IPv4 via ou IPv6 les deux.

Lorsque vous utilisez le AWS Management Console pour créer ou modifier une instance de base de données, vous pouvez spécifier le mode double pile dans la section Type de réseau. L’image suivante présente la section Network type (Type de réseau) dans la console.



Lorsque vous utilisez le AWS CLI pour créer ou modifier un cluster d' de base de données, définissez l'`--network-type`option `DUAL` pour utiliser le mode double pile. Lorsque vous utilisez l’API RDS pour créer ou modifier un cluster de bases de données, définissez le paramètre `NetworkType` sur `DUAL` pour utiliser le mode double pile. Lorsque vous modifiez le type de réseau d’une instance de base de données, une durée d’indisponibilité est possible. Si le mode double pile n’est pas pris en charge par la version du moteur de base de données ou le groupe de sous-réseau de base de données spécifié, l’erreur `NetworkTypeNotSupported` est renvoyée.

Pour plus d'informations sur la création d'un cluster de bases de données, consultez Création d'un cluster de bases de données Amazon Aurora. Pour plus d'informations sur la modification d'un cluster, consultez Modification d'un cluster de bases de données Amazon Aurora.

Pour déterminer si un cluster de bases de données est en mode double pile en utilisant la console, affichez Type de réseau dans l'onglet Connectivité et sécurité pour le cluster de bases de données.

Modification de IPv4 clusters d' de base de données réservés uniquement pour utiliser le mode double pile

Vous pouvez modifier un cluster d' de base de données réservé IPv4 uniquement pour utiliser le mode double pile. Pour ce faire, modifiez le type de réseau du cluster de bases de données. La modification peut entraîner une durée d'indisponibilité.

Nous vous recommandons de modifier le type de réseau de vos clusters de bases de données Amazon Aurora au cours d'une fenêtre de maintenance. Pour l'heure, il n'est pas possible de définir le type de réseau des nouvelles instances sur le mode double pile. Vous pouvez définir le type de réseau manuellement à l'aide de la commande `modify-db-cluster`.

Avant de modifier un cluster de bases de données pour utiliser le mode double pile, assurez-vous que son groupe de sous-réseau de base de données prend en charge le mode double pile. Si le groupe de sous-réseau de base de données associé au cluster de bases de données ne prend pas en charge le mode double pile, spécifiez un autre groupe de sous-réseau de base de données qui le prend en charge lorsque vous modifiez le cluster de bases de données. La modification du groupe de sous-réseaux de base de données d' un cluster de bases de données peut entraîner une interruption de service.

Si vous modifiez le groupe de sous-réseau de base de données d' un cluster de bases de données avant de modifier le cluster de bases de données pour utiliser le mode double pile, assurez-vous que le groupe de sous-réseau de base de données est valide pour le cluster de bases de données avant et après la modification.

Nous vous recommandons d'exécuter l'modify-db-clusterAPI uniquement avec le `--network-type` paramètre ayant une valeur `DUAL` pour faire passer le réseau d'un cluster Amazon Aurora en mode double pile. L'ajout d'autres paramètres en même temps que le paramètre `--network-type` dans le même appel d'API peut entraîner une durée d'indisponibilité.

Si vous ne parvenez pas à vous connecter au cluster d' de base de données après la modification, assurez-vous que les pare-feux de sécurité et les tables de routage du client et de la base de données sont correctement configurés pour autoriser le trafic vers la base de données sur le réseau

sélectionné (IPv4 ou non IPv6). Vous devrez peut-être également modifier les paramètres du système d'exploitation, les bibliothèques ou les pilotes pour vous connecter à l'aide d'une IPv6 adresse.

Pour modifier un cluster d' de base de données réservé IPv4 uniquement afin d'utiliser le mode double pile

1. Modifiez un groupe de sous-réseaux de base de données pour prendre en charge le mode double pile ou créez un groupe de sous-réseaux de base de données qui prend en charge le mode double pile :

   a. Associez un bloc IPv6 CIDR à votre VPC.

      Pour obtenir des instructions, consultez la section Ajouter un bloc IPv6 CIDR à votre VPC dans le guide de l'utilisateur Amazon VPC.

   b. Attachez le bloc IPv6 CIDR à tous les sous-réseaux de votre groupe de sous-réseaux de base de données.

      Pour obtenir des instructions, consultez la section Ajouter un bloc IPv6 CIDR à votre sous-réseau dans le guide de l'utilisateur Amazon VPC.

   c. Confirmez que le groupe de sous-réseaux de base de données prend en charge le mode double pile.

      Si vous utilisez le AWS Management Console, sélectionnez le groupe de sous-réseaux de base de données et assurez-vous que la valeur des types de réseau pris en charge est Dual, IPv4.

      Si vous utilisez le AWS CLI, exécutez la describe-db-subnet-groupscommande et assurez-vous que la `SupportedNetworkType` valeur de l'instance de base de données est`Dual, IPv4`.

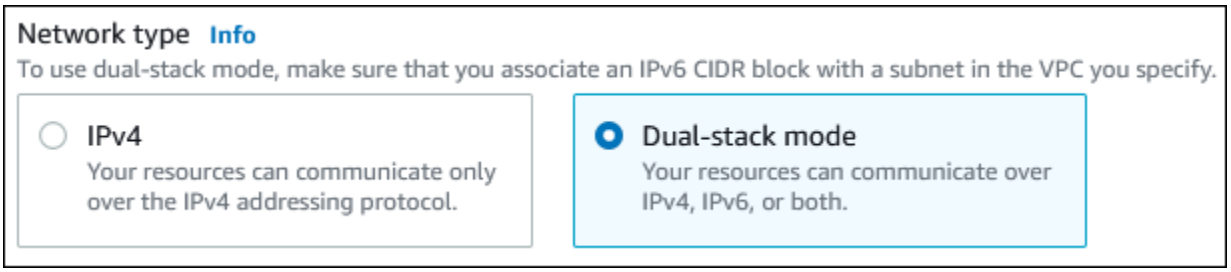
2. Modifiez le groupe de sécurité associé au cluster d' de base de données pour autoriser IPv6 les connexions à la base de données, ou créez un nouveau groupe de sécurité qui autorise IPv6 les connexions.

   Pour obtenir des instructions, consultez Security group rules (Règles des groupes de sécurité) dans le Guide de l’utilisateur Amazon VPC.

3. Modifiez le cluster de la base de données pour qu’il prenne en charge le mode double pile. Pour ce faire, réglez le Network type (Type de réseau) sur Dual-stack mode (Mode double pile).

Si vous utilisez la console, assurez-vous que les paramètres suivants sont corrects :

- Type de réseau : mode double pile



- DB subnet group (Groupe de sous-réseau de base de données) : le groupe de sous-réseau de base de données que vous avez configuré à l'étape précédente.
- Groupe de sécurité : la sécurité que vous avez configurée dans une étape précédente.

Si vous utilisez le AWS CLI, assurez-vous que les paramètres suivants sont corrects :

- `--network-type` – `dual`
- `--db-subnet-group-name` — le groupe de sous-réseau de base de données que vous avez configuré à l'étape précédente.
- `--vpc-security-group-ids` : le groupe de sécurité du VPC que vous avez configuré à l'étape précédente.

Par exemple :

```
aws rds modify-db-cluster --db-cluster-identifier my-cluster --network-type "DUAL"
```

4. Confirmez que le cluster de bases de données prend en charge le mode double pile.

   Si vous utilisez la console, choisissez l'onglet (Connectivité et sécurité)Configuration pour le cluster de la base de données. Dans cet onglet, assurez-vous que la valeur de Network type (Type de réseau) est Dual-stack mode (Mode double pile).

   Si vous utilisez le AWS CLI, exécutez la describe-db-clusterscommande et assurez-vous que la `NetworkType` valeur du cluster de base de données est`dual`.

   Exécutez la `dig` commande sur le point de terminaison de l'instance de base de données du rédacteur pour identifier l' IPv6adresse qui lui est associée.

```
dig db-instance-endpoint AAAA
```

Utilisez le point de terminaison de l'instance de base de données du rédacteur, et non l' IPv6 adresse, pour vous connecter au cluster d' de base de données.

Disponibilité de clusters de bases de données en réseau à double pile

Les clusters de base de données réseau à double pile sont disponibles dans tous les domaines Régions AWS , à l'exception des suivants :

- Asie-Pacifique (Hyderabad)
- Asie-Pacifique (Malaisie)
- Asie-Pacifique (Melbourne)
- Asie-Pacifique (Thaïlande)
- Canada-Ouest (Calgary)
- Europe (Espagne)
- Europe (Zurich)
- Israël (Tel Aviv)
- Mexique (Centre)
- Moyen-Orient (EAU)

Les versions suivantes du moteur de base de données prennent en charge les clusters de bases de données en réseau à double pile :

- Aurora MySQL versions :
  - 3.02 et versions 3 ultérieures
  - 2.09.1 et versions 2 ultérieures

  Pour plus d’informations sur les versions d’Aurora MySQL, consultez Notes de mise à jour d’Aurora MySQL.
- Versions d’Aurora PostgreSQL :
  - 15.2 et toutes les versions ultérieures
  - 14.3 et versions 14 ultérieures

- 13.7 et versions 13 ultérieures

Pour plus d'informations sur les versions d'Aurora PostgreSQL, consultez Notes de mise à jour d'Aurora PostgreSQL.

Limitations pour les clusters de base de données en réseau à double pile

Les limitations suivantes s'appliquent aux clusters de bases de données en réseau à double pile :

- de base de données ne peuvent pas utiliser le IPv6 protocole exclusivement. Ils peuvent utiliser IPv4 exclusivement, ou ils peuvent utiliser le IPv6 protocole IPv4 and (mode double pile).
- Amazon RDS ne prend pas en charge les IPv6 sous-réseaux natifs.
- Les clusters de bases de données qui utilisent le mode double pile doivent être privé(e)s. Ils/elles ne peuvent pas être publiquement accessibles.
- Vous ne pouvez pas utiliser le proxy RDS avec des clusters de bases de données en mode double pile.

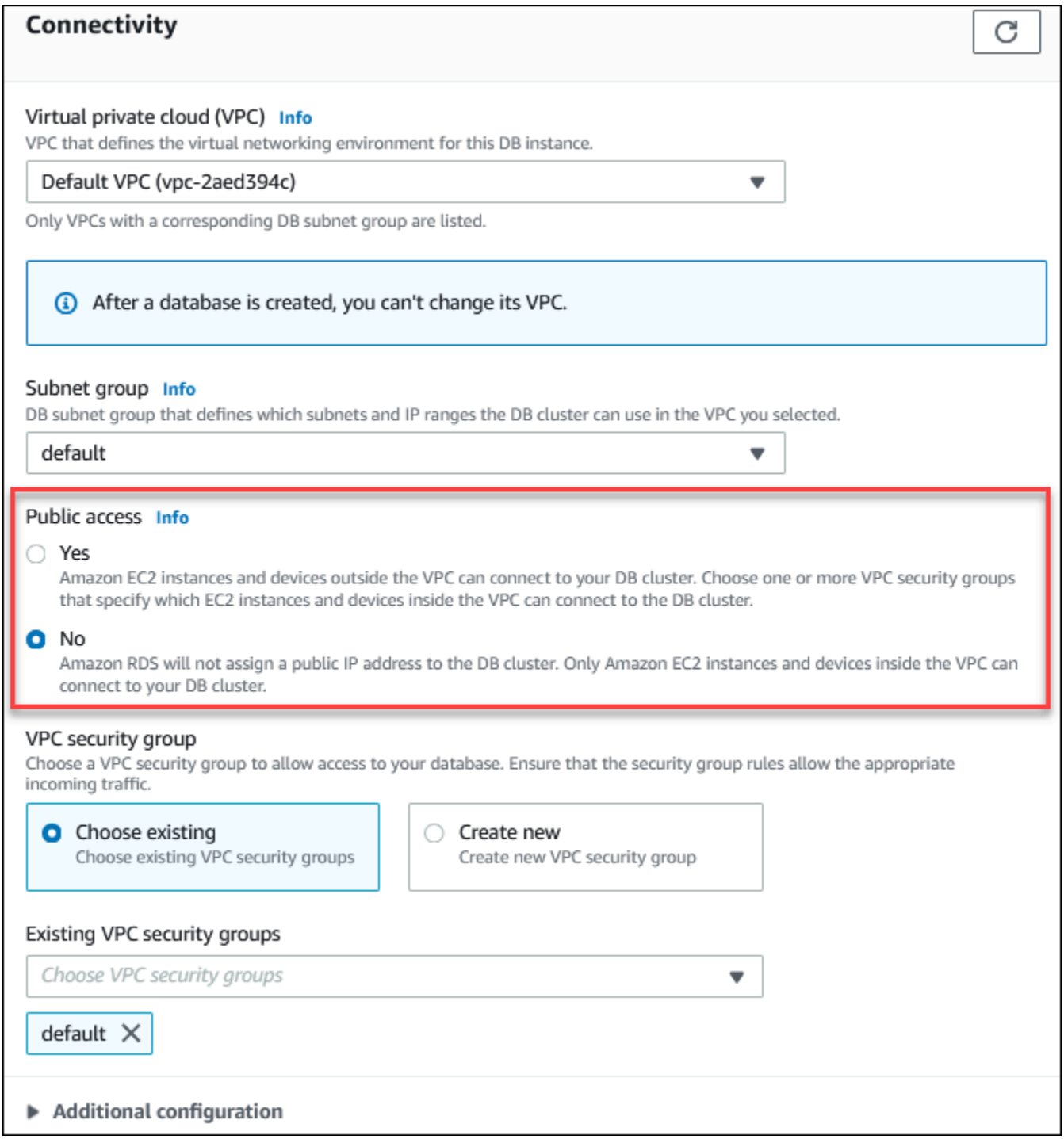
Masquer un cluster de bases de données dans un VPC depuis Internet

Par exemple, vous pouvez créer un VPC doté d'un sous-réseau public et d'un sous-réseau privé. EC2 les instances qui fonctionnent comme des serveurs Web peuvent être déployées dans le sous-réseau public. Les clusters de bases de données sont déployés dans le sous-réseau privé. Dans un tel déploiement, seuls les serveurs web ont accès aux clusters de base de données. Pour obtenir une illustration de ce scénario, consultez Un cluster de bases de données dans un VPC auquel accède une instance EC2 dans le même VPC.

Lorsque vous lancez un cluster de bases de données dans un VPC, le cluster de bases de données possède une adresse IP privée pour le trafic à l'intérieur du VPC. Cette adresse IP privée n'est pas accessible au public. Vous pouvez utiliser l'option Public access (Accès public) pour indiquer si le cluster de bases de données possède également une adresse IP publique en plus de l'adresse IP privée. Si le cluster de la base de données est désigné comme publiquement accessible, son point de terminaison DNS se résout à l'adresse IP privée à partir du VPC. Il renvoit à l'adresse IP publique depuis l'extérieur du VPC. L'accès au cluster de la base de données est contrôlé en dernier ressort par le groupe de sécurité qu'il utilise. Cet accès public n'est pas autorisé si le groupe de sécurité attribué au cluster de la base de données ne comprend pas de règles d'entrée qui l'autorisent. En outre, pour qu'un cluster de bases de données soit publiquement accessible, les sous-réseaux de

son groupe de sous-réseaux de base de données doivent avoir une passerelle Internet. Pour plus d'informations, consultez Impossible de se connecter à l'instance de base de données Amazon RDS.

Vous pouvez modifier un cluster de bases de données pour activer ou désactiver l'accessibilité publique en modifiant l'option Accès public. L'illustration suivante présente l'option Public Access (Accès public) dans la section Additional connectivity configuration (Configuration de connectivité supplémentaire). Pour définir cette option, ouvrez la section Additional connectivity configuration (Configuration de connectivité supplémentaire) dans la section Connectivity (Connectivité).



Pour plus d'informations sur la modification d'une instance de base de données afin de définir l'option Public access (Accès public), consultez Modification d'une instance de base de données dans un cluster de bases de données.

Création d’un cluster de bases de données dans un VPC

Les procédure suivantes vous aident à créer un cluster de bases de données dans un VPC. Pour utiliser le VPC par défaut, vous pouvez commencer par l’étape 2, et utiliser le groupe VPC et sous-réseau de base de données qui ont déjà été créés pour vous. Si vous souhaitez créer un VPC supplémentaire, vous pouvez créer un nouveau VPC.

> **Note**
>
> Si vous voulez que votre cluster de bases de données du VPC soit publiquement accessible, vous devez mettre à jour les informations DNS pour le VPC en activant les attributs VPC DNS hostnames (noms d’hôtes DNS) et DNS resolution (Résolution DNS). Pour plus d’informations sur la mise à jour des informations DNS pour une instance VPC, consultez Mise à jour de la prise en charge DNS pour votre VPC.

Suivez les étapes ci-après pour créer une instance de base de données dans un VPC:

- Étape 1 : Création d’un VPC
- Étape 2 : créer un groupe de sous-réseaux de base de données
- Étape 3 : créer un groupe de sécurité VPC
- Étape 4 : créer une instance de base de données dans le VPC

Étape 1 : Création d’un VPC

Créez un VPC avec des sous-réseaux dans au moins deux zones de disponibilité. Vous utilisez ces sous-réseaux lorsque vous créerez un groupe de sous-réseaux de base de données. Si vous avez un VPC par défaut, un sous-réseau est automatiquement créé pour vous dans chaque zone de disponibilité de la Région AWS.

Pour obtenir plus d’informations, consultez Créer un VPC avec des sous-réseaux publics et privés, ou Create a VPC (Créer un VPC) dans le Guide de l’utilisateur Amazon VPC.

Étape 2 : créer un groupe de sous-réseaux de base de données

Un groupe de sous-réseaux DB est une collection de sous-réseaux (généralement privés) que vous créez pour un VPC et que vous spécifiez alors pour vos clusters de base de données. Un groupe de

sous-réseaux de base de données vous permet de spécifier un VPC particulier lorsque vous créez des clusters d' de base de données à l'aide de l'API ou AWS CLI RDS. Si vous utilisez la console, vous pouvez simplement choisir le VPC et les sous-réseaux que vous voulez utiliser. Chaque groupe de sous-réseaux DB doit avoir au moins un sous-réseau dans au moins deux zones de disponibilité de la Région AWS. La bonne pratique est la suivante : chaque groupe de sous-réseaux de base de données doit être constitué d'au moins un sous-réseau pour chaque zone de disponibilité dans la Région AWS.

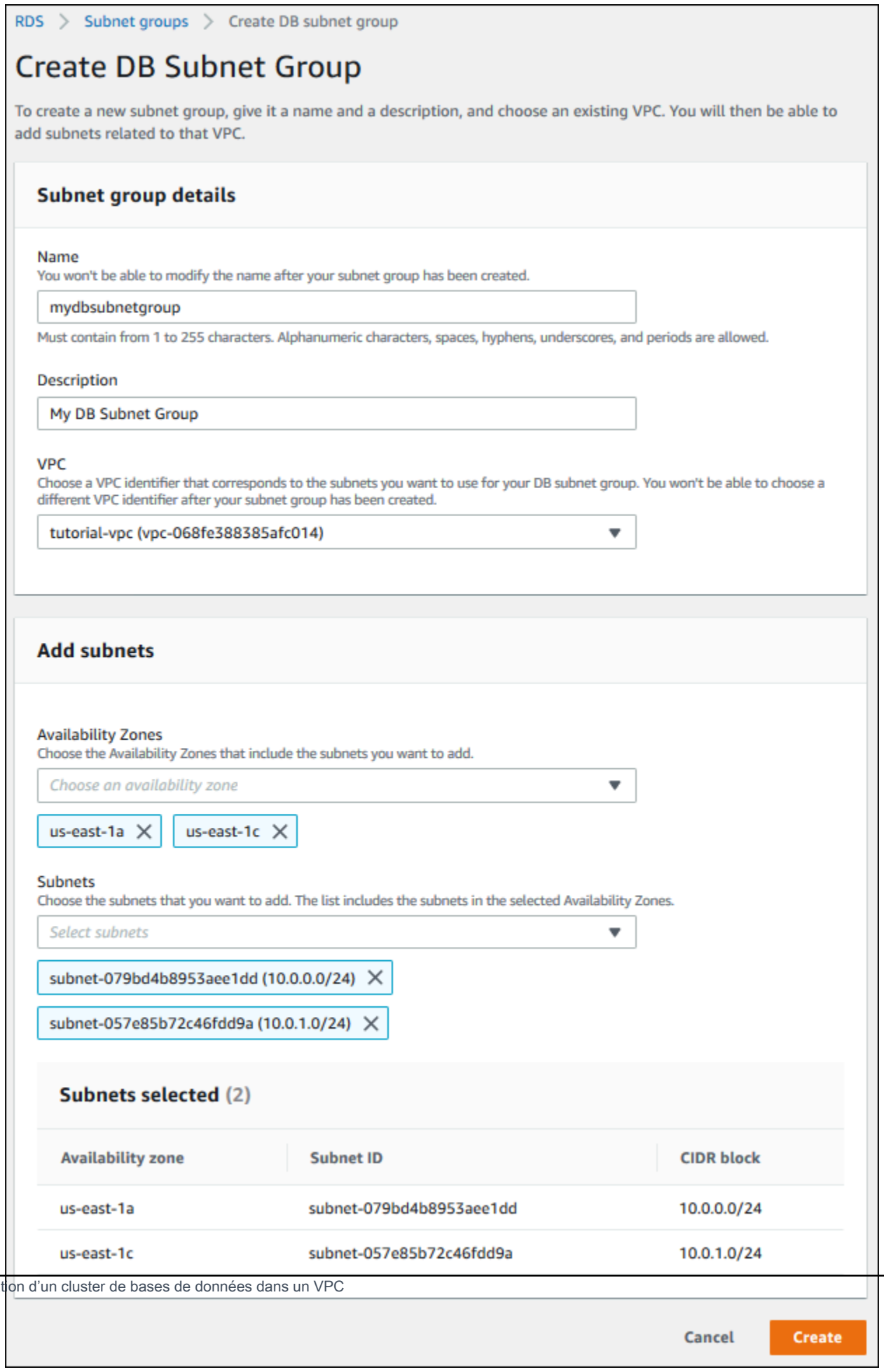
Pour qu'un cluster de bases de données soit publiquement accessible, les sous-réseaux du groupe de sous-réseaux de base de données doivent avoir une passerelle Internet. Pour plus d'informations sur les passerelles Internet pour les sous-réseaux, consultez Connect to the internet using an internet gateway (Se connecter à Internet à l'aide d'une passerelle Internet) dans le Guide de l'utilisateur Amazon VPC.

Lorsque vous créez un cluster de bases de données dans un VPC, vous pouvez choisir un groupe de sous-réseau de base de données. Amazon Aurora choisit dans ce sous-réseau un sous-réseau et une adresse IP à associer à votre cluster de bases de données. Si aucun groupe de sous-réseau de base de données n'existe, Amazon Aurora crée un groupe de sous-réseau par défaut lorsque vous créez un cluster de bases de données. Amazon Aurora crée une interface réseau Elastic pour votre cluster de bases de données, et l'associe à cette adresse IP. Le cluster de bases de données utilise la zone de disponibilité contenant le sous-réseau.

Dans cette étape, vous créez un groupe de sous-réseaux de base de données et ajoutez les sous-réseaux que vous avez créés pour votre VPC.

Pour créer un groupe de sous-réseaux de base de données

1. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, choisissez Subnet groups (Groupes de sous-réseaux).
3. Choisissez Create DB Subnet Group (Créer groupe de sous-réseaux de base de données).
4. Dans Nom, saisissez le nom de votre nouveau groupe de sous-réseaux de base de données.
5. Dans le champ Description, saisissez une description de votre groupe de sous-réseaux de base de données.
6. Pour le champ VPC, choisissez le VPC par défaut ou le VPC que vous avez créé.
7. Dans la section Ajouter des sous-réseaux, choisissez les zones de disponibilité qui incluent les sous-réseaux à partir de Zones de disponibilité, puis choisissez les sous-réseaux à partir de Sous-réseaux.


RDS > Subnet groups > Create DB subnet group
Create DB Subnet Group
To create a new subnet group, give it a name and a description, and choose an existing VPC. You will then be able to add subnets related to that VPC.
Subnet group details
Name
You won't be able to modify the name after your subnet group has been created.
mydbsubnetgroup
Must contain from 1 to 255 characters. Alphanumeric characters, spaces, hyphens, underscores, and periods are allowed.
Description
My DB Subnet Group
VPC
Choose a VPC identifier that corresponds to the subnets you want to use for your DB subnet group. You won't be able to choose a different VPC identifier after your subnet group has been created.
tutorial-vpc (vpc-068fe388385afc014)
Add subnets
Availability Zones
Choose the Availability Zones that include the subnets you want to add.
Choose an availability zone
us-east-1a
us-east-1c
Subnets
Choose the subnets that you want to add. The list includes the subnets in the selected Availability Zones.
Select subnets
subnet-079bd4b8953aee1dd (10.0.0.0/24)
subnet-057e85b72c46fdd9a (10.0.1.0/24)
Subnets selected (2)
Availability zone
Subnet ID
CIDR block
us-east-1a
subnet-079bd4b8953aee1dd
10.0.0.0/24
us-east-1c
subnet-057e85b72c46fdd9a
10.0.1.0/24
Cancel
Create

8. Sélectionnez Créer.

   Votre nouveau groupe de sous-réseaux DB apparaît dans la liste des groupes de sous-réseaux sur la console RDS. Vous pouvez choisir le groupe de sous-réseaux DB pour afficher les détails, y compris l'ensemble des sous-réseaux associés au groupe, dans le volet des détails en bas de la fenêtre.

Étape 3 : créer un groupe de sécurité VPC

Avant de créer votre cluster de bases de données, vous pouvez créer un groupe de sécurité VPC à associer à votre cluster de bases de données. Si vous ne créez pas de groupe de sécurité VPC, vous pouvez utiliser le groupe de sécurité par défaut lorsque vous créez un cluster de bases de données. Pour obtenir des instructions sur la création d'un groupe de sécurité pour votre cluster de bases de données, consultez Créer un groupe de sécurité VPC pour un cluster de bases de données privé(e), ou Control traffic to resources using security groups (Contrôler le trafic vers les ressources à l'aide de groupes de sécurité) dans le Guide de l'utilisateur Amazon VPC.

Étape 4 : créer une instance de base de données dans le VPC

Dans cette étape, vous créez un cluster de bases de données et utilisez le nom du VPC, le groupe de sous-réseaux de base de données et le groupe de sécurité VPC que vous avez créés dans les étapes précédentes.

> **Note**
>
> Si vous voulez que votre cluster de bases de données du VPC soit publiquement accessible, vous devez activer les attributs du VPC DNS hostnames (Noms d'hôte DNS) et DNS resolution (Résolution DNS). Pour plus d'informations, consultez DNS attributes for your VPC (Attributs DNS pour votre VPC) dans le Guide de l'utilisateur d'Amazon VPC.

Pour plus d'informations sur la création d'un cluster de bases de données, consultez Création d'un cluster de bases de données Amazon Aurora.

Lorsque vous y êtes invité dans la section Connectivity (Connectivité), saisissez le nom du VPC, le groupe de sous-réseaux de base de données et le groupe de sécurité VPC.

> **Note**
>
> La mise à jour VPCs n'est actuellement pas prise en charge pour les clusters de base de données Aurora.

Scénarios d’accès à un cluster de bases de données d’un VPC

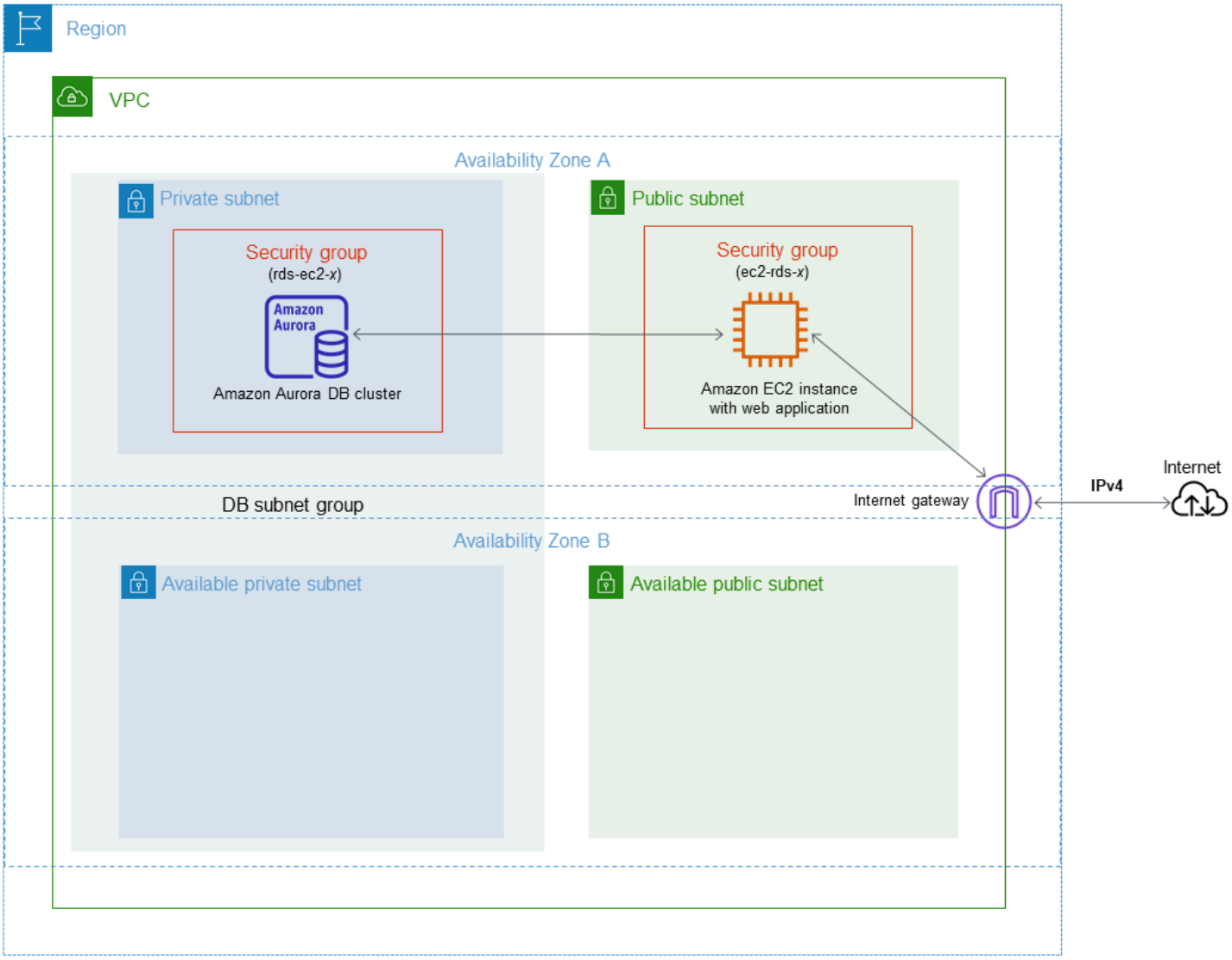
Amazon Aurora prend en charge les scénarios suivants pour accéder à un cluster de bases de données dans un VPC :

- Une instance Amazon EC2 dans le même VPC
- Une instance EC2 d’un autre VPC
- Une application cliente via Internet
- Un réseau privé

Un cluster de bases de données dans un VPC auquel accède une instance EC2 dans le même VPC

Une utilisation courante d’un cluster de bases de données d’un VPC consiste à partager les données avec un serveur d’application qui s’exécute dans une instance Amazon EC2 du même VPC.

Le schéma suivant illustre ce scénario.



La solution la plus simple pour gérer l'accès entre les instances EC2 et les clusters de bases de données du même VPC consiste à agir ainsi :

- Créez un groupe de sécurité VPC dans lequel seront placées vos clusters de base de données. Ce groupe de sécurité peut être utilisé pour restreindre l'accès aux clusters de bases de données. Par exemple, vous pouvez créer une règle personnalisée pour ce groupe de sécurité. Cela peut permettre un accès TCP en utilisant le port que vous avez attribué au cluster de la base de données lorsque vous l'avez créé et une adresse IP que vous utilisez pour accéder au cluster de la base de données à des fins de développement ou autres.
- Créez un groupe de sécurité VPC dans lequel seront placées vos instances EC2 (serveurs web et clients). Ce groupe de sécurité peut, si nécessaire, autoriser l'accès à l'instance EC2 à partir

d'Internet à l'aide de la table de routage du VPC. Par exemple, vous pouvez définir des règles sur ce groupe de sécurité pour autoriser l'accès TCP à l'instance EC2 sur le port 22.

- Créez des règles personnalisées dans le groupe de sécurité pour vos clusters de base de données qui autorisent les connexions depuis le groupe de sécurité que vous avez créé pour vos instances EC2. Ces règles peuvent permettre à tout membre du groupe de sécurité d'accéder aux clusters de la base de données.

Il existe un sous-réseau public et privé supplémentaire dans une zone de disponibilité distincte. Un groupe de sous-réseaux de base de données RDS nécessite un sous-réseau dans au moins deux zones de disponibilité. Le sous-réseau supplémentaire permet de passer facilement à un déploiement d'instance de base de données Multi-AZ à l'avenir.

Pour obtenir un didacticiel qui explique comment créer un VPC avec des sous-réseaux publics et privés pour ce scénario, consultez Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement).

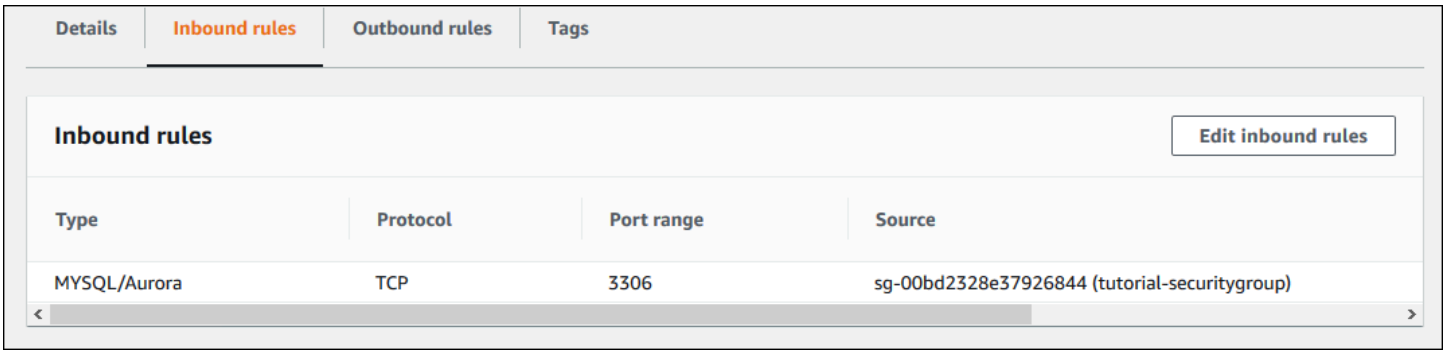
> **Tip**
>
> Vous pouvez configurer la connectivité réseau entre une instance Amazon EC2 et un cluster de base de données automatiquement lorsque vous créez le cluster de base de données. Pour plus d'informations, consultez Configurer la connectivité réseau automatique avec une instance EC2.

Pour créer une règle dans un groupe de sécurité VPC qui autorise les connexions à partir d'un autre groupe de sécurité, procédez comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc.
2. Dans le panneau de navigation, choisissez Groupes de sécurité.
3. Choisissez ou créez un groupe de sécurité auquel vous voulez autoriser les membres d'un autre groupe de sécurité à accéder. Dans le scénario précédent, il s'agit du groupe de sécurité que vous utilisez pour vos clusters de bases de données. Sélectionnez l'onglet Inbound Rules (Règles entrantes), puis Edit inbound rules (Modifier les règles entrantes).
4. Sur la page Edit inbound rules (Modifier les règles entrantes), cliquez sur Add Rule (Ajouter une règle).

5. Pour Type, choisissez l'entrée qui correspond au port que vous avez utilisé lorsque vous avez créé votre cluster de bases de données, par exemple MYSQL/Aurora.
6. Dans la zone Source, commencez à taper l'ID du groupe de sécurité, qui répertorie les groupes de sécurité correspondants. Choisissez le groupe de sécurité dont vous voulez autoriser les membres à accéder aux ressources protégées par ce groupe de sécurité. Dans le scénario précédent, il s'agit du groupe de sécurité que vous utilisez pour votre instance EC2.
7. Si nécessaire, répétez les étapes pour le protocole TCP en créant une règle avec Tous TCP comme Type et votre groupe de sécurité dans la zone Source. Si vous prévoyez d'utiliser le protocole UDP, créez une règle avec All UDP (Tous UDP) comme Type et votre groupe de sécurité dans Source.
8. Sélectionnez Enregistrer les règles.

L'écran suivant affiche une règle entrante, ainsi qu'un groupe de sécurité pour sa source.

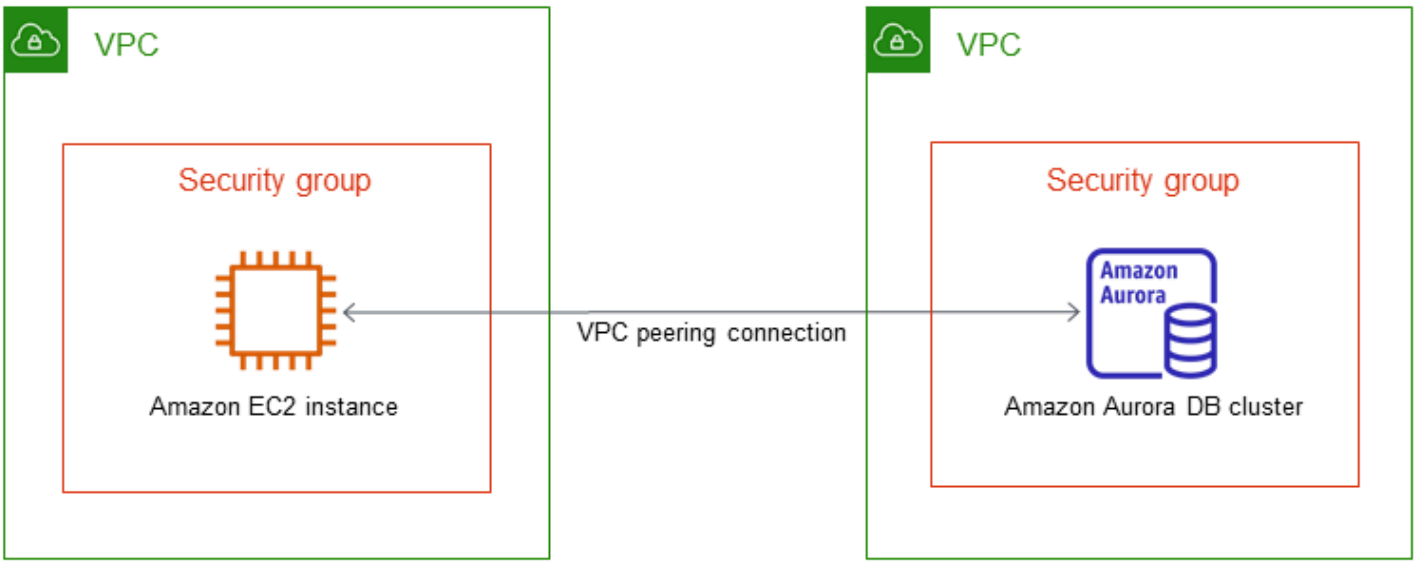


Pour plus d'informations sur la connexion à votre cluster de bases de données depuis votre instance EC2, consultez Connexion à un cluster de bases de données Amazon Aurora.

Un cluster de bases de données d'un VPC accessible par une instance EC2 d'un autre VPC

Quand vos clusters de bases de données se trouvent dans un VPC différent de l'instance EC2 que vous utilisez pour y accéder, vous pouvez utiliser l'appairage de VPC pour accéder au cluster de bases de données.

Le schéma suivant illustre ce scénario.

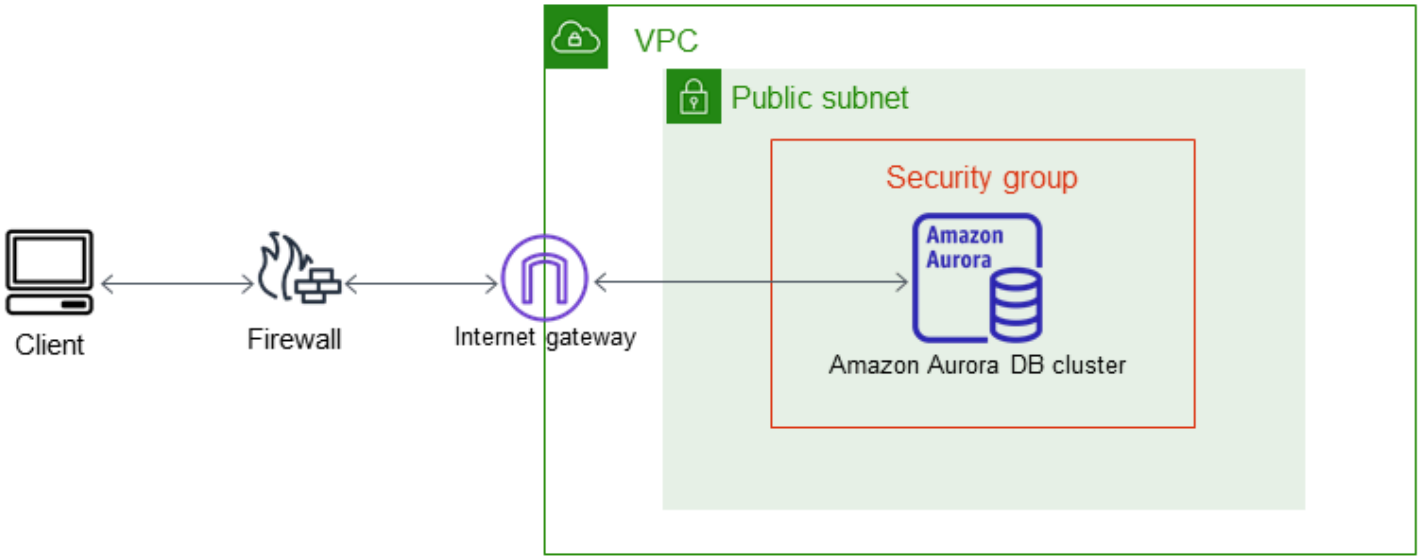


Une connexion d'appairage de VPC est une connexion de mise en réseau entre deux VPC qui permet de router le trafic entre ces derniers à l'aide d'adresses IP privées. Les ressources des deux VPC peuvent communiquer entre elles comme si elles se trouvaient dans le même réseau. Vous pouvez créer une connexion d'appairage de VPC entre vos propres VPC, avec un VPC situé dans un autre compte AWS ou avec un VPC au sein d'une autre Région AWS. Pour plus d'informations sur l'appairage de VPC, consultez Appairage de VPC dans le Guide de l'utilisateur Amazon Virtual Private Cloud.

Un cluster de bases de données d'un VPC accessible par une application cliente via Internet

Pour accéder à des clusters de bases de données d'un VPC à partir d'une application cliente via Internet, vous configurez un VPC avec un seul sous-réseau public et une passerelle Internet pour activer la communication sur Internet.

Le schéma suivant illustre ce scénario.



Nous recommandons la configuration suivante :

- Un VPC de taille /16 (par exemple, CIDR : 10.0.0.0/16). Cette taille fournit 65 536 adresses IP privées.
- Un sous-réseau de taille /24 (par exemple, CIDR : 10.0.0.0/24). Cette taille fournit 256 adresses IP privées.
- Un(e) cluster de bases de données Amazon Aurora qui est associé(e) au VPC et au sous-réseau. Amazon RDS affecte à votre cluster de bases de données une adresse IP au sein du sous-réseau.
- Une passerelle Internet qui connecte le VPC à Internet et à d'autres produits AWS.
- Groupe de sécurité associé au cluster de bases de données. Les règles de trafic entrant de votre groupe de sécurité permettent à votre application client d'accéder à votre cluster de bases de données.

Pour plus d'informations sur la création de clusters de bases de données dans un VPC, consultez Création d'un cluster de bases de données dans un VPC.

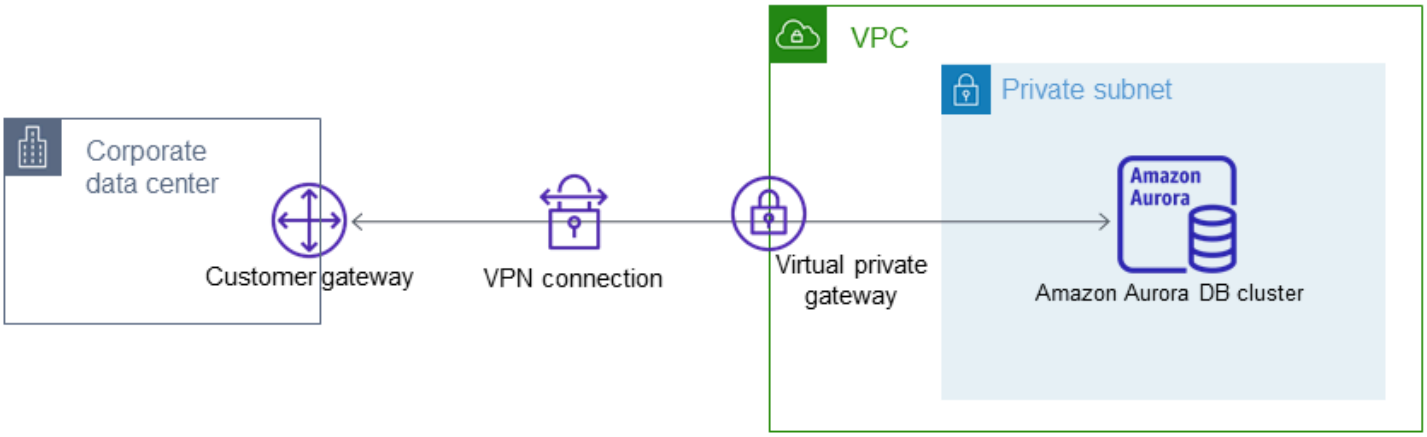
Un cluster de bases de données d'un VPC accessible par un réseau privé

Si votre cluster de bases de données n'est pas accessible publiquement, les options suivantes vous permettent d'y accéder à partir d'un réseau privé :

- Une connexion AWS Site-to-Site VPN. Pour plus d'informations, consultez Qu'est-ce qu'AWS Site-to-Site VPN ?

- Une connexion Direct Connect. Pour plus d'informations, consultez Qu'est-ce qu'Direct Connect ?
- Une connexion AWS Client VPN. Pour plus d'informations, consultez Qu'est-ce qu'AWS Client VPN ?

Le diagramme suivant illustre un scénario avec une connexion AWS Site-to-Site VPN.

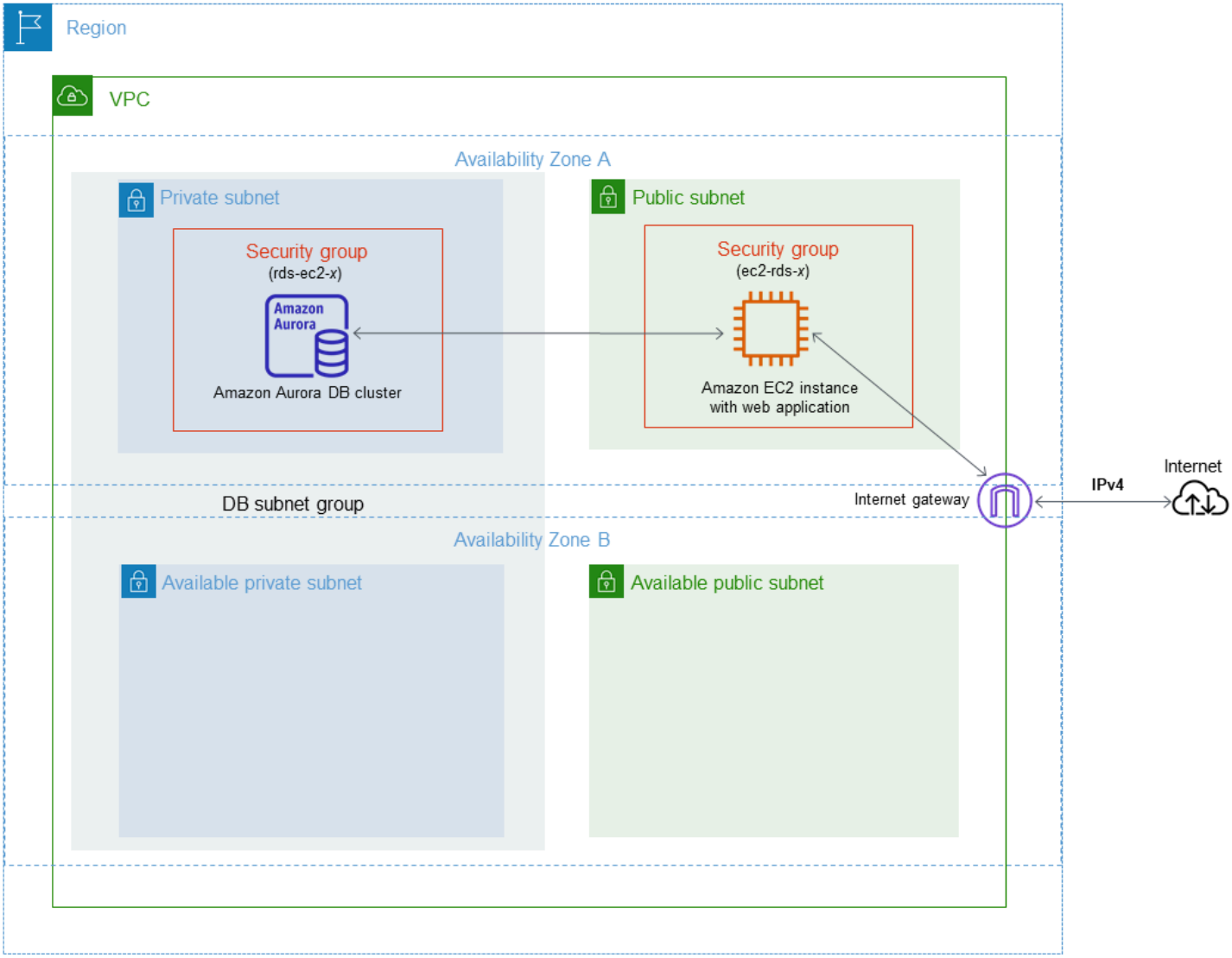


Pour plus d'informations, consultez Confidentialité du trafic inter-réseau.

Tutoriel : créer un VPC à utiliser avec un cluster de bases de données (IPv4 uniquement)

Un scénario courant comprend un cluster de bases de données dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC. Ce VPC partage des données avec un serveur web qui fonctionne dans le même VPC. Dans ce didacticiel, vous créez le VPC pour ce scénario.

Le schéma suivant illustre ce scénario. Pour plus d'informations sur d'autres scénarios, consultez Scénarios d'accès à un cluster de bases de données d'un VPC.



Votre cluster de bases de données doit être disponible uniquement pour votre serveur Web, et non pour l'Internet public. Vous créez ainsi un VPC avec des sous-réseaux publics et privés. Le serveur web étant hébergé dans le sous-réseau public, il peut atteindre Internet. Le cluster de bases de

données est hébergé(e) dans un sous-réseau privé. Le serveur web peut se connecter au cluster de bases de données, car il est hébergé dans le même VPC. Mais le cluster de bases de données n'est pas accessible à l'Internet public, ce qui assure une plus grande sécurité.

Ce tutoriel configure un sous-réseau public et privé supplémentaire dans une zone de disponibilité séparée. Ces sous-réseaux ne sont pas utilisés par le tutoriel. Un groupe de sous-réseaux de base de données RDS nécessite un sous-réseau dans au moins deux zones de disponibilité. Le sous-réseau supplémentaire facilite la configuration de plus d'une instance de base de données Aurora

Ce didacticiel décrit la configuration d'un VPC pour les clusters de bases de données Amazon Aurora. Pour obtenir un didacticiel qui vous montre comment créer un serveur Web pour ce scénario VPC, consultez Didacticiel : Créer un serveur web et une cluster de base de données Amazon Aurora. Pour plus d'informations sur Amazon VPC, consultez le Guide de mise en route Amazon VPC et le Guide de l'utilisateur Amazon VPC.

> **Tip**
>
> Vous pouvez configurer la connectivité réseau entre une instance Amazon EC2 et un cluster de base de données automatiquement lorsque vous créez le cluster de base de données. La configuration du réseau est similaire à celle décrite dans ce tutoriel. Pour plus d'informations, consultez Configurer la connectivité réseau automatique avec une instance EC2.

Créer un VPC avec des sous-réseaux publics et privés

Utilisez la procédure suivante pour créer un VPC avec des sous-réseaux publics et privés.

Pour créer un VPC et des sous-réseaux

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Dans le coin supérieur droit d'AWS Management Console, choisissez la région où vous voulez créer le VPC. Cet exemple utilise la région USA Ouest (Oregon).
3. Dans le coin supérieur gauche, choisissez VPC Dashboard (Tableau de bord VPC). Pour commencer à créer un VPC, sélectionnez Create VPC (Créer un VPC).
4. Pour Resources to create (Ressources à créer) sous VPC settings (Paramètres VPC), choisissez VPC and more (VPC et plus).
5. Pour VPC settings (Paramètres de VPC), définissez les valeurs suivantes :

- Name tag auto-generation (Génération automatique de balise de nom) : **`tutorial`**
- Bloc CIDR IPv4 : **`10.0.0.0/16`**
- Bloc CIDR IPv6 : Pas de bloc CIDR IPv6
- Tenancy (Location) : Default (Par défaut)
- Number of Availability Zones (AZs) [Nombre de zones de disponibilité (AZ)] : 2
- Customize AZs (Personnaliser les AZ) : conserver les valeurs par défaut.
- Number of public subnet (Nombre de sous-réseaux publics) : 2
- Number of private subnets (Nombre de sous-réseaux privés) : 2
- Customize subnets CIDR blocks (Personnaliser les blocs CIDR des sous-réseaux) : conserver les valeurs par défaut.
- NAT gateways ($) [Passerelles NAT ($)] : None (Aucune)
- VPC endpoints (Points de terminaison VPC) : None (Aucun)
- DNS options (Options DNS) : conservez les valeurs par défaut.

6. Sélectionnez Create VPC (Créer un VPC).

Créer un groupe de sécurité VPC pour un serveur web public

Ensuite, vous créez un groupe de sécurité pour l'accès public. Pour vous connecter aux instances EC2 publiques dans votre VPC, ajoutez des règles entrantes au groupe de sécurité de votre VPC. Elles permettent au trafic de se connecter depuis Internet.

Pour créer un groupe de sécurité VPC

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Choisissez successivement VPC Dashboard (Tableau de bord VPC), Security Groups (Groupes de sécurité) et Create Security Group (Créer un groupe de sécurité).
3. Sur la page Create Security Group (Créer un groupe de sécurité), définissez les valeurs suivantes :

   - Nom du groupe de sécurité : **`tutorial-securitygroup`**
   - Description : **`Tutorial Security Group`**
   - VPC : choisissez le VPC que vous avez créé précédemment, par exemple : vpc-***`identifier`*** (tutorial-vpc)

4. Ajoutez des règles entrantes au groupe de sécurité.

   a. Déterminez l'adresse IP à utiliser pour vous connecter aux instances EC2 de votre VPC à l'aide de Secure Shell (SSH). Pour déterminer votre adresse IP publique, dans une fenêtre ou un onglet de navigateur différent, vous pouvez utiliser le service à l'adresse https://checkip.amazonaws.com. Exemple d'adresse IP : `203.0.113.25/32`.

      Dans de nombreux cas, votre connexion s'effectue via un fournisseur de services Internet (FSI) ou derrière votre pare-feu sans adresse IP statique. Dans ce cas, trouvez la plage d'adresses IP utilisées par les ordinateurs clients.

      > **Warning**
      >
      > Si vous utilisez `0.0.0.0/0` pour l'accès SSH, vous permettez à toutes les adresses IP d'accéder à vos instances publiques par SSH. Cette approche est acceptable pour une brève durée dans un environnement de test, mais n'est pas sécurisée pour les environnements de production. Dans un environnement de production, autorisez uniquement l'accès à vos instances à l'aide de SSH pour une adresse IP ou une plage d'adresses spécifique.

   b. Dans la section Règles entrantes, choisissez Ajouter une règle.

   c. Définissez les valeurs suivantes pour que votre nouvelle règle entrante autorise l'accès SSH à votre instance Amazon EC2. Pour ce faire, vous pouvez vous connecter à votre instance Amazon EC2 pour installer le serveur web et d'autres utilitaires. Vous allez également vous connecter à votre instance EC2 afin de charger le contenu de votre serveur Web.

      - Type : **`SSH`**
      - Source : l'adresse IP ou la plage d'adresses IP de l'étape a ; par exemple : **`203.0.113.25/32`**.

   d. Choisissez Ajouter une règle.

   e. Définissez les valeurs suivantes pour que votre nouvelle règle entrante autorise HTTP à accéder à votre serveur Web :

      - Type : **`HTTP`**
      - Source : **`0.0.0.0/0`**

5. Choisissez Create security group (Créer un groupe de sécurité) pour créer le groupe de sécurité.

Notez l'ID du groupe de sécurité, car vous en aurez besoin ultérieurement dans ce didacticiel.

Créer un groupe de sécurité VPC pour un cluster de bases de données privé(e)

Pour que votre cluster de bases de données demeure privé(e), créez un deuxième groupe de sécurité pour l'accès privé. Pour vous connecter aux clusters de bases de données privés de votre VPC, vous devez ajouter des règles entrantes à votre groupe de sécurité VPC qui autorisent le trafic à partir de votre serveur web uniquement.

Pour créer un groupe de sécurité VPC

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Choisissez successivement VPC Dashboard (Tableau de bord VPC), Security Groups (Groupes de sécurité) et Create Security Group (Créer un groupe de sécurité).
3. Sur la page Create Security Group (Créer un groupe de sécurité), définissez les valeurs suivantes :
   - Nom du groupe de sécurité : **`tutorial-db-securitygroup`**
   - Description : **`Tutorial DB Instance Security Group`**
   - VPC : choisissez le VPC que vous avez créé précédemment, par exemple : vpc-***`identifier`*** (tutorial-vpc)
4. Ajoutez des règles entrantes au groupe de sécurité.
   a. Dans la section Règles entrantes, choisissez Ajouter une règle.
   b. Définissez les valeurs suivantes pour que votre nouvelle règle entrante autorise le trafic MySQL sur le port 3306 à partir de votre instance Amazon EC2. Dans ce cas, vous pouvez vous connecter du serveur Web à votre cluster de bases de données. Pour ce faire, vous pouvez stocker et extraire les données entre votre application web et votre base de données.
      - Type : **`MySQL/Aurora`**
      - Source : identifiant du groupe de sécurité tutorial-securitygroup que vous avez créé précédemment dans ce tutoriel, par exemple : sg-9edd5cfb.
5. Choisissez Create security group (Créer un groupe de sécurité) pour créer le groupe de sécurité.

Création d’un groupe de sous-réseaux de base de données

Un groupe de sous-réseaux de base de données désigne une collection de sous-réseaux que vous créez dans un VPC et que vous spécifiez alors pour vos clusters de bases de données. Un groupe de sous-réseaux de base de données vous permet de spécifier un VPC particulier lors de la création de clusters de bases de données.

Pour créer un groupe de sous-réseaux de base de données

1. Identifiez les sous-réseaux privés pour votre base de données dans le VPC.

   a. Ouvrez la console Amazon VPC à l’adresse https://console.aws.amazon.com/vpc/.

   b. Choisissez VPC Dashboard (Tableau de bord du VPC), puis Subnets (Sous-réseaux).

   c. Notez les ID de sous-réseau des sous-réseaux nommés tutorial-subnet-private1-us-west-2a et tutorial-subnet-private2-us-west-2b.

      Vous avez besoin des ID de sous-réseau lorsque vous créez votre groupe de sous-réseau de base de données.

2. Ouvrez la console Amazon RDS à l’adresse https://console.aws.amazon.com/rds/.

   Assurez-vous de vous connecter à la console Amazon RDS et non à la console Amazon VPC.

3. Dans le panneau de navigation, choisissez Subnet groups (Groupes de sous-réseaux).

4. Choisissez Create DB Subnet Group (Créer groupe de sous-réseaux de base de données).

5. Sur la page Create DB subnet group (Créer groupe de sous-réseaux de base de données), définissez ces valeurs dans Subnet group details (Détails de groupe de sous-réseaux) :

   - Nom: **`tutorial-db-subnet-group`**
   - Description : **`Tutorial DB Subnet Group`**
   - VPC : tutorial-vpc (vpc-***`identifier`***)

6. Dans la section Ajouter des sous-réseaux, choisissez les zones de disponibilité et les sous-réseaux.

   Pour ce tutoriel, choisissez us-west-2a et us-west-2b pour les Availability Zones (Zones de disponibilité). Pour Subnets (Sous-réseaux), choisissez les sous-réseaux privés que vous avez identifiés à l’étape précédente.

7. Choisissez Créer.

Votre nouveau groupe de sous-réseaux DB apparaît dans la liste des groupes de sous-réseaux sur la console RDS. Vous pouvez choisir le groupe de sous-réseaux DB pour afficher les détails dans le volet des détails en bas de la fenêtre. Ces détails comprennent tous les sous-réseaux employés par le groupe.

> **Note**
>
> Si vous avez créé ce VPC pour effectuer Didacticiel : Créer un serveur web et une cluster de base de données Amazon Aurora, créez le cluster de bases de données en suivant les instructions fournies dans Créer un cluster de bases de données Amazon Aurora.

Suppression du VPC

Après avoir créé le VPC et d'autres ressources pour ce didacticiel, vous pouvez les supprimer si elles ne sont plus nécessaires.

> **Note**
>
> Si vous avez ajouté des ressources dans le VPC que vous avez créé pour ce tutoriel, vous devrez peut-être les supprimer avant de pouvoir supprimer le VPC. Par exemple, ces ressources peuvent comprendre des instances Amazon EC2 ou des clusters de bases de données Amazon RDS. Pour plus d'informations, consultez Supprimer votre VPC dans le Guide de l'utilisateur Amazon VPC.

Pour supprimer un VPC et les ressources associées

1. Supprimez le groupe de sous-réseaux de base de données.
   a. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
   b. Dans le panneau de navigation, choisissez Subnet groups (Groupes de sous-réseaux).
   c. Sélectionnez le groupe de sous-réseaux de base de données que vous voulez supprimer, par exemple tutorial-db-subnet-group.
   d. Choisissez Supprimer, puis Supprimer dans la fenêtre de confirmation.
2. Notez l'ID du VPC.

a. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
b. Choisissez Tableau de bord du VPC, puis VPC.
c. Dans la liste, identifiez le VPC que vous avez créé, tel que tutorial-vpc.
d. Notez le VPC ID (ID de VPC) du VPC que vous avez créé. Vous aurez besoin de l'ID de VPC dans les étapes suivantes.

3. Suppression du groupe de sécurité

a. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
b. Choisissez Tableau de bord du VPC, puis Groupes de sécurité.
c. Sélectionnez le groupe de sécurité pour l'instance de base de données Amazon RDS, par exemple tutorial-db-securitygroup.
d. Pour Actions, choisissez Delete security groups (Supprimer des groupes de sécurité), puis Delete (Supprimer) sur la page de confirmation.
e. Sur la page Groupes de sécurité, sélectionnez le groupe de sécurité pour l'instance Amazon EC2, par exemple tutorial-securitygroup.
f. Pour Actions, choisissez Delete security groups (Supprimer des groupes de sécurité), puis Delete (Supprimer) sur la page de confirmation.

4. Supprimer le VPC.

a. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
b. Choisissez Tableau de bord du VPC, puis VPC.
c. Sélectionnez le VPC que vous voulez supprimer, tel que tutorial-vpc.
d. Pour Actions, choisissez Supprimer le VPC.

   La page de confirmation affiche les autres ressources associées au VPC qui seront également supprimées, y compris les sous-réseaux qui lui sont associés.
e. Sur la page de confirmation, entrez **`delete`** et choisissez Supprimer.

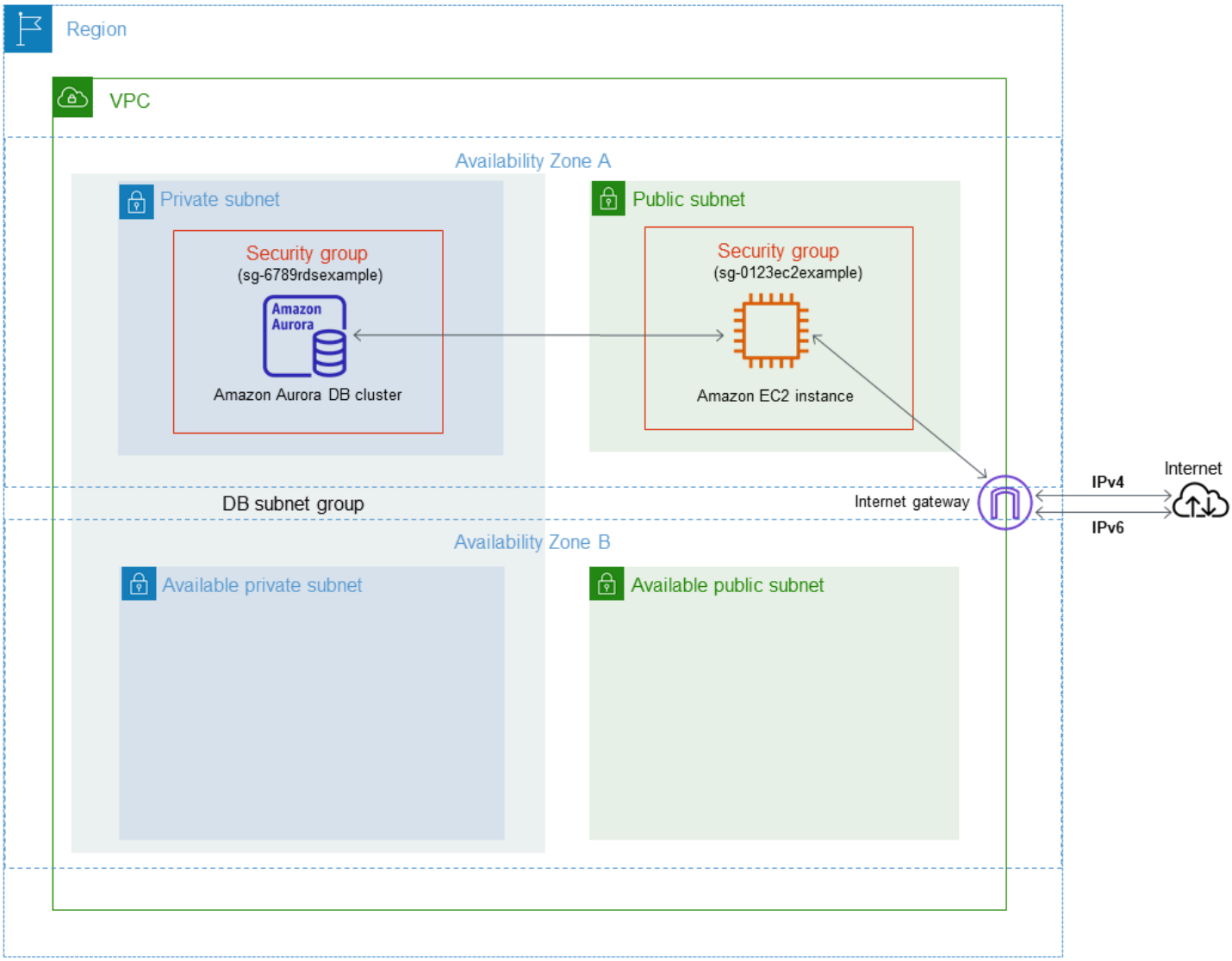
Tutoriel : Créer un VPC à utiliser avec un cluster de bases de données (mode double-pile)

Un scénario courant comprend un cluster de bases de données dans un cloud privé virtuel (VPC) basé sur le service Amazon VPC. Ce VPC partage des données avec une instance publique Amazon EC2 qui fonctionne dans le même VPC.

Dans ce tutoriel, vous créez le VPC pour ce scénario qui fonctionne avec une base de données en mode double pile. Le mode double pile active la connexion via le protocole d'adressage IPv6. Pour plus d'informations sur les adresses IP, consultez Adressage IP Amazon Aurora.

Les clusters de réseau à double pile sont pris en charge dans la plupart des régions. Pour plus d'informations, consultez Disponibilité de clusters de bases de données en réseau à double pile. Pour connaître les limites du mode à double pile, consultez Limitations pour les clusters de base de données en réseau à double pile.

Le schéma suivant illustre ce scénario.



Pour plus d'informations sur d'autres scénarios, consultez Scénarios d'accès à un cluster de bases de données d'un VPC.

Votre cluster de bases de données doit être disponible uniquement pour votre instance Amazon EC2, et non pour l'Internet public. Vous créez ainsi un VPC avec des sous-réseaux publics et privés. L'instance Amazon EC2 est hébergée dans le sous-réseau public, de sorte qu'elle peut accéder à l'Internet public. Le cluster de bases de données est hébergé(e) dans un sous-réseau privé. L'instance Amazon EC2 peut se connecter au cluster de bases de données, car elle est hébergée dans le même VPC. Cependant, le cluster de bases de données n'est pas accessible à l'Internet public, ce qui assure une plus grande sécurité.

Ce tutoriel configure un sous-réseau public et privé supplémentaire dans une zone de disponibilité séparée. Ces sous-réseaux ne sont pas utilisés par le tutoriel. Un groupe de sous-réseaux de base

de données RDS nécessite un sous-réseau dans au moins deux zones de disponibilité. Le sous-réseau supplémentaire permet de configurer facilement plus d'une instance de base de données Aurora.

Pour créer un cluster de bases de données qui utilise le mode double pile, spécifiez Dual-stack mode (mode double pile) pour le paramètre Network type (Type de réseau). Vous pouvez également modifier un cluster de bases de données avec le même paramètre. Pour plus d'informations sur la création d'un cluster de bases de données, consultez Création d'un cluster de bases de données Amazon Aurora. Pour plus d'informations sur la modification d'un cluster, consultez Modification d'un cluster de bases de données Amazon Aurora.

Ce didacticiel décrit la configuration d'un VPC pour les clusters de bases de données Amazon Aurora. Pour en savoir plus sur Amazon VPC, consultez le Guide de l'utilisateur Amazon VPC.

Créer un VPC avec des sous-réseaux publics et privés

Utilisez la procédure suivante pour créer un VPC avec des sous-réseaux publics et privés.

Pour créer un VPC et des sous-réseaux

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Dans le coin supérieur droit de la AWS Management Console, choisissez la région dans laquelle vous souhaitez créer votre VPC. L'exemple utilise la région USA Est (Ohio).
3. Dans le coin supérieur gauche, choisissez VPC Dashboard (Tableau de bord VPC). Pour commencer à créer un VPC, sélectionnez Create VPC (Créer un VPC).
4. Pour Resources to create (Ressources à créer) sous VPC settings (Paramètres VPC), choisissez VPC and more (VPC et plus).
5. Pour les valeurs VPC settings (Paramètres VPC) restantes, définissez ce qui suit :

   - Name tag auto-generation (Génération automatique de balise de nom) : **`tutorial-dual-stack`**
   - Bloc CIDR IPv4 : **`10.0.0.0/16`**
   - Bloc CIDR IPv6 : Bloc CIDR IPv6 fourni par Amazon
   - Tenancy (Location) : Default (Par défaut)
   - Number of Availability Zones (AZs) [Nombre de zones de disponibilité (AZ)] : 2
   - Customize AZs (Personnaliser les AZ) : conserver les valeurs par défaut.
   - Number of public subnet (Nombre de sous-réseaux publics) : 2

- Number of private subnets (Nombre de sous-réseaux privés) : 2
- Customize subnets CIDR blocks (Personnaliser les blocs CIDR des sous-réseaux) : conserver les valeurs par défaut.
- NAT gateways ($) [Passerelles NAT ($)] : None (Aucune)
- Egress only internet gateway (Passerelle Internet de sortie uniquement) : No (Non)
- VPC endpoints (Points de terminaison VPC) : None (Aucun)
- DNS options (Options DNS) : conservez les valeurs par défaut.

> **Note**
>
> Amazon RDS nécessite au moins deux sous-réseaux dans deux zones de disponibilité différentes pour prendre en charge les déploiements d'instances de base de données Multi-AZ. Ce tutoriel crée un déploiement Mono-AZ, mais l'exigence permet de le convertir facilement en un déploiement d'instance de base de données Multi-AZ à l'avenir.

6. Sélectionnez Create VPC (Créer un VPC).

Créer un groupe de sécurité VPC pour une instance publique Amazon EC2

Ensuite, vous créez un groupe de sécurité pour l'accès public. Pour vous connecter aux instance EC2 publiques de votre VPC, ajoutez des règles entrantes à votre groupe de sécurité VPC qui autorisent le trafic à se connecter depuis Internet.

Pour créer un groupe de sécurité VPC

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Choisissez successivement VPC Dashboard (Tableau de bord VPC), Security Groups (Groupes de sécurité) et Create Security Group (Créer un groupe de sécurité).
3. Sur la page Create Security Group (Créer un groupe de sécurité), définissez les valeurs suivantes :
   - Nom du groupe de sécurité : **`tutorial-dual-stack-securitygroup`**
   - Description : **`Tutorial Dual-Stack Security Group`**

- VPC : choisissez le VPC que vous avez créé précédemment, par exemple : vpc-***identifier*** (tutorial-dual-stack-vpc)

4. Ajoutez des règles entrantes au groupe de sécurité.

   a. Déterminez l'adresse IP à utiliser pour vous connecter aux instances EC2 de votre VPC à l'aide de Secure Shell (SSH).

      `203.0.113.25/32` est un exemple d'adresse IPv4 (Internet Protocol version 4). `2001:db8:1234:1a00::/64` est un exemple de plage d'adresses IPv6 (Internet Protocol version 6).

      Dans de nombreux cas, votre connexion s'effectue via un fournisseur de services Internet (FSI) ou derrière votre pare-feu sans adresse IP statique. Dans ce cas, trouvez la plage d'adresses IP utilisées par les ordinateurs clients.

      > ⚠ Warning
      >
      > Si vous utilisez `0.0.0.0/0` pour IPv4 ou `::0` pour IPv6, vous permettez à toutes les adresses IP d'accéder à vos instances publiques à l'aide de SSH. Cette approche est acceptable pour une brève durée dans un environnement de test, mais n'est pas sécurisée pour les environnements de production. Dans un environnement de production, autorisez uniquement une adresse IP ou une plage d'adresses IP spécifiques à accéder à vos instances.

   b. Dans la section Règles entrantes, choisissez Ajouter une règle.

   c. Définissez les valeurs suivantes pour que votre nouvelle règle entrante autorise l'accès Secure Shell (SSH) à votre instance Amazon EC2. Si vous faites cela, vous pouvez vous connecter à votre instance EC2 pour installer des clients SQL et d'autres applications. Indiquez une adresse IP pour accéder à votre instance EC2 :

      - Type : **`SSH`**
      - Source : l'adresse IP ou la plage de l'étape a. Exemple d'adresse IP IPv4 : **`203.0.113.25/32`**. Exemple d'adresse IP IPv6 : **`2001:DB8::/32`**.

5. Choisissez Create security group (Créer un groupe de sécurité) pour créer le groupe de sécurité.

   Notez l'ID du groupe de sécurité, car vous en aurez besoin ultérieurement dans ce didacticiel.

Créer un groupe de sécurité VPC pour un cluster de bases de données privé(e)

Pour que votre cluster de bases de données demeure privé(e), créez un deuxième groupe de sécurité pour l'accès privé. Pour vous connecter aux clusters de bases de données privé(e)s de votre VPC, ajoutez des règles entrantes à votre groupe de sécurité VPC. Elles autorisent le trafic provenant de votre instance Amazon EC2 uniquement.

Pour créer un groupe de sécurité VPC

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Choisissez successivement VPC Dashboard (Tableau de bord VPC), Security Groups (Groupes de sécurité) et Create Security Group (Créer un groupe de sécurité).
3. Sur la page Create Security Group (Créer un groupe de sécurité), définissez les valeurs suivantes :

   - Nom du groupe de sécurité : **`tutorial-dual-stack-db-securitygroup`**
   - Description : **`Tutorial Dual-Stack DB Instance Security Group`**
   - VPC : choisissez le VPC que vous avez créé précédemment, par exemple : vpc-***`identifier`*** (tutorial-dual-stack-vpc)
4. Ajoutez des règles entrantes au groupe de sécurité :

   a. Dans la section Règles entrantes, choisissez Ajouter une règle.

   b. Définissez les valeurs suivantes pour que votre nouvelle règle entrante autorise le trafic MySQL sur le port 3306 à partir de votre instance Amazon EC2. Dans ce cas, vous pouvez vous connecter de votre instance EC2 à votre cluster de bases de données. Cela signifie que vous pouvez envoyer des données de votre instance EC2 vers votre base de données.

      - Type : MySQL/Aurora
      - Source : identifiant du groupe de sécurité tutorial-dual-stack-securitygroup que vous avez créé précédemment dans ce tutoriel, par exemple : sg-9edd5cfb.
5. Pour créer le groupe de sécurité, choisissez Créer un groupe de sécurité.

Création d'un groupe de sous-réseaux de base de données

Un groupe de sous-réseaux de base de données désigne une collection de sous-réseaux que vous créez dans un VPC et que vous spécifiez alors pour vos clusters de bases de données. En utilisant un groupe de sous-réseau de base de données, vous pouvez spécifier un VPC particulier lors de

la création de clusters de bases de données. Pour créer un groupe de sous-réseaux de la base de données qui soit compatible DUAL, tous les sous-réseaux doivent être compatibles DUAL. Pour être compatible DUAL, un sous-réseau doit être associé à un CIDR IPv6.

Pour créer un groupe de sous-réseaux de base de données

1. Identifiez les sous-réseaux privés pour votre base de données dans le VPC.

   a. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.

   b. Choisissez VPC Dashboard (Tableau de bord du VPC), puis Subnets (Sous-réseaux).

   c. Notez les ID des sous-réseaux nommés tutorial-dual-stack-subnet-private1-us-west-2a et tutorial-dual-stack-subnet-private2-us-west-2b.

      Vous aurez besoin des ID de sous-réseau lorsque vous créerez votre groupe de sous-réseau de base de données.

2. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

   Assurez-vous de vous connecter à la console Amazon RDS et non à la console Amazon VPC.

3. Dans le panneau de navigation, choisissez Subnet groups (Groupes de sous-réseaux).

4. Choisissez Create DB Subnet Group (Créer groupe de sous-réseaux de base de données).

5. Sur la page Create DB subnet group (Créer groupe de sous-réseaux de base de données), définissez ces valeurs dans Subnet group details (Détails de groupe de sous-réseaux) :

   - Nom: **`tutorial-dual-stack-db-subnet-group`**
   - Description : **`Tutorial Dual-Stack DB Subnet Group`**
   - VPC : tutorial-dual-stack-vpc (vpc-***`identifier`***)

6. Dans la section Add subnets (Ajouter des sous-réseaux), choisissez des valeurs pour les options Availability Zones (Zones de disponibilité) et Subnets (Sous-réseaux).

   Pour ce tutoriel, choisissez us-east-2a et us-east-2b pour les Availability Zones (Zones de disponibilité). Pour Subnets (Sous-réseaux), choisissez les sous-réseaux privés que vous avez identifiés à l'étape précédente.

7. Choisissez Créer.

Votre nouveau groupe de sous-réseaux DB apparaît dans la liste des groupes de sous-réseaux sur la console RDS. Vous pouvez choisir le groupe de sous-réseau de base de données pour afficher

ses détails. Il s'agit notamment des protocoles d'adressage pris en charge, de tous les sous-réseaux associés au groupe et du type de réseau pris en charge par le groupe de sous-réseaux de base de données.

Créer une instance Amazon EC2 en mode double pile

Pour créer une instance Amazon EC2, suivez les instructions de la section Lancer une instance à l'aide du nouvel assistant de lancement d'instance du Guide de l'utilisateur Amazon EC2.

Sur la page Configure Instance Details (Configurer les détails d'instance), spécifiez les valeurs suivantes et conservez les valeurs par défaut des autres paramètres :

- Réseau – Choisissez un VPC existant avec des sous-réseaux publics et privés, tels que tutorial-dual-stack-vpc (vpc-*identifier*), créés dans Créer un VPC avec des sous-réseaux publics et privés.
- Subnet (Sous-réseau) : choisissez un sous-réseau public existant, tel que subnet-***identifier*** | tutorial-dual-stack-subnet-public1-us-east-2a | us-east-2a créé dans Créer un groupe de sécurité VPC pour une instance publique Amazon EC2.
- Auto-assign Public IP (Attribuer automatiquement l'adresse IP publique) : choisissez Enable (Activer).
- Auto-assign IPv6 IP (Affectation automatique de l'IPv6) : choisissez Enable (Activer).
- Firewall (security groups) [Pare-feu (groupes de sécurité)] : choisissez Select an existing security group (Sélectionnez un groupe de sécurité existant).
- Common security groups (Groupes de sécurité communs) : choisissez un groupe de sécurité existant, tel que le `tutorial-securitygroup` créé dans Créer un groupe de sécurité VPC pour une instance publique Amazon EC2. Assurez-vous que le groupe de sécurité que vous choisissez inclut des règles entrantes pour l'accès SSH (Secure Shell) et HTTP.

Création d'un cluster de bases de données en mode double pile

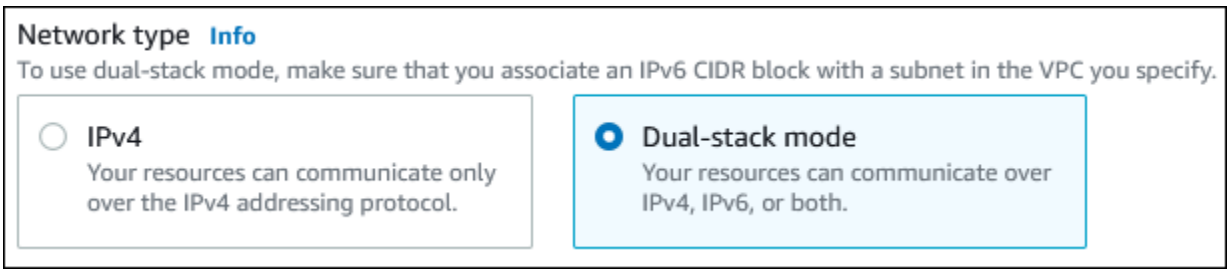
Dans cette étape, vous créez un cluster de bases de données qui fonctionne en mode double pile.

Pour créer une instance de base de données

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.

2. Dans le coin supérieur droit de la console, choisissez la Région AWS dans laquelle vous voulez créer le cluster de bases de données. L'exemple utilise la région USA Est (Ohio).
3. Dans la panneau de navigation, choisissez Bases de données.
4. Choisissez Create database (Créer une base de données).
5. Sur la page Create database (Créer une base de données), vérifiez que l'option Standard create (Création standard) est activée, puis choisissez le type de moteur de base de données Aurora MySQL.
6. Dans la section Connectivity (Connectivité), définissez les valeurs suivantes :

   - Network type (Type de réseau) – choisissez Dual-stack mode (Mode double pile)

   - Cloud privé virtuel (VPC) : choisissez un VPC existant avec des sous-réseaux publics et privés, tel que tutorial-dual-stack-vpc (vpc-*identifier*) créé dans Créer un VPC avec des sous-réseaux publics et privés.

     Le VPC doit avoir des sous-réseaux dans des zones de disponibilité différentes.

   - DB Subnet group (Groupe de sous-réseau de la base de données) : choisissez un groupe de sous-réseau de base de données pour le VPC, tel que tutorial-dual-stack-db-subnet-group créé dans Création d'un groupe de sous-réseaux de base de données.
   - Public access (Accès public) : choisissez No (Non).
   - VPC security group (firewall) [Groupe de sécurité VPC (pare-feu)] : sélectionnez Choose existing (Choisir l'existant).
   - Existing VPC security groups (Groupes de sécurité VPC existants) – choisissez un groupe de sécurité VPC existant qui est configuré pour un accès privé, tel que tutorial-dual-stack-db-securitygroup créé dans Créer un groupe de sécurité VPC pour un cluster de bases de données privé(e).

     Supprimez les autres groupes de sécurité, tels que le groupe de sécurité par défaut, en cliquant sur le signe X qui lui est associé.

   - Availability Zone (Zone de disponibilité) : choisissez us-west-2a.

Pour éviter le trafic inter-zones, assurez-vous que l'instance de base de données et l'instance EC2 se trouvent dans la même zone de disponibilité.

7. Pour les sections restantes, spécifiez vos paramètres de cluster de bases de données. Pour plus d'informations sur chaque paramètre, consultez Paramètres pour les clusters de bases de données Aurora.

Se connecter à votre instance Amazon EC2 et à votre cluster de bases de données

Une fois votre instance Amazon EC2 et votre cluster de bases de données créées en mode double pile, vous pouvez vous connecter à chacune d'elles à l'aide du protocole IPv6. Pour vous connecter à une instance Amazon EC2 à l'aide du protocole IPv6, suivez les instructions de la section Connexion à votre instance Linux du Guide de l'utilisateur Amazon EC2.

Pour vous connecter à votre cluster de bases de données Aurora MySQL depuis l'instance Amazon EC2, suivez les instructions dans Se connecter à un cluster de bases de données Aurora MySQL.

Suppression du VPC

Après avoir créé le VPC et d'autres ressources pour ce didacticiel, vous pouvez les supprimer si elles ne sont plus nécessaires.

Si vous avez ajouté des ressources dans le VPC que vous avez créé pour ce tutoriel, vous devrez peut-être les supprimer avant de pouvoir supprimer le VPC. Les instances Amazon EC2 ou les clusters de bases de données sont des exemples de ressources. Pour plus d'informations, consultez Supprimer votre VPC dans le Guide de l'utilisateur Amazon VPC.

Pour supprimer un VPC et les ressources associées

1. Supprimez le groupe de sous-réseaux de base de données :
   a. Ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
   b. Dans le panneau de navigation, choisissez Subnet groups (Groupes de sous-réseaux).
   c. Sélectionnez le groupe de sous-réseaux de base de données à supprimer, par exemple tutorial-db-subnet-group.
   d. Choisissez Supprimer, puis Supprimer dans la fenêtre de confirmation.
2. Notez l'ID du VPC :
   a. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.

b. Choisissez Tableau de bord du VPC, puis VPC.

c. Dans la liste, identifiez le VPC que vous avez créé, tel que tutorial-dual-stack-vpc.

d. Notez la valeur VPC ID (ID de VPC) du VPC que vous avez créé. Il vous servira dans les étapes suivantes.

3. Supprimez les groupes de sécurité :

a. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.

b. Choisissez Tableau de bord du VPC, puis Groupes de sécurité.

c. Sélectionnez le groupe de sécurité pour l'instance de base de données Amazon RDS, par exemple tutorial-dual-stack-db-securitygroup.

d. Pour Actions, choisissez Delete security groups (Supprimer des groupes de sécurité), puis Delete (Supprimer) sur la page de confirmation.

e. Sur la page Security Groups (Groupes de sécurité), sélectionnez le groupe de sécurité pour l'instance Amazon EC2, par exemple tutorial-dual-stack-securitygroup.

f. Pour Actions, choisissez Delete security groups (Supprimer des groupes de sécurité), puis Delete (Supprimer) sur la page de confirmation.

4. Supprimez la passerelle NAT :

a. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.

b. Choisissez Tableau de bord du VPC, puis Passerelles NAT.

c. Sélectionnez la passerelle NAT du VPC que vous avez créé. Utilisez l'ID de VPC pour identifier la passerelle NAT correcte.

d. Pour Actions, choisissez Delete NAT gateway (Supprimer la Passerelle NAT).

e. Sur la page de confirmation, entrez **`delete`** et choisissez Supprimer.

5. Supprimer le VPC

a. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.

b. Choisissez Tableau de bord du VPC, puis VPC.

c. Sélectionnez le VPC que vous voulez supprimer, tel que tutorial-dual-stack-vpc.

d. Pour Actions, choisissez Supprimer le VPC.

La page de confirmation affiche les autres ressources associées au VPC qui seront également supprimées, y compris les sous-réseaux qui lui sont associés.

e. Sur la page de confirmation, entrez **`delete`** et choisissez Supprimer.

6. Libérez les adresses IP élastiques :

    a. Ouvrez la console Amazon EC2 à l’adresse https://console.aws.amazon.com/ec2/.
    b. Choisissez Tableau de bord EC2, puis Adresses IP Elastic.
    c. Sélectionnez l’adresse IP élastique à libérer.
    d. Pour Actions, choisissez Release Elastic IP addresses (Libérer les adresses IP élastiques).
    e. Sur la page de confirmation, sélectionnez Libérer.

Quotas et contraintes pour Amazon Aurora

Vous trouverez ci-après une description des quotas de ressources et des contraintes d'attribution de noms pour Amazon Aurora.

Rubriques

Quotas dans Amazon Aurora

Chaque AWS compte dispose de quotas, pour chaque AWS région, sur le nombre de ressources Amazon Aurora qui peuvent être créées. Une fois qu'un quota de ressource a été atteint, les appels supplémentaires pour créer cette ressource échouent avec une exception.

Le tableau suivant répertorie les ressources et leurs quotas par AWS région.

Nom	Par défaut	Ajusta	Description
Autorisations par groupe de sécurité de base de données	Chaque Région prise en charge : 20	Non	Nombre d'autorisations de groupe de sécurité par groupe de sécurité de base de données
Points de terminaison personnalisés par cluster de bases de données	Chaque Région prise en charge : 5	Oui	Nombre maximal de points de terminaison personnalisés que vous pouvez créer par cluster de bases de données Aurora dans la région actuelle. Cette valeur reflète le plus grand nombre de points de terminaison personnal isés dans un cluster

Nom	Par défaut	Ajusta	Description
			de bases de données du compte. Les autres clusters de bases de données du compte peuvent avoir un nombre inférieur de points de terminaison personnalisés.
Versions de moteur personnalisées	Chaque Région prise en charge : 40	Oui	Nombre maximal de versions de moteur personnalisées autorisées sur ce compte dans la région actuelle
Groupes de paramètres de cluster DB	Chaque Région prise en charge : 50	Oui	Le nombre maximum de groupes de paramètres de cluster de bases de données
Clusters de bases de données	Chaque Région prise en charge : 40	Oui	Le nombre maximum de clusters Aurora autorisés sur ce compte dans la région actuelle
Instances de base de données	ap-south-1 : 20 Chacune des autres régions prises en charge : 40	Oui	Le nombre maximum d'instances de base de données autorisées dans ce compte dans la région actuelle

Nom	Par défaut	Ajusta	Description
Groupes de partitions de base de données	Chaque Région prise en charge : 5	Oui	Nombre maximal de groupes de partitions de base de données pour Aurora Limitless Database pour ce compte dans la région actuelle
Groupes de sous-réseaux DB	ap-south-1 : 20 Chacune des autres régions prises en charge : 50	Oui	Nombre maximal de groupes de sous-réseaux de base de données
Taille du corps de requête HTTP de l'API de données	Toutes les Régions prises en charge : 4 mégaoctets	Non	Taille maximale autorisée pour le corps de la demande HTTP.
Nombre maximal de paires cluster-secret simultanées de l'API de données	Chaque Région prise en charge : 30	Non	Nombre maximal de paires uniques de clusters de base de données Aurora Serverless v1 et de secrets dans les demandes d'API de données simultanées pour ce compte dans la AWS région actuelle.

Nom	Par défaut	Ajusta	Description
Nombre maximal de requêtes simultanées de l’API de données	Chaque Région prise en charge : 500	Non	Nombre maximal de demandes d’API de données envoyées à un cluster de bases de données Aurora sans serveur v1 qui utilisent le même secret et peuvent être traitées en même temps. Les demandes supplémentaires sont mises en file d’attente et traitées à mesure que les demandes en cours de traitement sont terminées.
Taille maximale du jeu de résultats d’API de données	Chaque Région prise en charge : 1 mégaoctet	Non	Taille maximale du jeu de résultats de base de données pouvant être renvoyé par l’API de données.
Taille maximale de l’API de données de la chaîne de réponse JSON	Toutes les régions prises en charge : 10 mégaoctets	Non	Taille maximale de la chaîne de réponse JSON simplifiée renvoyée par l’API de données RDS.
Demandes d’API de données par seconde	Chaque Région prise en charge : 1 000 par seconde	Non	Le nombre maximal de demandes à l'API de données par seconde autorisé pour ce compte dans la AWS région actuelle. Ce quota s’applique uniquement aux clusters Amazon Aurora sans serveur v1.

Nom	Par défaut	Ajusta	Description
Abonnements aux événements	Chaque Région prise en charge : 20	Oui	Le nombre maximum d’abonnements à des événements
Rôles IAM par cluster de bases de données	Chaque Région prise en charge : 5	Oui	Le nombre maximum de rôles IAM associés à un cluster de bases de données
Rôles IAM par instance de base de données	Chaque Région prise en charge : 5	Oui	Le nombre maximum de rôles IAM associés à une instance de base de données
Intégrations	Chaque Région prise en charge : 100	Non	Le nombre maximum d'intégrations autorisées dans ce compte dans la région actuelle AWS
Instantané de cluster de bases de données manuel	Chaque Région prise en charge : 100	Oui	Le nombre maximum d’instantanés manuels du cluster de bases de données
Instantanés d’instance de base de données manuels	Chaque Région prise en charge : 100	Oui	Le nombre maximum d’instantanés manuels de l’instance de base de données
Groupes d’options	Chaque Région prise en charge : 20	Oui	Le nombre maximum de groupes d’options

Nom	Par défaut	Ajusta	Description
Groupes de paramètres	ap-south-1 : 20 Chacune des autres régions prises en charge : 50	Oui	Le nombre maximum de groupes de paramètres
Proxys	Chaque Région prise en charge : 20	Oui	Le nombre maximum de proxys autorisés sur ce compte dans la région actuelle AWS
Réplicas en lecture par principale	Chaque région prise en charge : 15	Oui	Le nombre maximum de réplicas en lecture par instance de base de données principale. Ce quota ne peut pas être ajusté pour Amazon Aurora.
Instances de base de données réservées	ap-south-1 : 20 Chacune des autres régions prises en charge : 40	Oui	Le nombre maximum d'instances de base de données réservées autorisées dans ce compte dans la AWS région actuelle
Groupes de sécurité	ap-south-1 : 20 Chacune des autres régions prises en charge : 25	Oui	Le nombre maximum de groupes de sécurité de base de données

Nom	Par défaut	Ajusta	Description
Sous-réseaux par groupe de sous-réseaux de base de données	Chaque Région prise en charge : 20	Non	Le nombre maximum de sous-réseaux par groupe de sous-réseaux de base de données
Stockage total pour toutes les instances de base de données	Chaque région prise en charge : 100 000	Oui	Le stockage total maximal (en Go) sur les volumes EBS pour toutes les instances de base de données Amazon RDS additionnées. Ce quota ne s'applique pas à Amazon Aurora, qui dispose d'un volume de cluster maximal de 128 Tio pour chaque cluster de bases de données.

Note

Par défaut, vous pouvez avoir jusqu'à 40 instances de bases de données. Les instances de base de données RDS, les instances de base de données Aurora, les instances Amazon Neptune et les instances Amazon DocumentDB sont concernées par ce quota.
Si votre application nécessite plus d'instances de base de données, vous pouvez demander des instances de base de données supplémentaires en ouvrant la console Service Quotas. Dans le panneau de navigation, choisissez Services AWS . Choisissez Amazon Relational Database Service (Amazon RDS), choisissez un quota et suivez les instructions pour demander une augmentation de quota. Pour plus d'informations, consultez Demande d'augmentation de quota dans le Guide de l'utilisateur Service Quotas.
Pour RDS for Oracle, vous pouvez créer jusqu'à 15 réplicas en lecture par instance de base de données source dans chaque région, mais nous recommandons de limiter le nombre de réplicas à 5 afin de minimiser le retard de réplication.

Les sauvegardes gérées par AWS Backup sont considérées comme des instantanés de cluster de base de données manuels, mais ne sont pas prises en compte dans le quota de snapshots de cluster manuel. Pour plus d'informations à ce sujet AWS Backup, consultez le guide du AWS Backup développeur.

Si vous utilisez une opération d'API RDS et dépassez le quota par défaut pour le nombre d'appels par seconde, l'API Amazon RDS émet une erreur similaire à la suivante.

ClientError: Une erreur s'est produite (ThrottlingException) lors de l'appel de l'*API_name*opération : Débit dépassé.

Réduisez ici le nombre d'appels par seconde. Le quota est destiné à couvrir la plupart des cas d'utilisation. Si des quotas plus élevés sont nécessaires, vous pouvez demander une augmentation de quota en utilisant l'une des options suivantes :

- Dans la console, ouvrez la console Service Quotas.
- À partir de AWS CLI, utilisez la request-service-quota-increase AWS CLI commande.

Pour plus d'informations, consultez le Guide de l'utilisateur Service Quotas.

Contraintes d'affectation de noms dans Amazon Aurora

Les contraintes d'affectation de noms dans Amazon Aurora sont les suivantes :

- Identifiant de cluster de bases de données :
  - Doit contenir entre 1 et 63 caractères alphanumériques ou traits d'union.
  - Le premier caractère doit être une lettre.
  - Il ne peut pas se terminer par un trait d'union ou contenir deux traits d'union consécutifs.
  - Doit être unique pour toutes les instances de base de données par AWS compte et par AWS région.
- Nom de la base de données initiale : les contraintes de nom de base de données diffèrent entre Aurora MySQL et Aurora PostgreSQL. Pour plus d'informations, consultez les paramètres disponibles lors de la création de chaque cluster de bases de données.
- Nom d'utilisateur principal : les contraintes relatives à un nom d'utilisateur principal diffèrent pour chaque moteur de base de données. Pour plus d'informations, consultez les paramètres disponibles lors de la création de chaque cluster de bases de données.

- Mot de passe principal :
  - Le mot de passe de l'utilisateur principal de la base de données peut contenir tout caractère ASCII imprimable à l'exception de `/`, `'`, `"`, `@`, ou d'un espace.
  - Le mot de passe peut comporter le nombre suivant de caractères ASCII imprimables selon le moteur de base de données :
    - Aurora MySQL : 8–41
    - Aurora PostgreSQL : 8–99
- Groupe de paramètres de base de données :
  - Ils doivent contenir entre 1 et 255 caractères alphanumériques.
  - Le premier caractère doit être une lettre.
  - Les traits d'union sont autorisés, mais le nom ne peut pas se terminer par un trait d'union ni contenir deux traits d'union consécutifs.
- Groupe de sous-réseaux de base de données :
  - Il doivent contenir entre 1 et 255 caractères.
  - Les caractères alphanumériques, les espaces, les traits d'union, les traits de soulignement et les points sont autorisés.

Limites de taille Amazon Aurora

Limites de taille de stockage

La taille maximale du volume du cluster Aurora varie selon la version du moteur :

256 Tio maximum :

- Versions d'Aurora PostgreSQL :
  - 17.5 et toutes les versions ultérieures
  - 16.9 et ultérieures
  - 15.13 et ultérieures
- Aurora MySQL 3.10 (compatible avec MySQL 8.0.42) et versions ultérieures

128 Tio maximum :

- Toutes les versions antérieures d'Aurora PostgreSQL
- Toutes les versions disponibles d'Aurora MySQL 3 ; Aurora MySQL 2, versions 2.09 et ultérieures

Pour plus d’informations sur la mise à l’échelle automatique du stockage, consultez Redimensionnement automatique du stockage Aurora.

Pour surveiller l’espace de stockage restant, vous pouvez utiliser la métrique `AuroraVolumeBytesLeftTotal`. Pour plus d’informations, consultez Métriques de niveau cluster pour Amazon Aurora.

Limites de taille des tables SQL

Pour un cluster de bases de données Aurora MySQL, la taille maximale de la table est de 64 tébioctets (Tio). Pour un cluster de bases de données Aurora PostgreSQL, la taille maximale de la table est de 32 tébioctets (Tio). Nous vous recommandons de suivre ces bonnes pratiques de conception de tableaux, comme le partitionnement de grands tableaux.

Limites d’identifiant de l’espace de table

L’ID d’espace de table maximal pour Aurora MySQL est 2147483647. Si vous créez et supprimez fréquemment des tables, assurez-vous de connaître votre espace de table IDs et prévoyez d'utiliser des dumps logiques. Pour de plus amples informations, veuillez consulter Migration logique de MySQL vers Amazon Aurora MySQL à l’aide de mysqldump.

Dépannage d’Amazon Aurora

Utilisez les sections suivantes pour résoudre les problèmes que vous rencontrez avec les instances de base de données dans Amazon RDS et Amazon Aurora.

Rubriques

Pour plus d’informations sur le débogage des problèmes à l’aide de l’API Amazon RDS, consultez Applications de dépannage sur Aurora.

Impossible de se connecter à l’instance de base de données Amazon RDS

Voici des causes courantes empêchant la connexion à une instance de base de données :

- Règles entrantes – Les règles d’accès appliquées par votre pare-feu local et les adresses IP autorisées à accéder à votre instance de base de données peuvent ne pas correspondre. Le problème est probablement lié aux règles entrantes de votre groupe de sécurité.

  Par défaut, les instances de base de données n’autorisent pas l’accès. L’accès est accordé via un groupe de sécurité associé au VPC qui autorise le trafic entrant et sortant de l’instance de base de données. Si nécessaire, ajoutez au groupe de sécurité des règles entrantes et sortantes pour votre situation. Vous pouvez indiquer une adresse IP, une plage d’adresses IP ou un autre groupe de sécurité VPC.

> **Note**
>
> Lorsque vous ajoutez une nouvelle règle entrante, vous pouvez choisir Mon adresse IP pour Source afin d'autoriser l'accès à l'instance de base de données à partir de l'adresse IP détectée dans votre navigateur.

Pour plus d'informations sur la configuration des groupes de sécurité, consultez Créer un groupe de sécurité qui autorise l'accès au cluster de bases de données dans le VPC.

> **Note**
>
> Les connexions client à partir d'adresses IP dans la plage 169.254.0.0/16 ne sont pas autorisées. Il s'agit d'une plage d'adresses IP privées automatiques (APIPA, Automatic Private IP Addressing Range), qui est utilisée pour l'adressage de liens locaux.

- Accessibilité publique – Pour vous connecter à votre instance de base de données depuis l'extérieur du VPC, par exemple en utilisant une application cliente, une adresse IP publique doit lui être attribuée.

  Pour rendre l'instance accessible au public, modifiez-la et choisissez Oui sous Public accessibility (Accessibilité publique). Pour plus d'informations, consultez Masquer un cluster de bases de données dans un VPC depuis Internet.
- Port : le port que vous avez spécifié quand vous avez créé l'instance de base de données ne peut pas être utilisé pour envoyer ou recevoir des communications en raison des restrictions de votre pare-feu local. Pour déterminer si votre réseau autorise l'utilisation du port spécifié pour les communications entrantes et sortantes, vérifiez auprès de votre administrateur réseau.
- Disponibilité – Pour une instance de base de données récemment créée, celle-ci a un état `creating` (création) jusqu'à ce qu'elle soit prête à l'emploi. Lorsque l'état devient `available` (disponible), vous pouvez vous connecter à l'instance de base de données. Selon la taille de votre instance de base de données, vous devez parfois patienter une vingtaine de minutes avant qu'elle ne soit disponible.
- Passerelle Internet – Pour qu'une instance de base de données soit publiquement accessible, les sous-réseaux de son groupe de sous-réseaux de base de données doivent avoir une passerelle Internet.

Pour configurer une passerelle Internet pour un sous-réseau

1. Connectez-vous à la AWS Management Console et ouvrez la console Amazon RDS à l'adresse https://console.aws.amazon.com/rds/.
2. Dans le panneau de navigation, sélectionnez Bases de données, puis sélectionnez le nom de l'instance de base de données.
3. Dans l'onglet Connectivity & security (Connectivité et sécurité), notez les valeurs de l'ID du VPC sous VPC et de l'ID du sous-réseau sous Sous-réseaux (subnets).
4. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
5. Dans le panneau de navigation, choisissez Passerelles Internet. Vérifiez qu'il existe une passerelle Internet attachée à votre VPC. Sinon, choisissez Créer une passerelle Internet pour créer une passerelle Internet. Sélectionnez la passerelle Internet, puis choisissez Attacher au VPC et suivez les instructions pour l'attacher à votre VPC.
6. Dans le panneau de navigation, sélectionnez Sous-réseaux, puis sélectionnez votre sous-réseau.
7. Dans l'onglet Table de routage, vérifiez qu'il existe une route avec `0.0.0.0/0` comme destination et la passerelle Internet pour votre VPC comme cible.

   Si vous vous connectez à votre instance à l'aide de son adresse IPv6, vérifiez qu'il existe une route pour tout le trafic IPv6 (`::/0`) qui pointe vers la passerelle Internet. Sinon, procédez comme suit :

   a. Choisissez l'ID de la table de routage (rtb-xxxxxxxx) pour accéder à cette dernière.
   b. Dans l'onglet Routes, choisissez Edit routes (Modifier les routes). Choisissez Add route (Ajouter une route) et utilisez `0.0.0.0/0` comme destination et la passerelle Internet comme cible.

      Pour IPv6, choisissez Add route (Ajouter une route) et utilisez `::/0` comme destination et la passerelle Internet comme cible.
   c. Choisissez Save routes (Enregistrer les routes).

   En outre, si vous essayez de vous connecter à un point de terminaison IPv6, assurez-vous que la plage d'adresses IPv6 du client est autorisée à se connecter à l'instance de base de données.

Pour plus d'informations, consultez Utilisation d'un cluster de bases de données dans un VPC.

Test d'une connexion à une instance de base de données

Vous pouvez tester votre connexion à une instance de base de données à l'aide des outils courants Linux ou Microsoft Windows.

Depuis un terminal Linux ou Unix, vous pouvez tester la connexion en saisissant les informations suivantes. Remplacez *DB-instance-endpoint* par le point de terminaison et *port* par le port de votre instance de base de données.

```
nc -zv DB-instance-endpoint port
```

Par exemple, le code suivant illustre un exemple de commande et la valeur renvoyée.

```
nc -zv postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299

  Connection to postgresql1.c6c8mn7fake0.us-west-2.rds.amazonaws.com 8299 port [tcp/
vvr-data] succeeded!
```

Les utilisateurs Windows peuvent utiliser Telnet pour tester la connexion à une instance de base de données. Les actions Telnet ne sont prises en charge que pour le test de la connexion. Si l'opération aboutit, l'action ne retourne aucun message. Si une connexion n'aboutit pas, vous recevez un message d'erreur similaire au message suivant.

```
C:\>telnet sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com 819

  Connecting To sg-postgresql1.c6c8mntfake0.us-west-2.rds.amazonaws.com...Could not
 open
  connection to the host, on port 819: Connect failed
```

Si les actions Telnet aboutissent, votre groupe de sécurité est correctement configuré.

> **Note**
>
> Amazon RDS n'accepte pas le trafic ICMP (Internet Control Message Protocol), y compris ping.

Dépannage des problèmes d’authentification de connexion

Dans certains cas, vous pouvez vous connecter à votre instance de base de données, mais vous obtenez des erreurs d’authentification. Dans ce cas, il se peut que vous vouliez réinitialiser le mot de passe utilisateur maître de l’instance de base de données. Vous pouvez modifier l’instance RDS pour ce faire.

Problèmes de sécurité Amazon RDS

Pour éviter les problèmes de sécurité, n’utilisez jamais l’adresse e-mail ni le mot de passe d’utilisateur racine d’un Compte AWS pour un compte utilisateur. Une bonne pratique consiste à utiliser votre utilisateur racine pour créer des utilisateurs et les affecter à des comptes d’utilisateur de base de données. Vous pouvez aussi utiliser votre utilisateur racine pour créer d’autres comptes utilisateur si nécessaire.

Pour plus d’informations sur la création d’utilisateurs, consultez Création d’un utilisateur IAM dans votre Compte AWS. Pour plus d’informations sur la création d’utilisateurs dans AWS IAM Identity Center, consultez Manage identities in IAM Identity Center (Gestion des identités dans IAM Identity Center).

Message d’erreur « Échec de l’extraction des attributs du compte. Certaines fonctions de la console sont peut être dégradées. »

Plusieurs raisons peuvent expliquer cette erreur. Cela peut être dû au fait que votre compte ne dispose pas de certaines autorisations ou que votre compte n’a pas été correctement configuré. Si votre compte est nouveau, vous n’avez peut-être pas attendu qu’il soit prêt. S’il s’agit d’un compte existant, il est possible que vos stratégies d’accès ne contiennent pas certaines autorisations permettant d’exécuter certaines actions, comme la création d’une instance de base de données. Pour résoudre le problème, votre administrateur doit fournir les rôles nécessaires à votre compte. Pour plus d’informations, consultez la documentation IAM.

Réinitialisation du mot de passe du propriétaire de l’instance de base de données

Si l’accès à votre cluster de bases de données est verrouillé, vous pouvez vous connecter en tant qu’utilisateur principal. Ensuite, vous pouvez réinitialiser les informations d’identification pour d’autres utilisateurs ou rôles administratifs. Si vous ne parvenez pas à vous connecter en tant

qu'utilisateur principal, le propriétaire du compte AWS peut réinitialiser le mot de passe de l'utilisateur principal. Pour plus d'informations sur les comptes ou rôles administratifs que vous devrez peut-être réinitialiser, consultez Privilèges du compte utilisateur principal.

Vous pouvez modifier le mot de passe d'une instance de base de données à l'aide de la console Amazon RDS, de la commande de l'AWS CLI modify-db-instance ou de l'opération d'API ModifyDBInstance. Pour plus d'informations sur la modification d'une instance de base de données ou d'un cluster de bases de données, consultez Modification d'une instance de base de données dans un cluster de bases de données.

Panne ou redémarrage d'une instance de base de donnéesAmazon RDS

Une instance de base de données peut connaître une panne au redémarrage. Cela peut également se produire quand l'instance de base de données est placée dans un état qui empêche d'y accéder ou quand la base de données est redémarrée. Un redémarrage peut se produire lorsque vous redémarrez manuellement votre instance de base de données. Un redémarrage peut également se produire quand vous modifiez un paramètre de l'instance de base de données qui nécessite un redémarrage avant que la modification ne puisse prendre effet.

Un redémarrage de l'instance de base de données se produit lorsque vous démarrez un paramètre qui nécessite un redémarrage ou quand vous provoquez manuellement un redémarrage. Un redémarrage peut se produire immédiatement si vous modifiez un paramètre et demandez que la modification prenne effet immédiatement. Cela peut également se produire pendant la fenêtre de maintenance de l'instance de base de données.

Un redémarrage d'instance de base de données se produit immédiatement quand l'une des conditions suivantes est vraie :

- Vous remplacez la période de rétention des sauvegardes pour une instance de base de données de 0 par une valeur différente de zéro, ou d'une valeur différente de 0 par 0. Vous définissez ensuite Ajouter un rôle (Appliquer immédiatement) sur `true`.
- Vous pouvez modifier la classe d'instance de base de données et Appliquer immédiatement est défini sur la valeur `true` (vrai).

Un redémarrage d'instance de base de données se produit pendant la fenêtre de maintenance quand l'une des conditions suivantes est vraie :

- Vous remplacez la période de rétention des sauvegardes pour une instance de base de données de 0 par une valeur différente de zéro, ou d'une valeur différente de zéro par zéro, et Appliquer immédiatement est défini sur la valeur `false` (faux).
- Vous pouvez modifier la classe d'instance de base de données et Appliquer immédiatement est défini sur la valeur `false` (vrai).

Lorsque vous modifiez un paramètre statique d'un groupe de paramètres de base de données, la modification prend effet seulement après le redémarrage de l'instance de base de données associée au groupe de paramètres. La modification nécessite un redémarrage manuel. L'instance de base de données n'est pas redémarrée automatiquement pendant la fenêtre de maintenance.

Modifications de paramètre de base de données Amazon RDS n'entrant pas en vigueur

Dans certains cas, vous pouvez modifier un paramètre dans un groupe de paramètres de base de données, mais vous ne voyez pas que les modifications prennent effet. Vous devrez alors probablement redémarrer l'instance de base de données associée au groupe de paramètres de base de données. Lorsque vous modifiez un paramètre dynamique, la modification prend effet immédiatement. Lorsque vous modifiez un paramètre statique, la modification ne prend effet que lorsque vous redémarrez l'instance de base de données associée au groupe de paramètres.

Vous pouvez redémarrer une instance de base de données en utilisant la console RDS. Vous pouvez également appeler explicitement l'opération d'API `RebootDBInstance`. Vous pouvez redémarrer sans basculement si l'instance de base de données se trouve dans un déploiement multi-AZ. Les critères pour redémarrer l'instance de base de données associée après un changement de paramètre statique contribuent à atténuer le risque d'erreur de configuration d'un paramètre affectant un appel d'API. Par exemple, appeler `ModifyDBInstance` pour changer la classe de l'instance de base de données. Pour plus d'informations, consultez Modification de paramètres dans un groupe de paramètres de base de données dans Amazon Aurora.

Problèmes liés à la mémoire libérable dans Amazon Aurora

La mémoire libérable est la quantité totale de mémoire vive (RAM) sur une instance de base de données qui peut être mise à la disposition du moteur de base de données. Il s'agit de la somme de la mémoire libre du système d'exploitation et des mémoires tampon/cache de page disponibles. Le moteur de base de données utilise la plus grande partie de la mémoire sur l'hôte, mais les processus

du système d'exploitation utilisent également une partie de la mémoire vive. La mémoire actuellement allouée au moteur de base de données ou utilisée par les processus du système d'exploitation n'est pas incluse dans la mémoire libérable. Lorsque le moteur de base de données manque de mémoire, l'instance de base de données peut utiliser l'espace temporaire normalement utilisé pour la mise en mémoire tampon et la mise en cache. Comme mentionné précédemment, cet espace temporaire est inclus dans la mémoire libérable.

Vous utilisez la métrique `FreeableMemory` dans Amazon CloudWatch pour surveiller la mémoire libérable. Pour plus d'informations, consultez Surveillance des outils d'Amazon Aurora.

Si votre instance de base de données manque constamment de mémoire libérable ou utilise l'espace d'échange, envisagez d'augmenter la taille de la classe d'instance de base de données. Pour plus d'informations, consultez Classes d'instance de base de données Amazon Aurora.

Vous pouvez également modifier les paramètres de mémoire. Par exemple, sur Aurora MySQL , vous pouvez ajuster la taille du paramètre `innodb_buffer_pool_size`. Ce paramètre est défini par défaut sur 75 % de la mémoire physique. Pour obtenir des conseils de dépannage de MySQL, consultez Comment résoudre les problèmes liés à un manque de mémoire libérable dans une base de données Amazon RDS for MySQL ?

Pour Aurora Serverless v2, `FreeableMemory` représente la quantité de mémoire inutilisée qui est disponible lorsque l'instance de base de données Aurora Serverless v2 est mise à l'échelle jusqu'à sa capacité maximale. La capacité de l'instance peut être réduite à une valeur relativement faible, mais elle indique toujours une valeur élevée pour `FreeableMemory`, car la taille de l'instance peut augmenter. Cette mémoire n'est pas disponible pour le moment, mais vous pouvez l'obtenir si vous en avez besoin.

Pour chaque unité de capacité Aurora (ACU) dont la capacité actuelle est inférieure à la capacité maximale, `FreeableMemory` augmente d'environ 2 Gio. Par conséquent, cette métrique ne se rapproche pas de zéro tant que l'instance de base de données ne fait pas l'objet d'une augmentation d'échelle aussi élevée que possible.

Si cette métrique se rapproche de la valeur `0`, l'instance de base de données a fait l'objet d'une augmentation d'échelle aussi élevée que possible. Elle approche de sa limite de mémoire disponible. Envisagez d'augmenter le nombre maximal d'ACU pour le cluster. Si cette métrique se rapproche de la valeur `0` sur une instance de base de données de lecteur, envisagez d'ajouter des instances de base de données de lecteur supplémentaires au cluster. De cette façon, vous pouvez répartir la partie en lecture seule de la charge de travail sur un plus grand nombre d'instances de base de données, réduisant ainsi l'utilisation de la mémoire sur chaque instance de base de données de lecteur. Pour

plus d'informations, consultez Métriques Amazon CloudWatch importantes pour Aurora Serverless v2.

Pour Aurora Serverless v1, vous pouvez modifier la plage de capacité pour utiliser plus d'ACU. Pour plus d'informations, consultez Modification d'un cluster de bases de données Aurora Serverless v1.

Problèmes de réplication Amazon Aurora MySQL

Certains problèmes de réplication MySQL s'appliquent également à Aurora MySQL. Vous pouvez diagnostiquer et corriger ces problèmes.

Rubriques

- Diagnostic et résolution du retard entre réplicas en lecture
- Diagnostic et résolution d'une défaillance de la réplication en lecture MySQL
- Erreur d'arrêt de réplication
- La réplication du réplica en lecture ne parvient pas à initialiser la structure des métadonnées

Diagnostic et résolution du retard entre réplicas en lecture

Après que vous avez créé un réplica en lecture MySQL et que le réplica en lecture est disponible, Amazon RDS réplique d'abord les modifications apportées à l'instance de base de données source à partir du moment où l'opération de création du réplica en lecture a été initiée. Durant cette phase, la durée du retard de réplication pour le réplica en lecture est supérieure à 0. Vous pouvez superviser ce retard dans Amazon CloudWatch en affichant la métrique Amazon RDS `AuroraBinlogReplicaLag`.

La métrique `AuroraBinlogReplicaLag` contient la valeur du champ `Seconds_Behind_Master` de la commande MySQL `SHOW REPLICA STATUS`. Pour plus d'informations, consultez Instruction SHOW REPLICA STATUS dans la documentation sur MySQL.

Lorsque la métrique `AuroraBinlogReplicaLag` atteint 0, le réplica a rattrapé l'instance de base de données source. Si la métrique `AuroraBinlogReplicaLag` retourne -1, la réplication n'est probablement pas active. Pour résoudre une erreur de réplication, consultez Diagnostic et résolution d'une défaillance de la réplication en lecture MySQL. Une valeur de -1 pour `AuroraBinlogReplicaLag` peut également signifier que la valeur `Seconds_Behind_Master` ne peut pas être déterminée ou qu'elle est `NULL`.

> **Note**
>
> Les versions précédentes d'Aurora MySQL utilisaient `SHOW SLAVE STATUS` à la place de `SHOW REPLICA STATUS`. Si vous utilisez une version Aurora MySQL version 1 ou 2, utilisez alors `SHOW SLAVE STATUS`. Utilisez `SHOW REPLICA STATUS` pour Aurora MySQL version 3 et ultérieures.

La métrique `AuroraBinlogReplicaLag` retourne -1 pendant une panne réseau ou lorsqu'un correctif est appliqué pendant la fenêtre de maintenance. Dans ce cas, attendez que la connexion réseau soit restaurée ou que la fenêtre de maintenance finisse avant de vérifier à nouveau la métrique `AuroraBinlogReplicaLag`.

La technologie de réplication en lecture MySQL est asynchrone. Vous pouvez vous attendre à des augmentations occasionnelles de la métrique `BinLogDiskUsage` sur l'instance de base de données source, et de la métrique `AuroraBinlogReplicaLag` sur le réplica en lecture. Prenez l'exemple d'une situation dans laquelle un volume élevé d'opérations d'écriture sur l'instance de base de données source se produit en parallèle. Au même moment, les opérations d'écriture sur le réplica en lecture sont sérialisées à l'aide d'un seul thread d'I/O. Une telle situation peut entraîner un décalage entre l'instance source et le réplica en lecture.

Pour plus d'informations sur les réplicas en lecture et MySQL, consultez Détails d'implémentation de réplication dans la documentation MySQL.

Vous pouvez réduire le retard entre les mises à jour d'une instance de base de données source et les mises à jour suivantes du réplica en lecture en procédant comme suit :

- Définissez la classe d'instance de base de données du réplica en lecture de telle sorte que sa taille de stockage soit comparable à celle de l'instance de base de données source.
- Veillez à ce que les paramètres des groupes de paramètres de base de données utilisés par l'instance de base de données source et le réplica en lecture soient compatibles. Pour obtenir plus d'informations et un exemple, reportez-vous à la présentation du paramètre `max_allowed_packet` dans la section suivante.
- Désactivez le cache de requête. Pour les tables modifiées fréquemment, l'utilisation du cache de requête peut augmenter le retard, parce que le cache est verrouillé et souvent actualisé. Si tel est le cas, il se peut que vous constatiez un retard de réplica inférieur si vous désactivez le cache de requête. Vous pouvez désactiver le cache de requête en définissant le paramètre `query_cache_type parameter` avec la valeur 0 dans le groupe de paramètres DB de l'instance

de base de données. Pour plus d'informations sur le cache de requête, consultez Configuration du pare-feu Windows.

- Préparez le groupe de tampons sur le réplica en lecture pour InnoDB pour MySQL. Supposons par exemple que vous disposez d'un ensemble réduit de tables mises à jour fréquemment et que vous utilisez le schéma de table InnoDB ou XtraDB. Dans ce cas, videz ces tables sur le réplica en lecture. Le moteur de base de données analyse alors les lignes des tables du disque et les met en cache dans le groupe de tampons. Cette approche peut réduire le retard de réplica. Vous en trouverez un exemple ci-dessous.

  Pour Linux, macOS ou Unix :

```
PROMPT> mysqldump \
    -h <endpoint> \
    --port=<port> \
    -u=<username> \
    -p <password> \
    database_name table1 table2 > /dev/null
```

  Pour Windows :

```
PROMPT> mysqldump ^
    -h <endpoint> ^
    --port=<port> ^
    -u=<username> ^
    -p <password> ^
    database_name table1 table2 > /dev/null
```

Diagnostic et résolution d'une défaillance de la réplication en lecture MySQL

Amazon RDS surveille l'état de réplication de vos réplicas lus. RDS met à jour le champ Replication State (État de réplication) de l'instance du réplica en lecture sur `Error` si la réplication s'arrête pour une raison quelconque. Vous pouvez passer en revue les détails de l'erreur associée et déclenchée par les moteurs MySQL, en consultant le champ Erreur de réplication. Des événements indiquant l'état du réplica en lecture sont également générés, y compris RDS-EVENT-0045, RDS-EVENT-0046 et RDS-EVENT-0057. Pour plus d'informations sur les événements et l'abonnement aux événements, consultez Utiliser la notification d'événements d'Amazon RDS. Si un message d'erreur MySQL est renvoyé, consultez l'erreur dans la documentation sur les messages d'erreur MySQL.

Voici d'autres situations courantes susceptibles d'entraîner des erreurs de réplication :

- La valeur du paramètre `max_allowed_packet` d'un réplica en lecture est inférieure au paramètre `max_allowed_packet` de l'instance de base de données source.

  Le paramètre `max_allowed_packet` est un paramètre personnalisé que vous pouvez définir dans un groupe de paramètres de base de données. Le paramètre `max_allowed_packet` est utilisé pour spécifier la taille maximale du langage de manipulation de données (DML) qui peut être exécuté sur la base de données. Dans certains cas, la valeur `max_allowed_packet` de l'instance de base de données source peut être supérieure à la valeur `max_allowed_packet` du réplica en lecture. Dans ces cas, le processus de réplication peut lancer une erreur et arrêter la réplication. L'erreur la plus courante est `packet bigger than 'max_allowed_packet' bytes`. Vous pouvez corriger cette erreur en indiquant à la source et au réplica en lecture d'utiliser des groupes de paramètres de base de données avec les mêmes valeurs du paramètre `max_allowed_packet`.
- Écriture sur les tables d'un réplica en lecture. Si vous créez des index sur un réplica en lecture, le paramètre `read_only` doit être défini sur 0 pour créer les index. Si vous écrivez dans des tables sur le réplica en lecture, cela peut interrompre la réplication.
- Utilisation d'un moteur de stockage non transactionnel tel que MyISAM. Les réplicas en lecture nécessitent un moteur de stockage transactionnel. La réplication n'est prise en charge que pour les moteurs de stockage suivants : InnoDB pour MySQL ou MariaDB.

  Pour convertir une table MyISAM en InnoDB, exécutez la commande suivante :

  ```
  alter table <schema>.<table_name> engine=innodb;
  ```
- Utilisation de requêtes non déterministes non sécurisées telles que `SYSDATE()`. Pour plus d'informations, consultez Determination of Safe and Unsafe Statements in Binary Logging dans la documentation MySQL.

Les étapes suivantes peuvent vous aider à résoudre votre erreur de réplication :

- Si vous rencontrez une erreur logique et que vous pouvez l'ignorer en toute sécurité, suivez la procédure décrite dans Skipping the Current Replication Error (Ignorer l'erreur de réplication actuelle). Votre instance de base de données Aurora MySQL doit exécuter une version incluant la procédure `mysql_rds_skip_repl_error`. Pour plus d'informations, consultez mysql_rds_skip_repl_error.

- Si vous rencontrez un problème de position de journal binaire, vous pouvez modifier la position de relecture du réplica. Pour ce faire, utilisez la commande `mysql.rds_next_master_log` pour Aurora MySQL versions 1 et 2. Pour ce faire, utilisez la commande `mysql.rds_next_source_log` pour Aurora MySQL versions 3 et ultérieures. Votre instance de base de données Aurora MySQL doit exécuter une version prenant en charge cette commande afin de pouvoir modifier la position de relecture du réplica. Pour plus d'informations sur la version, consultez mysql_rds_next_master_log.
- Si vous rencontrez temporairement un problème de performance en raison d'une charge DML élevée, vous pouvez spécifier la valeur 2 pour le paramètre `innodb_flush_log_at_trx_commit` dans le groupe de paramètres de base de données du réplica en lecture. Cette action peut aider le réplica en lecture à se rattraper, même si l'atomicité, la cohérence, l'isolation et la durabilité s'en trouvent temporairement réduites.
- Vous pouvez supprimer le réplica en lecture et créer une instance à l'aide du même identifiant d'instance de base de données. Dans ce cas, le point de terminaison reste le même que celui de votre ancien réplica en lecture.

Si une erreur de réplication est corrigée, le champ Replication State (Statut de réplication) prend la valeur replicating (réplication en cours). Pour plus d'informations, consultez Résolution d'un problème de réplica en lecture MySQL.

Erreur d'arrêt de réplication

Lorsque vous appelez la commande `mysql.rds_skip_repl_error`, un message d'erreur peut s'afficher pour indiquer que la réplication a rencontré une erreur ou est désactivée.

Ce message d'erreur s'affiche, car la réplication a été arrêtée et ne peut pas être redémarrée.

Si vous avez besoin d'ignorer un grand nombre d'erreurs, le retard de réplication peut augmenter et dépasser la période de rétention par défaut pour les fichiers journaux binaires. Dans ce cas, vous pouvez rencontrer une erreur irrécupérable due à des fichiers journaux binaires purgés avant d'avoir été réutilisés sur le réplica. Cette purge entraîne l'arrêt de la réplication et vous ne pouvez plus appeler la commande `mysql.rds_skip_repl_error` pour ignorer les erreurs de réplication.

Vous pouvez atténuer ce problème en augmentant le nombre d'heures pendant lequel les fichiers journaux binaires sont conservés sur votre source de réplication. Une fois que vous avez augmenté le temps de rétention de journaux binaires, vous pouvez redémarrer la réplication et appeler la commande `mysql.rds_skip_repl_error` en fonction des besoins.

Pour définir le temps de rétention du journal binaire, utilisez la procédure mysql_rds_set_configuration. Spécifiez un paramètre de configuration des heures de rétention des journaux binaires, ainsi que le nombre d'heures pendant lequel conserver les fichiers journaux binaires sur le cluster de bases de données, 2 160 heures au plus (90 jours). La valeur par défaut de Aurora MySQL est de 24 heures (1 jour). L'exemple suivant définit la période de rétention des fichiers journaux binaires à 48 heures.

```
CALL mysql.rds_set_configuration('binlog retention hours', 48);
```

La réplication du réplica en lecture ne parvient pas à initialiser la structure des métadonnées

Lorsque vous avez essayé de démarrer la réplication, vous avez reçu le message d'erreur suivant :

```
Read Replica Replication Error - SQLError: 13124, reason: Replica failed to initialize
 applier metadata structure from the repository
```

Cette erreur se produit en cas de problème lié à la structure des métadonnées du réplica. Pour corriger la structure des métadonnées, vous devez créer un nouveau réplica.

Pour éviter que cela ne se reproduise à l'avenir, effectuez une des actions suivantes :

- Si possible, désactivez le multithreading sur vos réplicas. À partir de MySQL 8.0.27, le multithreading est activé par défaut.
- Si vous devez utiliser le multithreading sur vos réplicas, nous vous recommandons d'utiliser la réplication basée sur le GTID. Pour plus d'informations, consultez Utilisation de la réplication basée sur des identifiants de transaction globaux (GTID).

Référence d'API Amazon RDS

Outre la AWS Management Console et l'AWS Command Line Interface (AWS CLI), Amazon RDS fournit également une API. Vous pouvez utiliser l'API pour automatiser les tâches de gestion de vos instances de base de données et d'autres objets dans Amazon RDS.

- Pour obtenir la liste alphabétique des opérations d'API, consultez Actions.
- Pour obtenir la liste alphabétique des types de données, consultez Types de données.
- Pour consulter la liste des paramètres de requête courants, reportez-vous à la page Paramètres courants.
- Pour la description des codes d'erreur, veuillez consulter la page Erreurs courantes.

Pour plus d'informations sur l'AWS CLI, consultez Référence de l'AWS Command Line Interface pour Amazon RDS.

Rubriques

Utilisation de l'API Query

Les sections suivantes abordent brièvement l'authentification de la demande et les paramètres utilisés avec l'API Query.

Pour obtenir des informations générales sur le fonctionnement de l'API Query, veuillez consulter Demandes de requête dans le Amazon EC2 API Reference.

Paramètres Query (Requête)

Ces demandes basées sur Query HTTP sont des demandes HTTP qui utilisent le verbe HTTP GET ou POST et un paramètre Query appelé `Action`.

Chaque demande Query doit inclure certains paramètres communs pour gérer l'authentification et la sélection d'une action.

Certaines actions demandent des listes de paramètres. Ces listes sont spécifiées en utilisant la notation `param.`*n*. Les valeurs de *n* sont des nombres entiers à partir de 1.

Pour plus d'informations sur les régions et les points de terminaison Amazon RDS, consultez Amazon Relational Database Service (RDS) dans la section Régions et points de terminaison de la Référence générale d'Amazon Web Services.

Authentification de demande Query

Vous pouvez uniquement envoyer des demandes Query via HTTPS, et vous devez inclure une signature dans chaque demande Query. Vous devez utiliser le processus AWS Signature Version 4 ou 2. Pour de plus amples informations, veuillez consulter Processus de signature Signature Version 4 et Processus de signature Signature Version 2.

Applications de dépannage sur Aurora

Amazon RDS fournit des erreurs spécifiques et descriptives pour vous aider à résoudre vos problèmes tout en interagissant avec l'API Amazon RDS.

Rubriques

- Récupération d'erreurs
- Conseils pour le dépannage

Pour de plus amples informations sur le dépannage des instances de base de données Amazon RDS, veuillez consulter Dépannage d'Amazon Aurora.

Récupération d'erreurs

Généralement, vous souhaitez que votre application vérifie si une demande a généré une erreur avant de passer du temps à traiter les résultats. Le moyen le plus simple de déterminer si une erreur s'est produite est de rechercher un nœud `Error` dans la réponse de l'API Amazon RDS.

La syntaxe XPath fournit une méthode simple pour rechercher la présence d'un nœud `Error`. Elle fournit également un moyen relativement simple de récupérer le code et le message d'erreur. L'extrait de code suivant utilise Perl et le module XML::XPath pour déterminer si une erreur s'est produite lors d'une demande. Si une erreur s'est produite, le code imprime le premier code et message d'erreur dans la réponse.

```
use XML::XPath;
    my $xp = XML::XPath->new(xml =>$response);
    if ( $xp->find("//Error") )
```

```
    {print "There was an error processing your request:\n", " Error code: ",
    $xp->findvalue("//Error[1]/Code"), "\n", " ",
    $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Conseils pour le dépannage

Nous vous conseillons d'utiliser les processus suivants pour diagnostiquer et résoudre les problèmes avec l'API Amazon RDS :

- Vérifiez qu'Amazon RDS fonctionne normalement dans la région AWS que vous ciblez en consultant la page http://status.aws.amazon.com.
- Vérifiez la structure de votre demande.

  Chaque opération Amazon RDS possède une page de référence dans la référence de l'API Amazon RDS. Revérifiez que vous utilisez les paramètres correctement. Pour des idées sur les éventuels problèmes, observez les exemples de demandes ou de scénarios utilisateur pour voir s'ils effectuent des opérations similaires.
- Consultez AWS re:Post.

  Amazon RDS possède une communauté de développement où vous pouvez chercher des solutions aux problèmes rencontrés par d'autres. Pour consulter les rubriques, accédez à AWS re:Post.

Historique du document

Version de l'API actuelle : 2014-10-31

Le tableau suivant décrit les modifications importantes apportées au Guide de l'utilisateur Amazon Aurora. Pour recevoir les notifications des mises à jour de cette documentation, abonnez-vous à un flux RSS. Pour plus d'informations sur Amazon Relational Database Service (Amazon RDS), consultez le Amazon Relational Database Service User Guide (Guide de l'utilisateur Amazon Relational Database Service).

> **Note**
>
> Avant le 31 août 2018, Amazon Aurora était intégré au Guide de l'utilisateur Amazon Relational Database Service. Pour accéder à l'historique des documents Aurora antérieurs, consultez Historique du document dans le Guide de l'utilisateur Amazon Relational Database Service.

Vous pouvez filtrer les nouvelles fonctions de Amazon Aurora sur la page Nouveautés en matière de base de données. Pour Produits, choisissez Amazon Aurora. Ensuite, effectuez une recherche à l'aide de mots clés tels que **`global database`** ou **`Serverless`**.

Modification	Description	Date
Intégrations Aurora PostgreSQL Zero-ETL avec prise en charge de régions supplémentaires Amazon SageMaker	Les intégrations Zero-ETL avec Aurora PostgreSQL sont désormais disponibles dans Amazon SageMaker d'autres régions. Pour plus d'informations, consultez Régions prises en charge et moteurs de base de données Aurora pour les intégrations sans ETL.	15 octobre 2025
Base de données Aurora PostgreSQL Limitless	La base de données Aurora PostgreSQL Limitless est	16 septembre 2025

disponible dans une région supplémentaire AWS	désormais disponible dans le. AWS GovCloud (US) Regions Pour plus d'informations, consultez la section Exigences et considérations relatives à la base de données Aurora PostgreSQL Limitless.	
L'API de données Amazon RDS prend en charge IPv6	L'API Amazon RDS Data prend désormais en charge la IPv6 connectivité, ce qui vous permet de vous connecter à vos bases de données Aurora à l'aide d' IPv6 adresses. Pour plus d'informations, consultez Utilisation de l'API de données.	30 août 2025
Amazon Aurora est disponible dans la région Asie-Pacifique (Nouvelle Zélande)	Amazon Aurora est désormais disponible dans la région Asie-Pacifique (Nouvelle Zélande). Pour plus d'informations, consultez Régions et zones de disponibilité.	28 août 2025
API de données Amazon RDS disponible dans une région supplémentaire AWS	L'API de données Amazon RDS est désormais disponible dans la région Europe (Espagne). Pour plus d'informations, consultez Régions et moteurs de base de données pris en charge l'API de données RDS.	1er août 2025

Les intégrations zéro ETL à Amazon SageMaker Lakehouse sont généralement disponibles	Les intégrations zéro ETL rendent les données transacti onnelles disponibles dans Amazon SageMaker quelques secondes après leur écriture dans une instance de base de données Aurora MySQL. Pour plus d'informations, consultez Utilisation avec Aurora des intégrations zéro ETL.	30 juin 2025
Amazon Aurora est disponible dans la région Asie-Pacifique (Taipei)	Amazon Aurora est désormais disponible dans la région Asie-Pacifique (Taipei). Pour plus d'informations, consultez Régions et zones de disponibi lité.	5 juin 2025

End-of-life informations pour Performance Insights	AWS a annoncé la end-of-life date de Performance Insights : le 30 novembre 2025. Après cette date, Amazon RDS ne prendra plus en charge l'expérience de la console Performance Insights, les périodes de conservation flexibles et les tarifs associés. Nous vous recommandons de mettre à niveau vos clusters de bases de données vers Database Insights avant cette date. Pour obtenir plus d'informations, consultez Surveillance de la charge de la base de données avec Performance Insights sur Amazon Aurora.	30 mai 2025
La base de données Aurora PostgreSQL Limitless prend en charge Database Insights CloudWatch	Vous pouvez désormais utiliser CloudWatch Database Insights pour surveiller votre base de données Amazon Aurora PostgreSQL Limitless à grande échelle. Pour plus d'informations, consultez la section Surveillance de la base de données Amazon Aurora PostgreSQL Limitless avec Database Insights. CloudWatch	22 mai 2025

Affichage des dates de support pour les versions majeures du moteur open source	Vous pouvez désormais consulter les informations sur les dates de début et de fin des versions majeures des moteurs open source dans le support standard RDS et le support étendu RDS à l'aide de l'API AWS CLI et de l'API RDS. Pour plus d'informations, consultez Affichage des dates de support pour les versions de moteur dans Support étendu Amazon RDS.	20 mai 2025
Les intégrations zéro ETL d'Aurora PostgreSQL prennent en charge des régions supplémentaires.	Les intégrations zéro ETL d'Aurora PostgreSQL sont désormais disponibles dans des régions supplémentaires. Pour plus d'informations, consultez Régions et moteurs de base de données Aurora pour les intégrations zéro ETL à Amazon Redshift.	17 février 2025
Amazon Aurora est disponible dans la région du Mexique (centre)	Amazon Aurora est désormais disponible dans la région du Mexique (centre). Pour plus d'informations, consultez Régions et zones de disponibilité.	14 janvier 2025
Amazon Aurora est disponible dans la région Asie-Pacifique (Thaïlande)	Amazon Aurora est désormais disponible dans la région Asie-Pacifique (Thaïlande). Pour plus d'informations, consultez Régions et zones de disponibilité.	7 janvier 2025

End-of-life informations pour Aurora Serverless v1	AWS a annoncé la end-of-life date du Aurora Serverless v1 : 31 mars 2025. Nous recommandons vivement de mettre à niveau tous les clusters de bases de données Aurora Serverless v1 vers Aurora Serverless v2 avant cette date. La mise à niveau peut impliquer une modification du numéro de version majeure du moteur de base de données. Il est donc important de planifier, de tester et de mettre en œuvre cette transition avant la end-of-life date prévue. À compter du 8 janvier 2025, les clients ne seront plus en mesure de créer de nouveaux Aurora Serverless v1 clusters ou instances avec la CLI AWS Management Console ou la CLI. Pour plus d'informations sur la procédure de mise à niveau de Aurora Serverless v1 vers Aurora Serverless v2, consultez Mise à niveau d'un cluster Aurora Serverless v1 vers Aurora Serverless v2.	6 décembre 2024

Aurora prend en charge CloudWatch Database Insights	Vous pouvez désormais utiliser CloudWatch Database Insights pour surveiller vos clusters de bases de données Amazon Aurora à grande échelle. Pour plus d'informations, consultez la section Surveillance des bases de données Amazon Aurora avec CloudWatch Database Insights.	1er décembre 2024
Amazon Aurora prend en charge les classes d'instance db.r8g	Vous pouvez désormais utiliser les classes d'instance db.r8g pour créer des clusters de bases de données Aurora. Pour plus d'informations, consultez Classes d'instance de base de données Aurora.	21 novembre 2024
Aurora Serverless v2 prend en charge la mise en pause et la reprise automatiques	Aurora Serverless v2Les instances de base de données peuvent désormais être redimensionnées jusqu'à zéro unité de capacité Aurora (ACUs) après une période d'inactivité. Pour activer cette fonctionnalité, vous devez définir la valeur de capacité minimale sur 0 ACUs. Pour plus d'informations, consultez Mise en pause et reprise automatiques pour Aurora Serverless v2.	20 novembre 2024

Amazon Aurora prend en charge la migration automatique EC2 des bases de données	Vous pouvez utiliser la console RDS pour migrer une EC2 base de données vers Aurora. Aurora utilise AWS Database Migration Service (AWS DMS) pour migrer votre EC2 base de données source. AWS DMS vous permet de migrer des bases de données relationnelles dans votre AWS Cloud. Pour plus d'informations, consultez Migration automatique des EC2 bases de données vers Amazon Aurora à l'aide AWS Database Migration Service de.	20 novembre 2024
AWS Wrapper NodeJS avancé généralement disponible	Le wrapper Advanced NodeJS (AWS) Amazon Web Services complète et étend les fonctionnalités d'un pilote NodeJS existant. Pour plus d'informations, consultez la section Connexion aux clusters de base de données Aurora avec les AWS pilotes.	19 novembre 2024
Amazon Aurora prend en charge les classes d'instance db.r7i	Vous pouvez désormais utiliser les classes d'instance db.r7i pour créer des clusters de bases de données Aurora. Pour plus d'informations, consultez Classes d'instance de base de données Aurora.	18 novembre 2024

L’API de données RDS pour Aurora MySQL prend en charge toutes les vues Performance Insights	Vous pouvez désormais utiliser les vues Meilleurs hôtes ou Principales applications pour Performance Insights afin de surveiller les opérations SQL effectuées à l’aide de l’API de données RDS pour Aurora MySQL. Pour plus d’informations sur la prise en charge de Performance Insights, consultez Surveillance des requêtes de l’API de données RDS avec Performance Insights.	31 octobre 2024
Base de données Aurora PostgreSQL Limitless	La base de données Amazon Aurora PostgreSQL Limitless est une nouvelle fonctionnalité de mise à l’échelle horizontale (sharding) automatisée d’Amazon Aurora. La base de données Aurora PostgreSQL Limitless vous permet de dépasser les limites actuelles d’Aurora en matière de débit d’écriture et de stockage en répartissant la charge de travail d’une base de données sur plusieurs instances d’Aurora Writer, tout en conservant la possibilité de l’utiliser comme une seule base de données. Pour plus d’informations, consultez Base de données Amazon Aurora PostgreSQL Limitless.	31 octobre 2024

Fin du support standard d'Amazon Aurora MySQL version 2	La version 2 d'Aurora MySQL a atteint sa fin de support standard le 31 octobre 2024. Pour plus d'informations, consultez Préparation à la fin de prise en charge d'Amazon Aurora Édition compatible avec MySQL version 2.	31 octobre 2024
Amazon Aurora prend en charge les mises à niveau du système d'exploitation au niveau du cluster	Vous pouvez désormais appliquer les mises à niveau du système d'exploitation Aurora au niveau du cluster de bases de données. Les mises à niveau progressives appliquent automatiquement les mises à niveau à quelques instances de base de données de lecteur à la fois, préservant ainsi la disponibilité en lecture. Pour plus d'informations, consultez Mises à jour du système d'exploitation pour les clusters de bases de données Aurora.	30 octobre 2024

Les clusters globaux Aurora prennent en charge le balisage	Vous pouvez désormais ajouter des balises aux clusters globaux Aurora. Un cluster global Aurora est la ressource parent qui contient les clusters de bases de données principal et secondaire d'une base de données globale Aurora. Pour obtenir des informations générales sur le balisage, consultez Balisage des ressources Aurora et Amazon RDS. Pour plus d'informations concernant le balisage avec les bases de données globales Aurora, consultez Balisage des ressources de bases de données globales Aurora.	24 octobre 2024

Points de terminaison d'enregistreur Aurora Global Database	Chaque base de données Amazon Aurora Global Database est fournie avec un point de terminaison d'enregistreur qui est automatiquement mis à jour par Aurora pour acheminer les demandes vers l'instance d'enregistreur actuelle du cluster de bases de données primaire afin une opération de basculement et de bascule d'une base de données globale Aurora. Pour en savoir plus sur l'utilisation du point de terminaison d'enregistreur avec la bascule et le basculement d'Amazon Aurora Global Database, consultez Utilisation de la bascule ou du basculement dans Amazon Aurora Global Database.	22 octobre 2024
Intégrations zéro ETL Aurora PostgreSQL à Amazon Redshift désormais généralement disponibles	Les intégrations zéro ETL rendent les données transactionnelles disponibles dans Amazon Redshift quelques secondes après leur écriture dans un cluster de bases de données source Aurora PostgreSQL. La fonctionnalité est désormais généralement disponible. Pour plus d'informations, consultez Utilisation des intégrations zéro ETL d'Aurora à Amazon Redshift.	15 octobre 2024

Aurora Serverless v2prend en charge 256 ACUs	Vous pouvez désormais créer des clusters de Aurora Serverless v2 base de données d'une capacité maximale de 256 unités de capacité Aurora (ACUs). Pour plus amples informations, consultez Capacité d'Aurora Serverless v2.	3 octobre 2024
Amazon Aurora prend en charge Console-to-Code	Vous pouvez désormais l'utiliser Console-to-Code pour générer du code à partir d'actions que vous effectuez dans la console RDS. Le code généré peut vous aider à écrire du code pour automatis er votre utilisation d'autres AWS services. Pour plus d'informations, voir Utiliser Console-to-Code pour générer du code pour les actions de votre console Aurora.	3 octobre 2024
L'API de données RDS prend en charge Performance Insights	Vous pouvez désormais utiliser Performance Insights pour surveiller les opération s SQL effectuées à l'aide de l'API de données RDS. Pour plus d'informations sur la prise en charge de Performance Insights, consultez Surveilla nce des requêtes de l'API de données RDS avec Performan ce Insights.	26 septembre 2024

L’API de données RDS pour Aurora Serverless v2 et mise en service est disponible pour Aurora MySQL	L’API RDS Data est désormais disponible pour les clusters Aurora MySQL qui utilisent Aurora Serverless v2 ou des instances provisionnées. Pour obtenir des informations sur l’utilisation de l’API de données RDS, consultez Utilisation de l’API de données RDS. Pour plus d'informations sur la prise en charge des versions et des AWS régions d'Aurora MySQL pour l'API de données RDS, consultez la section API de données avec Aurora MySQL Serverless v2 et provisionnée.	26 septembre 2024
Amazon Aurora est disponible dans la Région Asie-Pacifique (Malaisie)	Amazon Aurora est désormais disponible dans la Région Asie-Pacifique (Malaisie). Pour plus d’informations, consultez Régions et zones de disponibilité.	22 août 2024
Mise à jour de la politique existante	Amazon RDS a supprimé l’autorisation `sns:Publish` de `AmazonRDSBetaServiceRolePolicy` du rôle lié au service `AWSServiceRoleForRDSBeta`. Pour plus d'informations, consultez les mises à jour des politiques AWS gérées par Amazon RDS.	7 août 2024

Mise à jour de la politique existante	Amazon RDS a supprimé l'autorisation `sns:Publish` de `AmazonRDSPreviewServiceRolePolicy` du rôle lié au service `AWSServiceRoleForRDSPreview` . Pour plus d'informations, consultez les mises à jour des politiques AWS gérées par Amazon RDS.	7 août 2024
Aurora Serverless v2prend en charge la pause automatique () scale-to-zero	Aurora Serverless v2Les instances de base de données peuvent désormais être redimensionnées jusqu'à zéro unité de capacité Aurora (ACUs) après une période d'inactivité. Pour activer cette fonctionnalité, définissez la valeur de capacité minimale sur 0. Pour plus d'informations, consultez Utilisation de Aurora Serverless v2.	25 juillet 2024
AWS Le pilote ODBC pour MySQL est généralement disponible	Le pilote ODBC Amazon Web Services (AWS) pour MySQL est un pilote client conçu pour la haute disponibilité d'Aurora MySQL. Pour en savoir plus, consultez Connexion à Aurora MySQL avec le pilote ODBC Amazon Web Services (AWS) pour MySQL.	18 juillet 2024

L’API de données RDS pour Aurora Serverless v2 est disponible dans un plus grand nombre de régions	L'API de données RDS est désormais disponible pour Aurora PostgreSQL dans plusieurs régions supplémentaires. AWS Pour plus d’informations sur la prise en charge de région pour l’API de données RDS, consultez API de données avec Aurora PostgreSQL sans serveur v2 et provisionnée.	9 juillet 2024
AWS Pilote Python généralement disponible	Le pilote Python Amazon Web Services (AWS) est conçu comme un wrapper Python avancé. Ce wrapper complète et étend les fonctionnalités du pilote open source Psycopg. Pour plus d'informations, consultez la section Connexion aux clusters de base de données Aurora avec les AWS pilotes.	23 mai 2024
Intégrations zéro ETL disponibles dans les régions de Chine	Les intégrations Zero-ETL sont désormais disponibles en Régions AWS Chine (Pékin) et en Chine (Ningxia). Pour plus d’informations, consultez Intégrations zéro ETL à Amazon Redshift.	21 mai 2024

Le proxy RDS est disponible dans des régions supplémentaires	Le proxy RDS est désormais disponible dans les régions Asie-Pacifique (Hyderabad), Asie-Pacifique (Melbourne), Moyen-Orient (EAU), Israël (Tel Aviv), Canada-Ouest (Calgary) et Europe (Zurich). Pour plus d'informations sur RDS Proxy, consultez Utilisation d'Amazon RDS Proxy.	21 mai 2024
Amazon RDS prend en charge un accès affiné pour Performance Insights	Vous pouvez désormais autoriser ou refuser l'accès à des dimensions individuelles dans Performance Insights. Cet accès affiné peut être utilisé pour les actions `GetResourceMetrics`, `DescribeDimensionKeys` et `GetDimensionKeyDetails`. Pour plus d'informations, consultez https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PerfInsights.access-control.dimensionAccess-policy.html.	21 mai 2024

Support étendu Amazon RDS	La création ou la restauration d'une base de données Aurora MySQL version 2 ou 3, ou Aurora PostgreSQL version 11 inscrit désormais automatiquement cette base de données dans Support étendu Amazon RDS afin que vos applications existantes continuent de fonctionner telles quelles. Vous pouvez vous désinscrire du support étendu RDS pour éviter des frais après la date de fin du support standard Aurora pour votre moteur de base de données. Pour plus d'informations, consultez Utilisation du support étendu Amazon RDS.	21 mars 2024
Filtrage des données pour les intégrations zéro ETL	Amazon RDS prend en charge le filtrage des données au niveau de la base de données et des tables pour les intégrations zéro ETL à Amazon Redshift. Pour plus d'informations, consultez Data filtering for Aurora zero-ETL integrations with Amazon Redshift.	20 mars 2024

Intégrations d’Aurora MySQL à Amazon Bedrock	Vous pouvez désormais intégrer les bases de données Amazon Aurora MySQL à Amazon Bedrock pour alimenter des applications d’IA génératives. Pour plus d’informations sur le machine learning Amazon Aurora, consultez Utilisation de l’apprentissage automatique d’Amazon Aurora avec Aurora My SQL.	8 mars 2024
Nouvelle politique AWS gérée	Amazon RDS a ajouté une nouvelle politique gérée nommée `AmazonRDS Custom InstanceProfileRo lePolicy` pour permettre à RDS Custom d'effectuer des actions d'automatisation et des tâches de gestion de base de données via un profil d' EC2 instance. Pour plus d’informa tions, consultez Mises à jour Amazon RDS des politiques gérées par AWS.	27 février 2024
Assistance Amazon RDS pour la AWS Secrets Manager région d'Israël (Tel Aviv)	Amazon RDS prend en charge Secrets Manager dans la région Israël (Tel Aviv). Pour plus d’informations, consultez Gestion des mots de passe avec Amazon RDS et AWS Secrets Manager.	21 février 2024

Support étendu Amazon RDS	Amazon RDS active désormais automatiquement Support étendu Amazon RDS lorsque les versions majeures de moteur Aurora MySQL et Aurora PostgreSQL dans vos clusters de bases de données et clusters globaux atteignent la date de fin du support standard Aurora. Pour plus d'informations, consultez Utilisation du support étendu Amazon RDS.	15 février 2024
Aurora PostgreSQL 16.1 prend en charge Babelfish pour Aurora PostgreSQL 4.0.0	Aurora PostgreSQL 16.1 prend en charge Babelfish 4.0.0. Pour obtenir la liste des nouvelles fonctions, consultez 16.1. Pour une liste des fonctionnalités prises en charge dans chaque version de Babelfish, consultez Supported functionality in Babelfish by version (Fonctionnalités prises en charge dans Babelfish par version). Pour obtenir des informations sur l'utilisation, consultez Working with Babelfish pour Aurora PostgreSQL (Utilisation de Babelfish pour Aurora PostgreSQL).	31 janvier 2024

Mise à jour du certificat CA par défaut	Le certificat CA par défaut est défini sur `rds-ca-rsa2048-g1` . Pour plus d'informations, consultez la section Utilisation SSL/TLS pour chiffrer une connexion à un cluster de bases de données.	26 janvier 2024
Le proxy RDS est disponible dans la région Europe (Espagne)	Le proxy RDS est désormais disponible dans la région Europe (Espagne). Pour plus d'informations sur RDS Proxy, consultez Utilisation d'Amazon RDS Proxy.	8 janvier 2024
API de données RDS avec Aurora PostgreSQL sans serveur v2 et provisionné	Vous pouvez désormais utiliser l'API de données RDS avec Aurora PostgreSQL sans serveur v2 et les clusters de bases de données provisionnés. Avec l'API de données RDS, vous pouvez accéder à vos clusters Aurora via un point de terminaison HTTP sécurisé et exécuter des instructions SQL sans utiliser de pilotes de base de données ni gérer de connexions. Pour plus d'informations, consultez Utilisation de l'API de données RDS.	21 décembre 2023

Intégrations d’Aurora PostgreSQL à Amazon Bedrock	Vous pouvez désormais intégrer les bases de données Amazon Aurora PostgreSQL à Amazon Bedrock pour alimenter des applications d’IA génératives. Pour plus d’informations sur le machine learning Amazon Aurora, consultez Utilisation de l’apprentissage automatique d’Amazon Aurora avec Aurora PostgreSQL.	21 décembre 2023
Amazon Aurora est disponible dans la région Canada-Ouest (Calgary)	Amazon Aurora est désormais disponible dans la région Canada-Ouest (Calgary). Pour plus d’informations, consultez Régions et zones de disponibilité.	20 décembre 2023
Amazon RDS permet de consulter les recommandations et d’y répondre	Les recommandations d’Amazon Aurora incluent désormais des recommandations proactives basées sur des seuils et des recommandations réactives basées sur le machine learning. Pour plus d’informations, consultez Recommandations pour Amazon Aurora.	19 décembre 2023

Intégrations zéro ETL d’Aurora PostgreSQL à Amazon Redshift (aperçu)	Vous pouvez désormais créer des intégrations zéro ETL à Amazon Redshift à l’aide d’un cluster de bases de données source Aurora PostgreSQL. Pour la version préliminaire, vous devez créer toutes les intégrations dans l'environnement de prévisualisation de base de données Amazon RDS, dans l'est des États-Unis (Ohio) (us-east-2). Région AWS Pour plus d’informations, consultez Utilisation des intégrations zéro ETL d’Aurora à Amazon Redshift.	28 novembre 2023
Amazon Aurora PostgreSQL prend en charge le transfert d’écriture dans la base de données globale	Vous pouvez désormais activer le transfert d’écriture sur des clusters secondaires dans une base de données globale basée sur Aurora PostgreSQL. Pour plus d’informations, consultez Utilisation du transfert d’écriture dans une base de données globale Aurora PostgreSQL.	9 novembre 2023

Prise en charge d’Aurora PostgreSQL pour Optimized Reads	Vous pouvez accélérer le traitement des requêtes pour Aurora PostgreSQL avec Aurora Optimized Reads. Pour plus d’informations, consultez Amélioration des performances des requêtes pour Aurora PostgreSQL avec Aurora Optimized Reads.	8 novembre 2023
Amazon RDS exporte les métriques Performance Insights vers Amazon CloudWatch	Performance Insights vous permet d'exporter les tableaux de bord de métriques préconfigurés ou personnalisés vers Amazon. CloudWatch Les tableaux de bord des métriques exportés peuvent être consultés dans la CloudWatch console. Vous pouvez également exporter un widget métrique Performance Insights sélectionné et consulter les données des métriques dans la CloudWatch console. Pour plus d'informations, consultez Exporter les métriques Performance Insights vers CloudWatch.	8 novembre 2023

Disponibilité générale des intégrations zéro ETL d'Aurora MySQL à Amazon Redshift	Les intégrations zéro ETL à Amazon Redshift sont désormais généralement disponibles pour Aurora MySQL. Pour plus d'informations, consultez Utilisation des intégrations zéro ETL d'Aurora à Amazon Redshift.	7 novembre 2023
Support d'Aurora PostgreSQL pour les déploiements RDS Blue/Green	Vous pouvez désormais créer un blue/green déploiement à partir d'un cluster de base de données Aurora PostgreSQL. Pour plus d'informations, consultez Utilisation des Blue/Green déploiements Amazon RDS pour les mises à jour de bases de données.	26 octobre 2023

Aurora MySQL prend en charge le chiffrement côté serveur avec AWS KMS keys (SSE-KMS)	Dans Aurora MySQL version 3.05 et versions ultérieures, vous pouvez utiliser SSE-KMS, y compris Clés gérées par AWS les clés gérées par le client, pour le chiffrement côté serveur des données que vous chargez ou enregistrez sur Amazon S3. Pour plus d'informations, consultez Chargement de données dans un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte d'un compartiment Amazon S3 et Enregistrement de données d'un cluster de bases de données Amazon Aurora MySQL à partir de fichiers texte d'un compartiment Amazon S3.	25 octobre 2023

Les optimisations d'Aurora MySQL réduisent le temps de redémarrage de la base de données	Dans Aurora MySQL 3.05 et versions ultérieures, nous avons introduit des optimisations qui réduisent le temps de redémarrage de la base de données. Ces optimisations permettent de réduire la durée d'indisponibilité de 65 % et atténuent les perturbations engendrées sur les charges de travail de votre base de données après un redémarrage. Pour plus d'informations, consultez Optimisations visant à réduire le temps de redémarrage de la base de données.	25 octobre 2023
Mise à jour des politiques AWS gérées	Les politiques gérées par `AmazonRDSPerformanceInsightsReadOnly` et `AmazonRDSPerformanceInsightsFullAccess` incluent désormais `Sid` (ID d'instruction) comme identifiant dans l'instruction de la politique. Pour plus d'informations, consultez Mises à jour Amazon RDS des politiques gérées par AWS.	23 octobre 2023

Amazon RDS publie les contre-métriques Performance Insights sur Amazon CloudWatch	La fonction mathématique des métriques DB_PERF_INSIGHTS de la CloudWatch console vous permet d'interroger Amazon RDS pour obtenir les indicateurs de compteur Performance Insights. Pour plus d'informations, consultez Création d'CloudWatch alarmes pour surveiller Amazon Aurora.	20 septembre 2023
Amazon Aurora prend en charge point-in-time la restauration (PITR) avec AWS Backup	Vous pouvez désormais gérer les sauvegardes automatisées (continues) d'Aurora AWS Backup et les restaurer à des dates spécifiées à partir de celles-ci. Pour plus d'informations, consultez Restauration d'un cluster de bases de données à un instant spécifié en utilisant AWS Backup.	7 septembre 2023
Support étendu Amazon RDS	Amazon Aurora annonce la possibilité de continuer à exécuter les versions majeures de moteur Aurora MySQL et Aurora PostgreSQL dans vos instances de base de données après la date de fin du support standard Aurora. Pour plus d'informations, consultez Utilisation du support étendu Amazon RDS.	1er septembre 2023

Amazon Aurora MySQL étend la prise en charge de Percona XtraBackup	Vous pouvez désormais effectuer des migration s physiques de bases de données MySQL 8.0 vers des clusters de bases de données Aurora MySQL version 3. Pour plus d'informations, consultez Migration physique depuis MySQL à l'aide de Percona XtraBackup et Amazon S3.	24 août 2023
La base de données globale Aurora prend en charge le basculement global de la base de données	La base de données globale Aurora prend désormais en charge le basculeme nt global géré, ce qui vous permet de récupérer plus facilement après une véritable catastrophe régionale ou une interruption complète du niveau de service. Pour en savoir plus sur cette fonctionnalité, consultez Réalisation de basculeme nts gérés pour les bases de données globales Aurora. La fonctionnalité précédemm ent appelée « basculement planifié géré » est désormais appelée « bascule ». Pour obtenir des informations sur les bascules, consultez Réalisation de bascules pour les bases de données Amazon Aurora Global Database.	21 août 2023

Mise à jour des autorisations de politique AWS gérées	La politique gérée `AmazonRDSFullAccess` dispose de nouvelles autorisations qui vous permettent de générer, d'afficher et de supprimer le rapport d'analyse des performances pendant une période donnée. Pour plus d'informations, consultez les mises à jour des politiques AWS gérées par Amazon RDS.	17 août 2023
Mise à jour des autorisations de politique AWS gérées	L'ajout de nouvelles autorisations à la politique gérée `AmazonRDSPerformanceInsightsReadOnly` et l'ajout d'une nouvelle politique gérée `AmazonRDSPerformanceInsightsFullAccess` vous permet de générer un rapport d'analyse de la charge de base de données pour une période donnée. Pour plus d'informations, consultez les mises à jour des politiques AWS gérées par Amazon RDS.	16 août 2023

Amazon RDS prend en charge l’analyse du temps de chargement de la base de données avec l’analyse des performances	L’analyse des performances vous permet de créer des rapports d’analyse des performances pour une période spécifique. Ce rapport fournit les informations identifiées et des recommandations pour résoudre les problèmes de performances. Pour plus d’informations, consultez Analyse de la charge de la base de données pour une période donnée (langue française non garantie).	16 août 2023
Amazon Aurora prend en charge la conservation des sauvegardes automatiques pour les clusters de bases de données	Vous pouvez désormais conserver les sauvegardes automatiques des clusters Aurora supprimés et les restaurer à un instant précis dans le passé. Pour plus d’informations, consultez Conservation des sauvegardes automatiques.	4 août 2023
Amazon Aurora est disponible dans la région Israël (Tel Aviv)	Amazon Aurora est désormais disponible dans la région Israël (Tel Aviv). Pour plus d’informations, consultez Régions et zones de disponibilité.	1er août 2023

Amazon Aurora MySQL prend en charge le transfert d'écriture local (intracluster)	Vous pouvez désormais transférer les opérations d'écriture d'une instance de base de données de lecteur vers une instance de base de données d'enregistreur au sein d'un cluster de bases de données Aurora MySQL. Pour plus d'informations, consultez Utilisation du transfert d'écriture dans un cluster de bases de données Amazon Aurora MySQL.	31 juillet 2023
Amazon Aurora prend Aurora Serverless v2 en charge un Région AWS	Vous pouvez désormais créer des clusters de Aurora Serverless v2 bases de données dans la région Asie-Pacifique (Melbourne) Région AWS. Pour plus d'informations sur Aurora Serverless v2, consultez Utilisation d'Aurora Serverless v2.	28 juin 2023

Amazon Aurora présente les intégrations zéro ETL à Amazon Redshift (version préliminaire)	Les intégrations zéro ETL fournissent une solution entièrement gérée pour rendre les données transactionnelles disponibles dans Amazon Redshift quelques secondes après leur écriture dans un cluster de bases de données Aurora MySQL. Pour plus d'informations, consultez Utilisation des intégrations zéro ETL d'Aurora à Amazon Redshift.	28 juin 2023
Amazon RDS fournit une vue combinée des CloudWatch statistiques et des statistiques sur les performances dans le tableau de bord Performance Insights	Amazon RDS fournit désormais une vue consolidé e des CloudWatch statistiques et indicateurs de performan ce dans le tableau de bord Performance Insights. Pour plus d'informations, consultez Affichage des métriques combinées avec le tableau de bord Performance Insights.	24 mai 2023
Amazon Aurora prend en charge les classes d'instance db.r7g	Vous pouvez désormais utiliser les classes d'instance db.r7g pour créer des clusters de bases de données Aurora. Pour plus d'informations, consultez Classes d'instance de base de données Aurora.	11 mai 2023

Amazon Aurora prend en charge une nouvelle configuration de stockage pour le cluster de bases de données	AvecAurora I/O-Optimized, vous ne payez que pour l'utilisation et le stockage de vos clusters de base de données, sans frais supplémentaires pour les I/O opérations de lecture et d'écriture. Pour plus d'informations, consultez Configurations du stockage pour les clusters de bases de données Amazon Aurora.	11 mai 2023
Amazon Aurora prend en charge Aurora Serverless v2 en outre Régions AWS	Vous pouvez désormais créer des Aurora Serverless v2 clusters de base de données dans les pays suivants Régions AWS : Asie-Pacifique (Hyderabad), Europe (Espagne), Europe (Zurich) et Moyen-Orient (Émirats arabes unis). Pour plus d'informations sur Aurora Serverless v2, consultez Utilisation d'Aurora Serverless v2.	4 mai 2023
Aurora Serverless v1 prend en charge la conversion en provisionné	Vous pouvez convertir un cluster de bases de données Aurora Serverless v1 directement en un cluster de bases de données provisionné. Pour plus d'informations, consultez Conversion d'un cluster de bases de données Aurora Serverless v1 en cluster provisionné.	27 avril 2023

Aurora Serverless v1 prend en charge Amazon Aurora PostgreSQL version 13	Vous pouvez désormais créer des clusters de bases de données Aurora Serverless v1 qui exécutent Aurora PostgreSQL version 13. Pour de plus amples informations, veuillez consulter Aurora Serverless v1.	27 avril 2023
Support d'Amazon Aurora pour AWS Secrets Manager les régions de Chine	Amazon Aurora prend en charge Secrets Manager dans les régions Chine (Pékin) et Chine (Ningxia). Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager.	20 avril 2023
Amazon Aurora prend en charge la publication d'événements avec des balises pour ses abonnés d'une rubrique	Les notifications d'événements Amazon Aurora envoyées à Amazon Simple Notification Service (Amazon SNS) ou EventBridge Amazon contiennent désormais des balises d'événement dans le corps du message. Ces balises fournissent des données sur la ressource affectée par l'événement de service. Pour plus d'informations, consultez Amazon RDS event notification tags and attributes (Balises et attributs de notification d'événement Amazon RDS).	17 avril 2023

Mise à jour des autorisations de rôle lié à un service IAM	Les `AmazonRDSReadOnlyAccess` politiques `AmazonRDSFullAccess` et accords accordent désormais des autorisations supplémentaires pour permettre l'affichage des résultats d'Amazon DevOps Guru dans la console RDS. Pour plus d'informations, consultez les mises à jour des politiques AWS gérées par Amazon RDS.	30 mars 2023
Amazon Aurora prend en charge des bases de données globales dans la région Asie-Pacifique (Melbourne)	Vous pouvez désormais créer des bases de données globales Aurora dans la région Asie-Pacifique (Melbourne). Pour plus d'informations sur les bases de données globales Aurora, consultez Utilisation de bases de données Amazon Aurora globales.	22 mars 2023
Mise à jour des autorisations de politique AWS gérées	Les `AmazonRDSReadOnlyAccess` politiques `AmazonRDSFullAccess` et accords accordent désormais des autorisations supplémentaires à Amazon CloudWatch. Pour plus d'informations, consultez les mises à jour des politiques AWS gérées par Amazon RDS.	16 mars 2023

RDS Proxy est disponible dans les régions de Chine	Le proxy RDS est désormais disponible dans les régions de Chine (Pékin) et de Chine (Ningxia). Pour plus d'informations sur RDS Proxy, consultez Utilisation d'Amazon RDS Proxy.	15 mars 2023
Amazon Aurora prend en charge Aurora Serverless v2 dans les régions de Chine	Aurora Serverless v2 est désormais disponible dans les régions de Chine (Pékin) et de Chine (Ningxia). Pour de plus amples informations, veuillez consulter Aurora Serverless v2.	15 mars 2023
RDS Proxy est disponible dans la région Asie-Pacifique (Jakarta)	Le proxy RDS est désormais disponible dans la région Asie-Pacifique (Jakarta). Pour plus d'informations sur RDS Proxy, consultez Utilisation d'Amazon RDS Proxy.	8 mars 2023
Amazon Aurora MySQL prend en charge l'authentification Kerberos	Vous pouvez désormais utiliser l'authentification Kerberos pour authentifier les utilisateurs qui se connecten t à vos clusters de bases de données Aurora MySQL. Pour plus d'informations, consultez Using Kerberos authentication for Aurora MySQL (Utilisation de l'authentification Kerberos pour Aurora MySQL).	8 mars 2023

Amazon Aurora prend également en charge les bases de données mondiales Régions AWS	Vous pouvez désormais créer des bases de données globales Aurora dans les régions suivants : Afrique (Le Cap), Asie-Pacifique (Hong Kong), Asie-Paci fique (Hyderabad), Asie-Pacifique (Jakarta), Europe (Milan), Europe (Espagne) , Europe (Zurich), Moyen-Orient (Bahreïn) et Moyen-Orient (EAU). Pour plus d'informations sur les bases de données globales Aurora, consultez Utilisation de bases de données Amazon Aurora globales.	06 mars 2023
Amazon Aurora prend en charge la copie d'instant anés de clusters de bases de données dans des applicati ons supplémentaires Régions AWS	Vous pouvez désormais copier des instantanés de clusters de bases de données dans les régions suivantes : Afrique (Le Cap), Asie-Paci fique (Hong Kong), Asie-Paci fique (Hyderabad), Asie-Paci fique (Jakarta), Asie-Paci fique (Melbourne), Europe (Milan), Europe (Espagne), Europe (Zurich), Moyen-Ori ent (Bahreïn) et Moyen-Ori ent (EAU). Pour plus d'informa tions sur la copie d'instant anés de clusters de bases de données, consultez Copie d'un instantané de cluster de bases de données.	06 mars 2023

Amazon DevOps Guru pour RDS permet d'obtenir des informations proactives	Amazon DevOps Guru for RDS publie des informations proactives contenant des recommandations pour vous aider à résoudre les problèmes liés à vos bases de données Aurora avant qu'ils ne se produisent. Pour plus d'informations, consultez Comment fonctionne DevOps Guru for RDS.	28 février 2023
Amazon Aurora MySQL version 1 est obsolète	Aurora MySQL version 1 (compatible avec MySQL 5.6) est désormais obsolète. Pour plus d'informations, consultez Durée de disponibilité des versions majeures d'Amazon Aurora	28 février 2023
Aurora Serverless v1 permet de définir la fenêtre de maintenance du cluster de bases de données	Vous pouvez maintenant définir la fenêtre de maintenance pour des clusters de bases de données Aurora Serverless v1. Pour plus d'informations, consultez Ajustement du créneau de maintenance préféré pour un cluster de bases de données.	27 février 2023
Amazon Aurora prend en charge le flux d'activités de base de données dans les régions Asie-Pacifique (Hyderabad), Europe (Espagne) et Moyen-Orient (EAU).	Pour plus d'informations, consultez Flux d'activité de base de données.	27 janvier 2023

Amazon Aurora est disponible dans la région Asie-Pacifique (Melbourne)	Amazon Aurora est désormais disponible dans la région Asie-Pacifique (Melbourne). Pour plus d'informations, consultez Régions et zones de disponibilité.	23 janvier 2023
Spécification de l'autorité de certification (CA) lors de création d'un cluster de bases de données	Vous pouvez désormais spécifier l'autorité de certification à utiliser pour le certificat de serveur d'un cluster de bases de données lors de la création d'un cluster de bases de données. Pour plus d'informations, consultez Autorités de certification.	5 janvier 2023
Prise en charge d'Aurora MySQL 3.* pour le retour sur trace	Les versions Aurora MySQL 3.* offrent désormais un moyen rapide de récupérer après des erreurs utilisateur, comme la suppression non souhaitée d'une table ou d'une ligne. Le retour sur trace vous permet de déplacer votre base de données vers un point précédent dans le temps sans nécessiter la restauration à partir d'une sauvegarde. Il s'exécute en quelques secondes, même pour les bases de données volumineuses. Pour plus d'informations, consultez Retour sur trace d'un cluster de bases de données Aurora.	4 janvier 2023

Utilisez les Blue/Green déploiements Amazon RDS disponibles en supplément Régions AWS	La fonctionnalité Blue/Green Déploiements est désormais disponible dans les régions de Chine (Pékin) et de Chine (Ningxia). Pour plus d'informations, consultez Utilisation des Blue/Green déploiements Amazon RDS pour les mises à jour de bases de données.	22 décembre 2022
Mise à jour des autorisations de rôle lié à un service IAM	La RDSService RolePolicy politique d'Amazon accorde désormais des autorisations supplémentaires à AWS Secrets Manager. Pour plus d'informations, consultez les mises à jour des politiques AWS gérées par Amazon RDS.	22 décembre 2022
Amazon Aurora s'intègre à la gestion AWS Secrets Manager des mots de passe	Aurora peut gérer le mot de passe d'utilisateur principal dans Secrets Manager pour un cluster de bases de données. Pour plus d'informations, consultez Gestion des mots de passe avec Amazon Aurora et AWS Secrets Manager.	22 décembre 2022
Amazon Aurora prend en charge Aurora Serverless v2 en outre Régions AWS	Aurora Serverless v2 est maintenant disponible dans les régions Afrique (Le Cap) et Europe (Milan). Pour de plus amples informations, veuillez consulter Aurora Serverless v2.	21 décembre 2022

Aurora PostgreSQL prend en charge le proxy RDS avec PostgreSQL 14	Vous pouvez désormais créer un proxy RDS avec un cluster de bases de données Aurora PostgreSQL 14. Pour plus d'informations sur RDS Proxy, consultez Utilisation d'Amazon RDS Proxy.	13 décembre 2022
Amazon Aurora vous avertit des anomalies récemment détectées par Amazon DevOps Guru	La page de détails de la base de données de la console vous avertit à la fois des anomalies actuelles et des anomalies survenues au cours des dernières 24 heures. Pour plus d'informations, consultez Comment fonctionne DevOps Guru for RDS.	13 décembre 2022
Le proxy Amazon RDS prend en charge les bases de données globales	Vous pouvez désormais utiliser le proxy RDS avec les bases de données globales Aurora. Pour plus d'informations, consultez Using RDS Proxy with Aurora global databases (Utilisation du proxy RDS avec des bases de données globales Aurora).	7 décembre 2022

Les clusters de bases de données Aurora PostgreSQL prennent en charge le kit Trusted Language Extensions pour PostgreSQL	Trusted Language Extensions for PostgreSQL est un kit de développement open source qui vous permet de créer des extensions PostgreSQL à hautes performances et de les exécuter en toute sécurité sur votre cluster de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez Working with Trusted Language Extensions for PostgreSQL (Utilisation de Trusted Language Extensions pour PostgreSQL).	30 novembre 2022

Amazon GuardDuty RDS Protection surveille les menaces d'accès	Lorsque vous activez la protection GuardDuty RDS, vous GuardDuty consommez les événements de connexion RDS de vos bases de données Aurora, surveillez ces événements et établissez un profil pour détecter d'éventuelles menaces internes ou externes. Lorsque GuardDuty RDS Protection détecte une menace potentielle, GuardDuty génère une nouvelle découverte contenant des détails sur la base de données potentiellement compromise. Pour plus d'informations, consultez la section Surveillance des menaces à l'aide de la protection GuardDuty RDS.	30 novembre 2022

Utiliser les Blue/Green déploiements Amazon RDS pour les mises à jour des bases de données	Vous pouvez apporter des modifications à un cluster de bases de données dans un environnement intermédiaire et tester les modifications sans affecter votre cluster de bases de données de production. Lorsque vous êtes prêt, vous pouvez promouvoir l'environ nement intermédiaire comme nouvel environnement de production, avec une durée d'indisponibilité minimale. Pour plus d'informations, consultez Utilisation des Blue/Green déploiements Amazon RDS pour les mises à jour de bases de données.	27 novembre 2022
Amazon Aurora est disponible dans la région Asie-Pacifique (Hyderabad)	Amazon Aurora est désormais disponible dans la région Asie-Pacifique (Hyderabad). Pour plus d'informations, consultez Régions et zones de disponibi lité.	22 novembre 2022
Amazon Aurora est disponibl e dans la région Europe (Espagne)	Amazon Aurora est désormais disponible dans la région Europe (Espagne). Pour plus d'informations, consultez Régions et zones de disponibi lité.	16 novembre 2022

Amazon Aurora est disponible dans la région Europe (Zurich)	Amazon Aurora est désormais disponible dans la région Europe (Zurich). Pour plus d'informations, consultez Régions et zones de disponibilité.	9 novembre 2022
Amazon Aurora prend en charge l'exportation de données vers Amazon S3 à partir de clusters de bases de données	Vous pouvez désormais exporter les données du cluster Aurora directement vers S3, sans avoir à créer d'instantané au préalable . Pour plus d'informations, consultez Exporting DB cluster data to Amazon S3 (Exportation de données de cluster de bases de données vers Amazon S3).	27 octobre 2022
Amazon Aurora MySQL prend en charge des exportations plus rapides vers Amazon S3	Vous pouvez désormais bénéficier de performances jusqu'à 10 fois plus rapides lors de l'exportation de données d'instantanés de cluster de bases de données vers S3 pour les clusters Aurora MySQL compatibles MySQL 5.7 et 8.0. Pour plus d'informations, consultez Exportation de données d'instantanés de cluster de bases de données vers Amazon S3.	20 octobre 2022

Amazon Aurora prend en charge la configuration automatique de la connectivité entre un cluster de base de données Aurora et une EC2 instance	Vous pouvez utiliser le AWS Management Console pour configurer la connectivité entre un cluster de base de données Aurora existant et une EC2 instance. Pour plus d'informations, consultez Connexion automatique d'une EC2 instance et d'un cluster de base de données Aurora.	14 octobre 2022
AWS Le pilote JDBC pour PostgreSQL est généralement disponible	Le pilote AWS JDBC pour PostgreSQL est un pilote client conçu pour Aurora PostgreSQL. Le pilote AWS JDBC pour PostgreSQL est désormais disponible pour tous. Pour plus d'informations, consultez Connexion avec le pilote AWS JDBC pour PostgreSQL.	6 octobre 2022
Amazon Aurora prend en charge la mise à niveau en place pour Aurora MySQL compatible avec MySQL 5.7.	Vous pouvez effectuer une mise à niveau sur place pour convertir un cluster Aurora MySQL compatible avec MySQL 5.7 existant en un cluster Aurora MySQL compatible avec MySQL 8.0. Pour plus d'informations, consultez Mise à niveau d'Aurora MySQL 2.x vers 3.x.	26 septembre 2022

Performance Insights affiche les 25 principales requêtes SQL	Dans le tableau de bord Performance Insights, l'onglet SQL maximum présente les 25 requêtes SQL qui contribuent le plus à la charge de la base de données. Pour plus d'informations, consultez Présentation de l'onglet SQL maximum.	13 septembre 2022
Aurora MySQL prend en charge une nouvelle classe d'instance de base de données.	Vous pouvez désormais utiliser la classe d'instance de base de données db.r6i pour les clusters de bases de données Aurora MySQL. Pour plus d'informations, consultez Classes d'instance de base de données.	13 septembre 2022
Amazon Aurora est disponible dans la région Moyen-Orient (EAU)	Amazon Aurora est désormais disponible dans la région Moyen-Orient (EAU). Pour plus d'informations, consultez Régions et zones de disponibilité.	30 août 2022

Amazon Aurora prend en charge la configuration automatique de la connectivité avec une EC2 instance	Lorsque vous créez un cluster de base de données Aurora, vous pouvez l'utiliser AWS Management Console pour configurer la connectivité entre une instance Amazon Elastic Compute Cloud et le nouveau cluster de base de données. Pour plus d'informations, consultez Configurer la connectivité réseau automatique avec une EC2 instance.	22 août 2022
Amazon Aurora prend en charge le mode double pile	Les clusters de base de données peuvent désormais fonctionner en mode double pile. En mode double pile, les ressources peuvent communiquer avec le cluster de base de données IPv4 via ou IPv6 les deux. Pour plus d'informations, consultez Amazon Aurora IP addressing (Adressage IP d'Amazon Aurora).	17 août 2022

Amazon Aurora prend en charge la mise à niveau sur place pour les systèmes Aurora Serverless v1 compatibles avec PostgreSQL	Vous pouvez effectuer une mise à niveau sur place pour un cluster Aurora Serverless v1 compatible avec PostgreSQL 10 afin de transformer un cluster existant en un cluster Aurora Serverless v1 compatible avec PostgreSQL 11. Pour connaître la procédure de mise à niveau sur place, consultez Modification d'un cluster de bases de données Aurora Serverless v1.	8 août 2022
Performance Insights prend en charge la région Asie-Pacifique (Jakarta)	Auparavant, vous ne pouviez pas utiliser Performance Insights dans la région Asie-Pacifique (Jakarta). Cette restriction a été supprimée. Pour plus d'informations, consultez Région AWS support for Performance Insights (Prise en charge des Région AWS pour Performance Insights).	21 juillet 2022
Amazon Aurora prend en charge une nouvelle classe d'instance de base de données.	Vous pouvez désormais utiliser la classe d'instance de base de données db.r6i pour les clusters de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez Classes d'instance de base de données.	14 juillet 2022

RDS Performance Insights prend en charge des périodes de conservation supplémentaires	Auparavant, Performance Insights ne proposait que deux périodes de conservation : 7 jours (par défaut) ou 2 ans (731 jours). Désormais, si vous avez besoin de conserver vos données de performance pendant plus de 7 jours, vous pouvez spécifier de 1 à 24 mois. Pour plus d'informations, consultez Pricing and data retention for Performance Insights (Tarification et conservation des données pour Performance Insights).	1er juillet 2022
Amazon Aurora prend en charge la mise à niveau en place pour les systèmes Aurora Serverless v1 compatibles avec MySQL	Vous pouvez effectuer une mise à niveau sur place pour un cluster Aurora Serverless v1 compatible avec MySQL 5.6 afin de convertir un cluster existant en cluster Aurora Serverless v1 compatible avec MySQL 5.7. Pour connaître la procédure de mise à niveau sur place, consultez Modification d'un cluster de bases de données Aurora Serverless v1.	16 juin 2022

Aurora prend en charge l'activation d'Amazon DevOps Guru dans la console RDS	Vous pouvez activer la couverture DevOps Guru pour vos bases de données Aurora depuis la console RDS. Pour plus d'informations, consultez Configuration de DevOps Guru pour RDS.	9 juin 2022
Amazon Aurora prend en charge la publication d’événements dans des rubriques Amazon SNS chiffrées	Amazon Aurora peut désormais publier des événements dans des rubriques Amazon Simple Notification Service (Amazon SNS) où le chiffrement côté serveur (SSE) est activé, afin de renforcer la protection des événements contenant des données sensibles. Pour plus d’informations, consultez Abonnement à la notification d’événement Amazon RDS.	1er juin 2022
Amazon RDS publie des statistiques d'utilisation sur Amazon CloudWatch	L'espace de `AWS/Usage` noms d'Amazon CloudWatch inclut les mesures d'utilisation au niveau du compte pour vos quotas de service Amazon RDS. Pour plus d'informations, consultez les métriques CloudWatch d'utilisation d'Amazon pour Amazon Aurora.	28 avril 2022

Ensemble de résultats de l’API de données au format JSON	Un paramètre facultatif de la fonction `ExecuteStatement` permet de renvoyer l’ensemble des résultats de la requête sous forme de chaîne au format JSON. L’ensemble de résultats JSON est simple et pratique à transformer en une structure de données dans le langage de votre application. Pour plus d’informations, voir Processing query results in JSON format (Traitement des résultats de la requête au format JSON).	27 avril 2022
Amazon Aurora Serverless v2 est désormais globalement disponible	Amazon Aurora Serverless v2 est globalement disponible pour tous les utilisateurs. Pour plus d’informations, consultez Utilisation de Aurora Serverless v2.	21 avril 2022

Aurora MySQL prend en charge les suites de chiffrement configurables	Avec Aurora MySQL, vous pouvez désormais utiliser des suites de chiffrement configurables pour contrôler le chiffrement des connexions que votre serveur de base de données accepte. Pour plus d'informations, consultez Configuring cipher suites for connections to Aurora MySQL DB clusters (Configuration des suites de chiffrement pour les connexions aux clusters de bases de données Aurora MySQL).	15 avril 2022
Aurora PostgreSQL prend en charge RDS Proxy avec PostgreSQL 13	Vous pouvez désormais créer un proxy RDS avec un cluster de bases de données Aurora PostgreSQL 13. Pour plus d'informations sur RDS Proxy, consultez Utilisation d'Amazon RDS Proxy.	4 avril 2022
Notes de mise à jour d'Aurora PostgreSQL	Il existe maintenant un guide séparé pour les notes de mise à jour d'Amazon Aurora PostgreSQL. Pour plus d'informations, consultez Notes de mise à jour d'Aurora PostgreSQL.	22 mars 2022
Notes de mise à jour d'Aurora MySQL	Il existe maintenant un guide séparé pour les notes de mise à jour d'Amazon Aurora MySQL. Pour plus d'informations, consultez Notes de mise à jour d'Aurora MySQL.	22 mars 2022

Aurora PostgreSQL prend en charge les mises à niveau vers de multiples versions majeures	Vous pouvez désormais effectuer des mises à niveau de version des clusters de bases de données Aurora PostgreSQL vers de multiples versions majeures. Pour plus d'informations, consultez Comment effectuer une mise à niveau de version majeure.	4 mars 2022
Aurora PostgreSQL prend en charge les suites de chiffrement configurables	Avec Aurora PostgreSQL versions 11.8 et ultérieures, vous pouvez désormais utiliser des suites de chiffrement configurables pour contrôler le chiffrement de connexion accepté par votre serveur de base de données. Pour obtenir des informations sur l'utilisation des suites de chiffrement configurables avec Aurora PostgreSQL, consultez Configuration des suites de chiffrement pour les connexions aux clusters de bases de données Aurora PostgreSQL.	4 mars 2022

AWS Le pilote JDBC pour MySQL est généralement disponible	Le pilote AWS JDBC pour MySQL est un pilote client conçu pour la haute disponibilité d'Aurora MySQL. Le pilote AWS JDBC pour MySQL est désormais disponible pour tous. Pour en savoir plus, consultez Connexion avec le pilote JDBC Amazon Web Services pour MySQL.	2 mars 2022
Aurora PostgreSQL 13.5 prend en charge Babelfish pour Aurora PostgreSQL 1.1.0	Aurora PostgreSQL 13.5 prend en charge Babelfish 1.1.0. Pour obtenir la liste des nouvelles fonctions, consultez 13.5. Pour une liste des fonctionnalités prises en charge dans chaque version de Babelfish, consultez Supported functionality in Babelfish by version (Fonctionnalités prises en charge dans Babelfish par version). Pour obtenir des informations sur l'utilisation, consultez Working with Babelfish pour Aurora PostgreSQL (Utilisation de Babelfish pour Aurora PostgreSQL).	28 février 2022
Amazon Aurora prend en charge Database Activity Streams (Flux d'activités de base de données) dans la Région Asie-Pacifique (Jakarta)	Pour plus d'informations, consultez la section Support Régions AWS pour les flux d'activité des bases de données.	16 février 2022

Performance Insights prend en charge les nouvelles API	Performance Insights prend désormais en charge les opérations API suivantes : `GetResourceMetadata`, `ListAvailableResourceDimensions` et `ListAvailableResourceMetrics`. Pour plus d'informations, consultez Récupération de métriques avec l'API Performance Insights dans ce manuel ainsi que la Référence d'API d'analyse des performances d'Amazon RDS.	12 janvier 2022
Amazon RDS Proxy prend en charge les événements	Le proxy RDS génère désormais des événements auxquels vous pouvez vous abonner et consulter dans CloudWatch Events ou configurer pour les envoyer à Amazon EventBridge. Pour plus d'informations, consultez Utilisation des événements RDS Proxy.	11 janvier 2022

Proxy RDS disponible en supplément Régions AWS	Le proxy RDS est désormais disponible dans les Régions suivantes : Afrique (Le Cap), Asie-Pacifique (Hong Kong), Asie-Pacifique (Osaka), Europe (Milan), Europe (Paris), Europe (Stockholm), Moyen-Orient (Bahreïn) et Amérique du Sud (São Paulo). Pour plus d'informations sur RDS Proxy, consultez Utilisation d'Amazon RDS Proxy.	5 janvier 2022
Amazon Aurora est disponible dans la région Asie-Pacifique (Jakarta)	Amazon Aurora est désormais disponible dans la région Asie-Pacifique (Jakarta). Pour plus d'informations, consultez Régions et zones de disponibilité.	13 décembre 2021

DevOpsGuru for Amazon RDS fournit des informations détaillées et des recommandations pour Amazon Aurora	DevOpsGuru for RDS analyse les données relatives aux performances grâce à Performance Insights. À l'aide de ces données, le service analyse les performances de vos instances de base de données Amazon Aurora et peut vous aider à résoudre les problèmes de performances. Pour en savoir plus, consultez la section Analyse des anomalies de performances avec DevOps Guru for RDS dans ce guide et consultez la section Présentation de DevOps Guru for RDS dans le guide de l'utilisateur Amazon DevOps Guru.	1er décembre 2021
Aurora PostgreSQL prend en charge RDS Proxy avec PostgreSQL 12	Vous pouvez désormais créer un proxy RDS avec un cluster de bases de données Aurora PostgreSQL 12. Pour plus d'informations sur RDS Proxy, consultez Utilisation d'Amazon RDS Proxy.	22 novembre 2021
Aurora prend en charge les classes d'instances AWS Graviton2 pour les flux d'activité des bases de données	Vous pouvez utiliser des flux d'activité de base de données avec la classe d'instance db.r6g pour Aurora MySQL et Aurora PostgreSQL. Pour plus d'informations, consultez Classes d'instance de bases de données prises en charge.	03 novembre 2021

Support d'Amazon Aurora pour les comptes multiples AWS KMS keys	Vous pouvez utiliser une clé KMS d'une autre Compte AWS pour le chiffrement lorsque vous exportez des instantanés de base de données vers Amazon S3. Pour plus d'informations, consultez Exportation de données d'instantanés de bases de données vers Amazon S3.	3 novembre 2021
Amazon Aurora prend en charge Babelfish pour Aurora PostgreSQL	Babelfish pour Aurora PostgreSQL étend l'Édition compatible avec PostgreSQL de votre base de données Amazon Aurora et permet d'accepter les connexions de base de données à partir de clients Microsoft SQL Server. Pour plus d'informations, consultez Utilisation de Babelfish pour Aurora PostgreSQL.	28 octobre 2021
Aurora Serverless v1 peut exiger SSL pour les connexions	Les paramètres de cluster Aurora `force_ssl` pour PostgreSQL et `require_secure_transport` pour MySQL sont désormais pris en charge pour Aurora Serverless version 1. Pour plus d'informations, consultez la section Utilisation TLS/SSL avec la Aurora Serverless version 1.	26 octobre 2021

Amazon Aurora prend également en charge Performance Insights Régions AWS	Performance Insights est disponible dans les régions suivantes : Moyen-Orient (Bahreïn), Afrique (Le Cap), Europe (Milan) et Asie-Pacifique (Osaka). Pour plus d'informations, consultez Région AWS support for Performance Insights (Prise en charge des Région AWS pour Performance Insights).	5 octobre 2021
Délai d'attente de scalabilité automatique configurable pour Aurora Serverless v1	Vous pouvez choisir combien de temps Aurora Serverless v1 attend pour trouver un point de scalabilité automatique. Si aucun point de scalabilité automatique n'est trouvé pendant cette période, Aurora Serverless v1 annule l'événement de mise à l'échelle ou force le changement de capacité, en fonction de l'action de délai d'attente que vous avez sélectionnée. Pour plus d'informations, consultez Autoscaling pour Aurora Serverless v1.	10 septembre 2021

Aurora prend en charge les classes d'instance X2g et T4g	Aurora MySQL et Aurora PostgreSQL peuvent désormais utiliser des classes d'instance X2g et T4g. Les classes d'instance que vous pouvez utiliser dépendent de la version d'Aurora MySQL ou d'Aurora PostgreSQL. Pour plus d'informations sur les types d'instances pris en charge, consultez Classes d'instance de base de données.	10 septembre 2021
Amazon RDS prend en charge RDS Proxy dans un VPC partagé	Vous pouvez maintenant créer un proxy RDS dans un cloud privé virtuel (VPC) partagé. Pour plus d'informations sur le proxy RDS, consultez « Gestion des connexions avec le proxy Amazon RDS » dans le Guide de l'utilisateur Amazon RDS ou le Guide de l'utilisateur Aurora.	6 août 2021
Page sur la politique de version d'Aurora	Le Guide de l'utilisateur Amazon Aurora comprend désormais une section contenant des informations générales sur les versions d'Aurora et les stratégies associées. Pour plus d'informations, consultez Versions d'Amazon Aurora.	14 juillet 2021

Exclure les événements de l'API de données d'un AWS CloudTrail historique	Vous pouvez exclure les événements de l'API de données d'un CloudTrail suivi. Pour plus d'informations, voir Exclure les événements de l'API de données d' AWS CloudTrail un historique.	2 juillet 2021
Amazon Aurora Édition compatible avec PostgreSQL prend en charge des extensions supplémentaires	Les nouvelles extensions prises en charge sont pg_bigm, pg_cron, pg_partman et pg_proctab. Pour plus d'informations, consultez Versions d'extension pour Amazon Aurora Édition compatible avec PostgreSQL.	17 juin 2021
Clonage pour clusters Aurora Serverless	Vous pouvez désormais créer des clusters clonés Aurora Serverless. Pour plus d'informations sur le clonage, consultez Clonage d'un volume pour un cluster de bases de données Aurora.	16 juin 2021
Les bases de données globales Aurora sont désormais disponibles dans les régions Chine (Pékin) et Chine (Ningxia)	Vous pouvez désormais créer des bases de données globales Aurora dans les régions Chine (Pékin) et Chine (Ningxia). Pour plus d'informations concernant les bases de données globales Aurora, consultez Utilisation de bases de données Amazon Aurora globales.	19 mai 2021

Prise en charge de FIPS 140-2 pour l’API de données	L'API de données prend en charge la norme fédérale de traitement de l'information 140-2 (FIPS 140-2) pour les connexions. SSL/TLS Pour plus d’informations, consultez la rubrique Disponibilité de l’API de données.	14 mai 2021
AWS Pilote JDBC pour PostgreSQL (version préliminaire)	Le pilote AWS JDBC pour PostgreSQL, désormais disponible en version préliminaire, est un pilote client conçu pour la haute disponibilité d'Aurora PostgreSQL. Pour plus d’informations, consultez la rubriqueConnexion avec le Pilote JDBC Amazon Web Services pour PostgreSQL (version préliminaire).	27 avril 2021
L'API de données est disponible en supplément Régions AWS	API de données maintenant disponible dans les régions Asie-Pacifique (Séoul) et Canada (Centre). Pour plus d’informations, consultez Disponibilité de l’API de données.	9 avril 2021

Amazon Aurora prend en charge les classes d’instance de base de données Graviton2	Vous pouvez désormais utiliser les classes d’instance de base de données Graviton2 db.r6g.x pour créer des clusters de bases de données exécutant MySQL ou PostgreSQL. Pour plus d’informations, consultez Classes d’instance de base de données.	12 mars 2021

Améliorations du point de terminaison proxy RDS	Vous pouvez créer d'autres points de terminaison associés à chaque proxy RDS. La création d'un point de terminaison dans un autre VPC permet un accès entre VPC pour le proxy. Les proxies pour les clusters Aurora MySQL peuvent également avoir des points de terminaison en lecture seule. Ces points de terminaison du lecteur se connectent aux instances de base de données de lecteurs dans les clusters et peuvent améliorer l'évolutivité et la disponibilité de la lecture pour les applications exigeantes en requêtes. Pour plus d'informations sur le proxy RDS, consultez « Gestion des connexions avec le proxy Amazon RDS » dans le Guide de l'utilisateur Amazon RDS ou le Guide de l'utilisateur Aurora.	8 mars 2021
Amazon Aurora est disponible dans la région Asie-Pacifique (Osaka)	Amazon Aurora est désormais disponible dans la région Asie-Pacifique (Osaka). Pour plus d'informations, consultez Régions et zones de disponibilité.	1er mars 2021

Aurora PostgreSQL prend en charge l'activation de l'authentification IAM et Kerberos sur le même cluster de bases de données	Aurora PostgreSQL prend désormais en charge l'activation de l'authentification IAM et de l'authentification Kerberos sur le même cluster de bases de données. Pour plus d'informations, consultez Authentification de base de données avec Amazon Aurora.	24 février 2021
La base de données globale Aurora prend désormais en charge le basculement planifié géré	La base de données globale Aurora prend désormais en charge le basculement planifié géré, ce qui vous permet de modifier plus facilement la région AWS principale de votre base de données globale Aurora. Vous ne pouvez utiliser le basculement planifié géré qu'avec des bases de données globales Aurora saines. Pour en savoir plus, consultez Reprise après sinistre et bases de données globales Amazon Aurora. Pour plus d'informations de référence, consultez `FailoverGlobalCluster`, dans le Amazon RDS API Reference.	11 février 2021

L’API de données pour Aurora Serverless prend désormais en charge plus de types de données	Avec l’API de données pour Aurora Serverless, vous pouvez désormais utiliser les types de données `UUID` et `JSON` comme entrée de votre base de données. De plus, avec l’API de données pour Aurora Serverless, vous pouvez désormais avoir une valeur de type `LONG` renvoyée de votre base de données en tant que valeur `STRING`. Pour en savoir plus, consultez Appel à l’API de données. Pour obtenir des informations de référence sur les types de données pris en charge, consultez `SqlParameter` dans le Référence API des services de données Amazon RDS.	2 février 2021
Aurora PostgreSQL prend en charge les mises à niveau de version majeure vers PostgreSQL 12	Aurora PostgreSQL vous permet désormais de mettre à niveau le moteur de base de données vers une version 12 majeure. Pour plus d’informations, consultez Mise à niveau du moteur de base de données PostgreSQL pour Aurora PostgreSQL.	28 janvier 2021

Aurora MySQL prend en charge la mise à niveau sur place	Vous pouvez mettre à niveau votre cluster Aurora MySQL 1.x vers Aurora MySQL 2.x, en préservant les instances de base de données, les points de terminaison et autres éléments du cluster d'origine . Cette technique de mise à niveau sur place évite les inconvénients de la configura tion d'un tout nouveau cluster en restaurant un instantan é. Elle évite également le surcoût de la copie de toutes vos données de table dans un nouveau cluster. Pour plus d'informations, consultez Mise à niveau de la version majeure d'un cluster de bases de données Aurora MySQL de 1.x à 2.x.	11 janvier 2021
AWS Pilote JDBC pour MySQL (version préliminaire)	Le pilote AWS JDBC pour MySQL, désormais disponibl e en version préliminaire, est un pilote client conçu pour la haute disponibilité d'Aurora MySQL. Pour en savoir plus, consultez Connexion avec le pilote JDBC Amazon Web Services pour MySQL (version préliminaire).	7 janvier 2021

Aurora prend en charge les flux d'activité de base de données sur les clusters secondaires d'une base de données globale	Vous pouvez démarrer un flux d'activité de base de données sur un cluster principal ou secondaire de Aurora PostgreSQL ou Aurora MySQL. Pour connaître les versions de moteur prises en charge, consultez la section Utilisation de bases de données Amazon Aurora globales.	22 décembre 2020
Clusters multi-maîtres avec quatre instances de base de données	Le nombre maximal d'instances de base de données dans un cluster Aurora MySQL multi-maîtres est maintenant de quatre. Auparavant, il était de deux. Pour plus d'informations, consultez Utilisation des clusters multi-maîtres Aurora.	17 décembre 2020
Aurora AWS Lambda PostgreSQL prend en charge les fonctions	Vous pouvez désormais invoquer une AWS Lambda fonction pour vos clusters de bases de données Aurora PostgreSQL. Pour en savoir plus, consultez Invocation d'une fonction Lambda à partir d'un cluster de bases de données Aurora PostgreSQL.	11 décembre 2020

Amazon Aurora prend en charge les classes d'instance de base de données Graviton2 en aperçu	Vous pouvez désormais utiliser les classes d'instance de base de données Graviton2 db.r6g.x en aperçu pour créer des clusters de bases de données exécutant MySQL ou PostgreSQL. Pour plus d'informations, consultez Classes d'instance de base de données.	11 décembre 2020
Amazon Aurora Serverless v2 est désormais disponible en aperçu.	Amazon Aurora Serverless v2 est disponible en aperçu. Pour travailler avec Amazon Aurora Serverless v2, demandez l'accès. Pour en savoir plus, consultez la page Aurora Serverless v2.	1er décembre 2020

Aurora PostgreSQL est désormais disponible pour Aurora Serverless dans davantage de Régions AWS.	Aurora PostgreSQL est désormais disponible pour Aurora Serverless dans davantage de Régions AWS. Vous pouvez désormais choisir de Aurora PostgreSQL Serverless v1 participer à la Régions AWS même offreAurora MySQL Serverless v1. Parmi les autres pays Régions AWS bénéficiant du Aurora Serverless soutien, citons l'ouest des États-Unis (Californie du Nord), l'Asie-Pacifique (Singapour), l'Asie-Pacifique (Sydney), l'Asie-Pacifique (Séoul), l'Asie-Pacifique (Mumbai), le Canada, l'Europe (centrale) (Londres) et l'Europe (Paris). Pour obtenir la liste complète des régions et des moteurs de base de données pour Aurora Serverless, consultez Régions et moteurs de base de données Aurora pris en charge pour Aurora sans serveur v1. L'API de données Amazon RDS for Aurora Serverless est également disponible dans ces mêmes Régions AWS. Pour obtenir la liste de toutes les régions prenant en charge l'API de données pour Aurora Serverless, consultez API de	24 novembre 2020

	données avec Aurora MySQL sans serveur v1.	
Amazon RDS Performance Insights introduit de nouvelles dimensions	Vous pouvez regrouper la charge de base de données en fonction des groupes de dimensions pour la base de données, l'application (PostgreSQL) et le type de séance (PostgreSQL). Amazon RDS prend également en charge les dimensions db.name, db.application.name (PostgreSQL) et db.session_type.name (PostgreSQL). Pour plus d'informations, consultez Tableau des principaux éléments de charge.	24 novembre 2020
Aurora Serverless prend en charge Aurora PostgreSQL version 10.12	Aurora PostgreSQL pour Aurora Serverless a été mis à niveau vers Aurora PostgreSQL version 10.12 dans les régions AWS où Aurora PostgreSQL pour Aurora Serverless est pris en charge. Pour plus d'informations, consultez Régions et moteurs de base de données Aurora pris en charge pour Aurora sans serveur v1.	4 novembre 2020

L’API de données prend désormais en charge l’autorisation basée sur les balises	L’API de données prend en charge l’autorisation basée sur les balises. Si vous avez étiqueté vos ressources de cluster RDS avec des balises, vous pouvez utiliser ces dernières dans vos instructions de politique pour contrôler l’accès via l’API de données. Pour plus d’informations, consultez Autorisation de l’accès à l’API de données.	27 octobre 2020
Amazon Aurora étend la prise en charge de l’exportation d’instantanés vers Amazon S3	Vous pouvez désormais exporter des données d’instantané de bases de données vers Amazon S3 dans toutes les Régions AWS commerciales. Pour plus d’informations, consultez Exportation de données d’instantanés de bases de données vers Amazon S3.	22 octobre 2020

La base de données globale Aurora prend en charge le clonage	Vous pouvez désormais créer des clones des clusters de base de données principal e et secondaire de vos bases de données globales Aurora. Vous pouvez le faire en utilisant l'option AWS Management Console et en choisissant l'option de menu Créer un clone. Vous pouvez également utiliser AWS CLI et exécuter la `restore-db-cluster-to-point-in-time` commande avec l'`--restore-type copy-on-write` option. À l'aide du AWS Management Console ou du AWS CLI, vous pouvez également cloner des clusters de bases de données à partir de vos bases de données globales Aurora sur plusieurs AWS comptes. Pour plus d'informations sur le clonage, consultez Clonage d'un volume de cluster de bases de données Aurora.	19 octobre 2020

Amazon Aurora prend en charge le redimensionnement dynamique du volume du cluster	À partir d'Aurora MySQL 1.23 et 2.09, d'Aurora PostgreSQL 3.3.0 et d'Aurora PostgreSQL 2.6.0, Aurora réduit la taille du volume du cluster après avoir supprimé des données via des opérations telles que `DROP TABLE`. Pour tirer parti de cette amélioration, effectuez une mise à niveau vers l'une des versions appropriées en fonction du moteur de base de données utilisé par votre cluster. Pour plus d'informations sur cette fonction et sur la façon de vérifier l'espace de stockage utilisé et disponible pour un cluster Aurora, consultez Gestion des performances et dimensionnement des clusters de bases de données Aurora.	13 octobre 2020
Amazon Aurora prend en charge les tailles de volume jusqu'à 128 Tio	Les volumes de cluster Aurora nouveaux et existants peuvent désormais atteindre une taille maximale de 128 tebibytes (TiB). Pour plus d'informations, consultez Évolutivité du stockage Aurora.	22 septembre 2020

Aurora PostgreSQL prend en charge les classes d'instance de base de données db.r5 et db.t3 dans la région Chine (Ningxia)	Vous pouvez désormais créer des clusters de bases de données Aurora PostgreSQL dans la région Chine (Ningxia), qui utilisent les classes d'instance de base de données db.r5 et db.t3. Pour plus d'informations, consultez Classes d'instance de base de données.	3 septembre 2020

Améliorations des requêtes parallèles Aurora	À partir de Aurora MySQL 2.09 et 1.23, vous pouvez tirer parti des améliorations apportées à la fonction de requête parallèle. La création d'un cluster de requête parallèle ne nécessite plus un mode de moteur spécial. Vous pouvez désormais activer et désactiver les requêtes parallèles à l'aide de l'option de configuration `aurora_parallel_query` pour tout cluster alloué qui exécute une version Aurora MySQL compatible. Vous pouvez mettre à niveau un cluster existant vers une version Aurora MySQL compatible et utiliser une requête parallèle, au lieu de créer un nouveau cluster et d'y importer des données. Vous pouvez utiliser Performance Insights pour les clusters de requête parallèle. Vous pouvez arrêter et démarrer des clusters de requête parallèle. Vous pouvez créer des clusters de requête parallèle Aurora compatibles avec MySQL 5.7. La requête parallèle fonctionne pour les tables qui utilisent le format de ligne `DYNAMIC`. Les clusters de requêtes parallèles peuvent utiliser l'authent	2 septembre 2020

	ification Gestion des identités et des accès AWS (IAM). Les instances de base de données de lecteur dans des clusters de requête parallèle peuvent tirer parti du niveau d'isolement `READ COMMITTED`. Vous pouvez également créer des clusters de requête parallèle dans d'autres Régions AWS. Pour plus d'informations sur la fonction de requête parallèle et ces améliorations, consultez Requête parallèle pour Amazon Aurora My SQL.	
Modifications du paramètre Aurora MySQL binlog_rows_query_log_events	Vous pouvez désormais modifier la valeur du paramètre de configuration Aurora MySQL `binlog_rows_query_log_events`. Auparavant, ce paramètre n'était pas modifiable.	26 août 2020

Prise en charge des mises à niveau automatiques des versions mineures Aurora MySQL	Avec Aurora MySQL, le paramètre Activer la mise à niveau automatique des versions mineures prend désormais effet lorsque vous le spécifiez pour un cluster de bases de données Aurora MySQL. Lorsque vous activez la mise à niveau automatiq ue des versions mineures, Aurora procède à la mise à niveau automatique vers les nouvelles versions mineures à mesure qu'elles sont publiées. Les mises à niveau automatiq ues se produisent pendant la fenêtre de maintenance de la base de données. Pour Aurora MySQL, cette fonction s'applique uniquement à Aurora MySQL version 2, compatible avec MySQL 5.7. Initialement, la procédure de mise à niveau automatiq ue active la version 2.07.2 pour les clusters de base de données Aurora MySQL. Pour plus d'informations sur cette fonction Aurora MySQL, consultez Mises à niveau et correctifs de base de données pour Amazon Aurora MySQL.	3 août 2020

Aurora PostgreSQL prend en charge les mises à niveau de version majeure vers PostgreSQL version 11	Aurora PostgreSQL vous permet désormais de mettre à niveau le moteur de base de données vers une version 11 majeure. Pour plus d'informations, consultez Mise à niveau du moteur de base de données PostgreSQL pour Aurora PostgreSQL.	28 juillet 2020
Amazon Aurora prend en charge AWS PrivateLink	Amazon Aurora prend désormais en charge la création de points de terminaison Amazon VPC pour les appels d'API Amazon RDS afin de maintenir le trafic entre les applications et Aurora sur le réseau. AWS Pour plus d'informations, consultez Amazon Aurora et points de terminaison de VPC d'interface (AWS PrivateLink).	9 juillet 2020

RDS Proxy généralement disponible	RDS Proxy est désormais généralement disponible. Vous pouvez utiliser RDS Proxy avec RDS for MySQL, Aurora MySQL, RDS pour PostgreSQL et Aurora PostgreSQL pour les charges de travail de production. Pour plus d'informations sur le proxy RDS, consultez « Gestion des connexions avec le proxy Amazon RDS » dans le Guide de l'utilisateur Amazon RDS ou le Guide de l'utilisateur Aurora.	30 juin 2020
Transfert d'écriture dans base de données Aurora globale	Vous pouvez désormais activer la capacité d'écriture sur des clusters secondaires dans une base de données globale. Avec le transfert d'écriture, vous émettez des instructions DML sur un cluster secondaire, Aurora transfère l'écriture au cluster principal et les données mises à jour sont répliquées sur tous les clusters secondaires. Pour plus d'informations, voir Transfert d'écriture pour le secondaire Régions AWS avec une base de données globale Aurora.	18 juin 2020

Aurora prend en charge l'intégration avec AWS Backup	Vous pouvez l'utiliser AWS Backup pour gérer les sauvegardes des clusters de bases de données Aurora. Pour plus d'informations, consultez Présentation de la sauvegarde et de la restauration d'un cluster de bases de données Aurora.	10 juin 2020
Aurora PostgreSQL prend en charge les classes d'instance de base de données db.t3.large	Vous pouvez désormais créer des clusters de bases de données Aurora PostgreSQL qui utilisent les classes d'instance de base de données db.t3.large. Pour plus d'informations, consultez Classes d'instance de base de données.	5 juin 2020
Aurora Global Database prend en charge PostgreSQL version 11.7 et l'objectif de point de reprise (RPO) géré	Vous pouvez désormais créer des bases de données globales Aurora pour le moteur de base de données PostgreSQL version 11.7. Vous pouvez également gérer la manière dont une base de données globale PostgreSQL se récupère d'une défaillance à l'aide d'un objectif de point de reprise (RPO). Pour plus d'informations, consultez Reprise après sinistre entre régions pour les bases de données globales Aurora.	4 juin 2020

Aurora MySQL prend en charge la surveillance de base de données avec flux d’activités de base de données	Aurora MySQL inclut désormais des flux d'activité de base de données, qui fournissent un flux de near-real-time données sur l'activité de la base de données dans votre base de données relationnelle. Pour plus d’informations, consultez Utilisation des flux d’activité de base de données.	2 juin 2020
L'éditeur de requêtes est disponible en supplément Régions AWS	L'éditeur de requêtes pour Aurora Serverless est désormais disponible en supplément Régions AWS. Pour plus d’informations, consultez Disponibilité de l’éditeur de requête.	28 mai 2020
L'API de données est disponible en supplément Régions AWS	L'API Data est désormais disponible en supplément Régions AWS. Pour plus d’informations, consultez Disponibilité de l’API de données.	28 mai 2020
Proxy RDS disponible dans la Région Canada (Centre)	Vous pouvez maintenant utiliser la version préliminaire du proxy RDS dans la Région Canada (Centre). Pour plus d’informations sur le proxy RDS, consultez Gestion des connexions avec le proxy Amazon RDS (version préliminaire).	28 mai 2020

Aurora Global Database et réplicas en lecture entre régions	Pour une base de données globale Aurora,vous ne pouvez pas créer un réplica en lecture entre régions Aurora MySQL à partir du cluster principal dans la même région qu'un cluster secondaire. Pour de plus amples information sur Aurora Global Database et les réplicas en lecture entre régions, consultez Utilisation d'Amazon Aurora Global Database et Réplication de bases de données Amazon Aurora MySQL.	18 mai 2020
Proxy RDS disponible en plusieurs versions Régions AWS	Vous pouvez désormais utiliser la version préliminaire de RDS Proxy dans les régions USA Ouest (Californie du Nord), Europe (Londres), Europe (Francfort), Asie-Pacifique (Séoul), Asie-Pacifique (Mumbai), Asie-Pacifique (Singapour) et Asie-Pacifique (Sydney). Pour plus d'informations sur le proxy RDS, consultez Gestion des connexions avec le proxy Amazon RDS (version préliminaire).	13 mai 2020

Édition compatible avec Aurora PostgreSQL prend en charge Microsoft Active Directory sur site ou auto-hébergé	Vous pouvez désormais utiliser un Active Directory sur site ou auto-hébergé pour l'authentification Kerberos des utilisateurs lorsqu'ils se connectent à vos clusters de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez Utilisation de l'authentification Kerberos avec Aurora PostgreSQL.	7 mai 2020
Les clusters multi-maîtres Aurora MySQL sont disponibles en plusieurs versions Régions AWS	Vous pouvez désormais créer des clusters à plusieurs maîtres Aurora dans les régions Asie-Pacifique (Séoul), Asie-Pacifique (Tokyo), Asie-Pacifique (Mumbai) et Europe (Francfort). Pour plus d'informations sur les clusters multi-maîtres, consultez Utilisation des clusters Aurora multi-maîtres.	7 mai 2020
Performance Insights prend en charge l'analyse des statistiques des requêtes Aurora MySQL en cours d'exécution	Il est désormais possible d'analyser les statistiques des requêtes en cours d'exécution à l'aide de Performance Insights pour les instances de bases de données Aurora MySQL. Pour plus d'informations, consultez Analyse des statistiques pour les requêtes en cours d'exécution.	5 mai 2020

Disponibilité générale de la bibliothèque client Java pour l'API de données	La bibliothèque client Java pour l'API de données est désormais disponible pour tous. Vous pouvez télécharger et utiliser une bibliothèque client Java pour l'API Data. Elle permet de mapper les classes côté client aux demandes et réponses de l'API de données. Pour plus d'informations, consultez Utilisation de la bibliothèque client Java pour l'API de données.	30 avril 2020
Amazon Aurora est disponible dans la région Europe (Milan)	Amazon Aurora est désormais disponible dans la région Europe (Milan). Pour plus d'informations, consultez Régions et zones de disponibilité.	28 avril 2020
Amazon Aurora est disponible dans la région Europe (Milan)	Amazon Aurora est désormais disponible dans la région Europe (Milan). Pour plus d'informations, consultez Régions et zones de disponibilité.	27 avril 2020
Amazon Aurora est disponible dans la région Afrique (Le Cap)	Amazon Aurora est désormais disponible dans la région Afrique (Le Cap). Pour plus d'informations, consultez Régions et zones de disponibilité.	22 avril 2020

Aurora PostgreSQL prend désormais en charge les classes d'instance de base de données db.r5.16xlarge et db.r5.8xlarge	Vous pouvez désormais créer des clusters de bases de données Aurora PostgreSQL exécutant PostgreSQL qui utilisent les classes d'instance de base de données db.r5.16xlarge et db.r5.8xlarge. Pour plus d'informations, consultez l'article relatif aux spécifications matérielles de toutes les classes d'instance de base de données pour Aurora.	8 avril 2020
Proxy Amazon RDS pour PostgreSQL	Le proxy Amazon RDS est désormais disponible pour PostgreSQL. Vous pouvez utiliser le proxy RDS pour réduire la surcharge de connexions sur votre cluster ainsi que le risque d'erreurs liées à un trop grand nombre de connexions. Le proxy RDS est actuellement disponible en version préliminaire pour PostgreSQL. Pour plus d'informations, consultez Gestion des connexions avec le proxy Amazon RDS (version préliminaire).	8 avril 2020

Les bases de données globales Aurora prennent désormais en charge Aurora PostgreSQL	Vous pouvez désormais créer des bases de données globales Aurora pour le moteur de base de données PostgreSQL. Une base de données mondiale Aurora couvre plusieurs bases de données Régions AWS, ce qui permet des lectures globales à faible latence et une reprise après sinistre en cas de panne à l'échelle de la région. Pour plus d'informations, consultez Utilisation d'Amazon Aurora Global Database.	10 mars 2020
Prise en charge des mises à niveau de version majeure pour Aurora PostgreSQL	Aurora PostgreSQL vous permet désormais de mettre à niveau le moteur de base de données vers une version majeure. En procédant ainsi, vous pouvez passer à une nouvelle version majeure lorsque vous mettez à niveau certaines versions de moteur PostgreSQL. Pour plus d'informations, consultez Mise à niveau du moteur de base de données PostgreSQL pour Aurora PostgreSQL.	4 mars 2020

Aurora PostgreSQL prend en charge l’authentification Kerberos	Vous pouvez désormais utiliser l’authentification Kerberos pour authentifier les utilisateurs qui se connecten t à vos clusters de bases de données Aurora PostgreSQ L. Pour plus d’informations, consultez Utilisation de l’authentification Kerberos avec Aurora PostgreSQL.	28 février 2020
L'API Data prend en charge AWS PrivateLink	L'API Data prend désormais en charge la création de points de terminaison Amazon VPC pour les appels d'API de données afin de maintenir le trafic entre les applicati ons et l'API de données sur le réseau. AWS Pour plus d’informations, consultez Création d’un point de terminaison de VPC Amazon (AWS PrivateLink) pour l’API de données.	6 février 2020

Prise en charge du machine learning Aurora dans Aurora PostgreSQL	L'extension `aws_ml` Aurora PostgreSQL fournit des fonctions que vous utilisez dans vos requêtes de base de données pour appeler Amazon Comprehend pour l'analyse des sentiments SageMaker et l'IA pour exécuter vos propres modèles de machine learning. Pour plus d'informations, consultez Utilisation des fonctionnalités du Machine Learning (ML) avec Aurora.	5 février 2020
Aurora PostgreSQL prend en charge l'exportation de données vers Amazon S3	Vous pouvez interroger des données à partir d'un cluster de bases de données Aurora PostgreSQL et les exporter directement dans des fichiers stockés dans un compartiment Amazon S3. Pour plus d'informations, consultez Exportation de données d'un cluster de bases de données PostgreSQL Aurora vers Amazon S3.	5 février 2020

Prise en charge de l’exportation des données d’instantanés de bases de données vers Amazon S3	Amazon Aurora prend en charge l’exportation des données d’instantanés de bases de données vers Amazon S3 pour MySQL et PostgreSQL. Pour plus d’informations, consultez Exportation de données d’instantanés de bases de données vers Amazon S3.	9 janvier 2020
Notes de mises à jour d’Aurora MySQL dans l’historique de document	Cette section inclut désormais l’historique des notes de mise à jour Édition compatible avec Aurora MySQL pour les versions publiées après le 31 août 2018. Pour obtenir les notes de mise à jour complètes d’une version spécifique, sélectionnez le lien dans la première colonne de l’entrée de l’historique.	13 décembre 2019

Proxy Amazon RDS	Vous pouvez réduire la surcharge de gestion des connexions sur votre cluster et réduire le risque d'erreurs liées au « nombre de connexion s trop élevé » à l'aide du proxy Amazon RDS. Vous associez chaque proxy à une instance de base de données RDS ou un cluster de bases de données Aurora. Ensuite, vous utilisez le point de terminaison du proxy dans la chaîne de connexion de votre application. Le proxy Amazon RDS est actuellem ent en version préliminaire publique. Il prend en charge le moteur de base de données Aurora MySQL. Pour plus d'informations, consultez Gestion des connexions avec le proxy Amazon RDS (version préliminaire).	3 décembre 2019
L'API de données pour Aurora Serverless v1 prend en charge les indicateurs de mappage de type de données.	Désormais, vous pouvez utiliser un indicateur pour demander à l'API Data pour Aurora Serverless v1 d'envoyer une valeur `String` à la base de données comme type différent. Pour plus d'informations, consultez Appel à l'API de données.	26 novembre 2019

L’API de données pour Aurora Serverless v1 prend en charge une bibliothèque client Java (version préliminaire)	Vous pouvez télécharger et utiliser une bibliothèque client Java pour l’API Data. Elle permet de mapper les classes côté client aux demandes et réponses de l’API de données. Pour plus d’informations, consultez Utilisation de la bibliothèque client Java pour l’API de données.	26 novembre 2019
Aurora PostgreSQL est éligible FedRAMP HIGH	Aurora PostgreSQL est éligible FedRAMP HIGH. Pour plus de détails sur AWS les efforts de conformité et les efforts de mise en conformité, voir les AWS services concernés par le programme de conformité.	26 novembre 2019

Niveau d’isolement READ COMMITTED activé pour les réplicas Amazon Aurora MySQL	Vous pouvez désormais activer le niveau d’isolement `READ COMMITTED` sur les réplicas Aurora MySQL. Cette action nécessite l’activation du paramètre de configuration `aurora_read_replica_read_committed_isolation_enabled` au niveau de la session. L’utilisation du niveau d’isolement `READ COMMITTED` pour les requêtes de longue durée sur les clusters OLTP peut contribuer à résoudre les problèmes liés à la longueur des listes d’historique. Avant d’activer ce paramètre, assurez-vous de comprendre comment le comportement d’isolement sur les réplicas Aurora diffère de l’implémentation MySQL traditionnelle de `READ COMMITTED` . Pour plus d’informations, consultez Niveaux d’isolement Aurora MySQL.	25 novembre 2019

Performance Insights prend en charge l’analyse statistique de l’exécution de requêtes Aurora PostgreSQL.	Il est désormais possible d’analyser les statistiques des requêtes en cours d’exécuti on à l’aide de Performance Insights pour les instances de base de données Aurora PostgreSQL. Pour plus d’informations, consultez Analyse des statistiques pour les requêtes en cours d’exécution.	25 novembre 2019
Plus de clusters dans une base de données globale Aurora	Vous pouvez maintenan t ajouter plusieurs régions secondaires à une base de données globale Aurora. Vous pouvez tirer parti des lectures globales à faible latence et la reprise après sinistre sur une aire géographique plus étendue. Pour plus d’informa tions concernant les bases de données globales Aurora, consultez Utilisation de bases de données Amazon Aurora globales.	25 novembre 2019

Prise en charge du machine learning dans Aurora MySQL	Dans Aurora MySQL 2.07 et versions ultérieures, vous pouvez faire appel à Amazon Comprehend pour l'analyse des sentiments SageMaker et à l'IA pour un large éventail d'algorithmes d'apprentissage automatique. Vous utilisez directement les résultats dans votre application de base de données en intégrant les appels aux fonctions stockées à vos requêtes. Pour plus d'informations, consultez Utilisation des fonctionnalités du Machine Learning (ML) avec Aurora.	25 novembre 2019

La base de données globale Aurora ne requiert plus le paramètre relatif au mode moteur	Il n’est plus nécessaire de spécifier `--engine-mode=global` lors de la création d’un cluster destiné à faire partie d’une base de données globale Aurora. Tous les clusters Aurora qui satisfont aux exigences de compatibilité sont éligibles pour faire partie d’une base de données globale. Par exemple, le cluster doit actuellement utiliser Aurora MySQL version 1 compatible MySQL 5.6. Pour plus d’informations concernant les bases de données globales Aurora, consultez Utilisation de bases de données Amazon Aurora globales.	25 novembre 2019

La base de données globale Aurora est disponible pour Aurora MySQL version 2	Depuis Aurora MySQL 2.07, vous pouvez créer une base de données globale Aurora compatible avec MySQL 5.7. Il n'est pas nécessaire de spécifier le mode du moteur `global` pour les clusters principaux ou secondaires. Vous pouvez ajouter un nouveau cluster provisionné avec Aurora MySQL 2.07 ou version ultérieure à une base de données Aurora Global Database. Pour plus d'informations concernant les bases de données Aurora Global Database, consultez Utilisation des bases de données Amazon Aurora Global Database.	25 novembre 2019
Optimisation des conflits entre les lignes critiques Aurora MySQL disponible sans le mode Lab	L'optimisation des conflits entre les lignes critiques est désormais disponible pour Aurora MySQL et ne nécessite pas que le mode lab Aurora soit activé. Cette fonction améliore nettement le débit pour les charges où de nombreuses transactions sont en conflit pour les lignes d'une même page. Cette amélioration suppose de modifier l'algorithme de libération de verrou utilisé par Aurora MySQL.	19 novembre 2019

Jointures par hachage Aurora MySQL disponibles sans mode Lab	La fonctionnalité de jointure de hachage est désormais disponible pour Aurora MySQL et ne nécessite pas que le mode lab Aurora soit activé. Cette fonctionnalité peut améliorer les performances de requêtes lorsque vous devez joindre une grande quantité de données au moyen d'une équijointure. Pour plus d'informations sur l'utilisation de cette fonctionnalité, consultez Utilisation des jointures de hachage dans Aurora MySQL.	19 novembre 2019
Aurora MySQL 2.* Prise en charge de classes d'instance db.r5 supplémentaires	Les clusters Aurora MySQL prennent désormais en charge les types d'instance db.r5.8xlarge, db.r5.16xlarge et db.r5.24xlarge. Pour plus d'informations sur les types d'instance pour les clusters Aurora MySQL, consultez Choix de la classe d'instance de base de données.	19 novembre 2019

Aurora MySQL 2.* Prise en charge du retour sur trace	Les versions Aurora MySQL 2.* offrent désormais un moyen rapide de récupérer après des erreurs utilisateur, comme la suppression non souhaitée d'une table ou d'une ligne. Le retour sur trace vous permet de déplacer votre base de données vers un point précédent dans le temps sans nécessiter la restauration à partir d'une sauvegarde. Il s'exécute en quelques secondes, même pour les bases de données volumineuses. Pour plus d'informations, consultez Retour sur trace d'un cluster de bases de données Aurora.	19 novembre 2019

Prise en charge des balises de facturation pour Aurora	Vous pouvez désormais utiliser des balises pour conserver une trace de l'allocation des coûts pour les ressources telles que les clusters Aurora, les instances de base de données au sein des clusters Aurora, les I/O, les sauvegardes, les instantanés, etc. Vous pouvez consulter les coûts associés à chaque balise à l'aide de AWS Cost Explorer. Pour plus d'informations sur l'utilisation de balises avec Aurora, consultez Balisage des ressources Amazon RDS. Pour obtenir des informations générales sur les balises et sur la façon de les utiliser pour les analyses de coûts, consultez Utilisation des balises d'allocation des coûts et Balises d'allocation des coûts définies par l'utilisateur.	23 octobre 2019
API de données pour Aurora PostgreSQL	Aurora PostgreSQL prend désormais en charge l'utilisation de l'API Data avec les clusters de bases de données Amazon Aurora Serverless v1. Pour plus d'informations, consultez Utilisation de l'API Data pour Aurora Serverless v1.	23 septembre 2019

Aurora PostgreSQL prend en charge le téléchargement des journaux de base de données vers les journaux CloudWatch	Vous pouvez configurer votre cluster de base de données Aurora PostgreSQL pour publier les données de journal dans un groupe de journaux dans Amazon Logs. CloudWatch Avec CloudWatch Logs, vous pouvez effectuer une analyse en temps réel des données du journal, puis les utiliser CloudWatch pour créer des alarmes et afficher des métriques. Vous pouvez utiliser CloudWatch les journaux pour stocker vos enregistrements de journal dans un espace de stockage hautement durable. Pour plus d'informations, consultez la section Publication des journaux Aurora PostgreSQL sur Amazon Logs. CloudWatch	9 août 2019
Clusters multi-maîtres pour Aurora MySQL	Vous pouvez configurer des clusters multi-maîtres Aurora MySQL. Dans ces clusters, chaque instance de base de données possède des capacités de lecture/écriture. Pour plus d'informations, consultez Utilisation des clusters multi-maîtres Aurora.	8 août 2019

Aurora PostgreSQL prend en charge Aurora Serverless v1	Vous pouvez désormais utiliser Amazon Aurora Serverless v1 avec Aurora PostgreSQL. Un cluster de bases de données Aurora sans serveur démarre, s'arrête et augmente ou réduit sa capacité de calcul en fonction des besoins de votre application, le tout de manière automatique. Pour plus d'informations, consultez Utilisation de Amazon Aurora Serverless v1.	9 juillet 2019
Clonage entre comptes pour Aurora MySQL	Vous pouvez désormais cloner le volume de cluster d'un cluster de base de données Aurora MySQL entre AWS comptes. Vous autorisez le partage via AWS Resource Access Manager (AWS RAM). Le volume de cluster cloné utilise un copy-on-write mécanisme qui ne nécessite de stockage supplémen taire que pour les données nouvelles ou modifiées. Pour plus d'informations sur le clonage d'Aurora, consultez Clonage de base de données dans un cluster de bases de données Aurora.	2 juillet 2019

Aurora PostgreSQL prend en charge les classes d'instance de base de données db.t3	Vous pouvez désormais créer des clusters de bases de données Aurora PostgreSQL utilisant les classes d'instance de base de données db.t3. Pour plus d'informations, consultez Classe d'instance de base de données.	20 juin 2019
Prise en charge de l'importation de données de Amazon S3 pour Aurora PostgreSQL	Vous pouvez désormais importer les données des fichiers Amazon S3 dans une table de cluster de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez Importation de données Amazon S3 dans un cluster de bases de données Aurora PostgreSQL.	19 juin 2019
Aurora PostgreSQL permet désormais une récupération avec basculement rapide avec gestion des caches de clusters	Aurora PostgreSQL fournit désormais la gestion des caches de clusters pour garantir une récupération rapide de l'instance de base de données principale en cas de basculement. Pour plus d'informations, consultez Récupération rapide après basculement avec la gestion des caches de clusters.	11 juin 2019

API de données pour Aurora Serverless v1 globalement disponible	Vous pouvez accéder aux clusters Aurora Serverless v1 avec les applications de services Web à l'aide de l'API Data. Pour plus d'informations, consultez Utilisation de l'API Data pour Aurora Serverless v1.	30 mai 2019
Aurora PostgreSQL prend en charge la surveillance de base de données avec flux d'activités de la base de données	Aurora PostgreSQL inclut désormais des flux d'activité de base de données, qui fournissent near-real-time un flux de données sur l'activité de la base de données dans votre base de données relationnelle. Pour plus d'informations, consultez Utilisation des flux d'activité de base de données.	30 mai 2019
Recommandations concernant Amazon Aurora	Amazon Aurora fournit désormais des recommandations automatisées pour les ressources Aurora. Pour plus d'informations, consultez Utilisations des recommandations Amazon Aurora.	22 mai 2019

Performance Insights prend en charge les bases de données globales Aurora	Vous pouvez désormais utiliser Performance Insights pour les bases de données globales Aurora. Pour plus d'informations sur Performance Insights pour Aurora, consultez Utilisation de l'analyse des performances d'Amazon RDS. Pour plus d'informations concernant les bases de données globales Aurora, consultez Utilisation des bases de données globalesAurora.	13 mai 2019
Performance Insights est désormais disponible pour Aurora MySQL 5.7	Amazon RDS Performance Insights est désormais disponible pour les versions Aurora MySQL 2.x, qui sont compatibles avec MySQL 5.7. Pour plus d'informations, consultez Utilisation de l'analyse des performances d'Amazon RDS.	3 mai 2019
Les bases de données mondiales Aurora sont disponibles en plus Régions AWS	Vous pouvez désormais créer des bases de données globales Aurora dans la plupart des Régions AWS sites où Aurora est disponible. Pour plus d'informations concernant les bases de données globales Aurora, consultez Utilisation de bases de données Amazon Aurora globales.	30 avril 2019

Capacité minimale de 1 pour Aurora Serverless v1	La capacité minimale que vous pouvez utiliser pour un cluster Aurora Serverless v1 est 1. Auparavant, la capacité minimale était 2. Pour plus d'informations sur la spécification de valeurs de capacité Aurora sans serveur, consultez Définition de la capacité d'un cluster de bases de données Aurora Serverless v1.	29 avril 2019
Action de délai d'attente Aurora Serverless v1	Vous pouvez désormais spécifier l'action à effectuer lorsqu'une modification de la capacité Aurora Serverless v1 expire. Pour plus d'informations, vous pouvez consulter la section relative à l'action d'expiration en cas de modification de la capacité.	29 avril 2019
Facturation à la seconde	Amazon RDS est désormais facturé par tranches d'une seconde partout Régions AWS sauf AWS GovCloud aux États-Unis pour les instances à la demande. Pour plus d'informations, consultez Facturation des instances de base de données pour Aurora.	25 avril 2019

Partage d'Aurora Serverless v1instantanés entre Régions AWS	Avec Aurora Serverless v1, les instantanés sont toujours chiffrés. Si vous chiffrez l'instantané avec le vôtre AWS KMS key, vous pouvez désormais le copier ou le partager. Régions AWS Pour en savoir plus sur les instantanés de clusters de bases de données Aurora Serverless v1, consultez la section Aurora Serverless v1 et instantanés.	17 avril 2019
Restauration des sauvegardes MySQL 5.7 à partir d'Amazon S3	Vous pouvez désormais créer une sauvegarde de votre base de données MySQL 5.7, la stocker sur Amazon S3, puis restaurer le fichier de sauvegarde sur un nouveau cluster de bases de données Aurora MySQL. Pour plus d'informations, consultez Migration des données d'une base de données MySQL externe vers un cluster de bases de données Aurora MySQL.	17 avril 2019

Partage d’instantanés Aurora Serverless v1 entre régions	Avec Aurora Serverless v1, les instantanés sont toujours chiffrés. Si vous chiffrez l'instantané avec le vôtre AWS KMS key, vous pouvez désormais le copier ou le partager entre les régions. Pour plus d’informations sur les instantanés des clusters de bases de données Aurora Serverless v1, consultez Aurora sans serveur et instantanés.	16 avril 2019
proof-of-conceptTutoriel Aurora	Vous pouvez apprendre à exécuter une preuve de concept afin de tester votre application et votre charge de travail avec Aurora. Pour accéder au didacticiel complet, consultez la page relative à l’exécution d’une preuve de concept Aurora.	16 avril 2019
Aurora Serverless v1 prend en charge la restauration depuis une sauvegarde Amazon S3	Vous pouvez désormais importer des sauvegardes depuis Amazon S3 vers un cluster Aurora Serverles s. Pour plus de détails sur cette procédure, consultez Migration des données à partir de MySQL en utilisant un compartiment Amazon S3.	16 avril 2019

Nouveaux paramètres modifiables pour Aurora Serverless v1	Vous pouvez désormais modifier les paramètres de base de données suivants pour un cluster Aurora Serverless v1 : `innodb_file_format`, `innodb_file_per_table`, `innodb_large_prefix`, `innodb_lock_wait_timeout`, `innodb_monitor_disable`, `innodb_monitor_enable`, `innodb_monitor_reset`, `innodb_monitor_reset_all`, `innodb_print_all_deadlocks`, `log_warnings`, `net_read_timeout`, `net_retry_count`, `net_write_timeout`, `sql_mode` et `tx_isolation`. Pour en savoir plus sur les paramètres de configuration des clusters Aurora Serverless v1, consultez la section Aurora Serverless v1 et groupes de paramètres.	4 avril 2019

Aurora PostgreSQL prend en charge les classes d'instance de base de données db.r5	Vous pouvez désormais créer des clusters de bases de données Aurora PostgreSQL utilisant les classes d'instance de base de données db.r5. Pour plus d'informations, consultez Classe d'instance de base de données.	4 avril 2019
Réplication logique Aurora PostgreSQL	La réplication logique PostgreSQL vous permet de répliquer des parties d'une base de données pour un cluster de bases de données Aurora PostgreSQL. Pour plus d'informations, consultez Utilisation de la réplication logique PostgreSQL.	28 mars 2019

Prise en charge des identifiants de transaction globaux (GTID) pour Aurora MySQL 2.04	Vous pouvez utiliser la réplication avec la fonction d'identifiant de transaction global (GTID) MySQL 5.7. Cette fonction simplifie la réplication des journaux binaires (binlog) entre Aurora MySQL et une base de données externe compatible avec MySQL 5.7. La réplication peut utiliser le cluster Aurora MySQL en tant que source ou en tant que destination. Cette fonction est disponible pour Aurora MySQL 2.04 et les versions ultérieures. Pour plus d'informations sur la réplication basée sur les identifiants de transaction globaux (GTID) et Aurora MySQL, consultez Utilisation de la réplication basée sur des identifiants de transaction globaux (GTID) pourAurora MySQL.	25 mars 2019

Téléchargement de Aurora Serverless v1 journaux sur Amazon CloudWatch	Vous pouvez désormais demander à Aurora de télécharger les journaux de base de données CloudWatch d'un Aurora Serverless v1 cluster. Pour plus d'informations, consultez Affichage de clusters de bases de données Aurora sans serveur. Cette amélioration vous permet désormais de définir des valeurs pour les paramètres de niveau instance dans un groupe de paramètres de cluster de bases de données, et ces valeurs s'appliquent à toutes les instances de base de données du cluster, sauf si vous les remplacez dans le groupe de paramètres de base de données. Pour plus d'informations, consultez Utilisation de groupes de paramètres de base de données et de groupes de paramètres de cluster de bases de données.	25 février 2019
Aurora MySQL prend en charge les classes d'instance de base de données db.t3	Vous pouvez désormais créer des clusters de bases de données Aurora MySQL utilisant les classes d'instance de base de données db.t3. Pour plus d'informations, consultez Classe d'instance de base de données.	25 février 2019

Aurora MySQL prend en charge les classes d'instance de base de données db.r5	Vous pouvez désormais créer des clusters de bases de données Aurora MySQL utilisant les classes d'instance de base de données db.r5. Pour plus d'informations, consultez Classe d'instance de base de données.	25 février 2019
Compteurs Performance Insights pour Aurora MySQL	Vous pouvez désormais ajouter des compteurs de performances à vos graphiques Performance Insights pour les instances de base de données Aurora MySQL. Pour plus d'informations, consultez Composants du tableau de bord de Performance Insights.	19 février 2019
Amazon RDS Performance Insights prend en charge l'affichage de texte SQL supplémentaire pour Aurora MySQL	Amazon RDS Performance Insights prend désormais en charge l'affichage de texte SQL supplémentaire dans le tableau de bord Performance Insights pour les instances de base de données Aurora MySQL. Pour plus d'informations, consultez Affichage de texte SQL supplémentaire sur le tableau de bord de Performance Insights.	6 février 2019

Amazon RDS Performance Insights prend en charge l'affichage de texte SQL supplémentaire pour Aurora PostgreSQL	Amazon RDS Performance Insights prend désormais en charge l'affichage de texte SQL supplémentaire dans le tableau de bord Performance Insights pour les instances de base de données Aurora PostgreSQL. Pour plus d'informations, consultez Affichage de texte SQL supplémentaire sur le tableau de bord de Performance Insights.	24 janvier 2019
Facturation des sauvegardes Aurora	Vous pouvez utiliser les CloudWatch métriques Amazon `TotalBackupStorageBilled` `SnapshotStorageUsed` , et `BackupRetentionPeriodStorageUsed` pour surveiller l'utilisation de l'espace de vos sauvegardes Aurora. Pour plus d'informations sur l'utilisation CloudWatch des métriques, voir Présentation de la surveillance. Pour plus d'informations sur la gestion du stockage pour les données de sauvegarde, consultez Présentation de l'utilisation du stockage de sauvegarde Aurora.	3 janvier 2019

Compteurs Performance Insights	Vous pouvez désormais ajouter des compteurs de performances à vos graphiques Performance Insights. Pour plus d'informations, consultez Composants du tableau de bord de Performance Insights.	6 décembre 2018
Aurora Global Database	Vous pouvez désormais créer des bases de données globales Aurora. Une base de données mondiale Aurora couvre plusieurs bases de données Régions AWS, ce qui permet des lectures globales à faible latence et une reprise après sinistre en cas de panne à l'échelle de la région. Pour plus d'informations, consultez Utilisation d'Amazon Aurora Global Database.	28 novembre 2018
Gestion de plans de requêtes dans Aurora PostgreSQL	Aurora PostgreSQL assure désormais la gestion des plans de requêtes, qui est une fonctionnalité vous permettant de gérer les plans d'exécution de requêtes PostgreSQL. Pour plus d'informations, consultez Gestion des plans d'exécution de requêtes pour Aurora PostgreSQL.	20 novembre 2018

Éditeur de requête pour Aurora Serverless v1 (version bêta)	Vous pouvez exécuter les instructions SQL dans la console Amazon RDS sur les clusters Aurora Serverless v1. Pour plus d'informations, consultez Utilisation de l'éditeur de requête pour Aurora Serverless v1.	20 novembre 2018
API Data pour Aurora Serverless v1 (bêta)	Vous pouvez accéder aux clusters Aurora Serverless v1 avec les applications de services Web à l'aide de l'API Data. Pour plus d'informations, consultez Utilisation de l'API de données pour Aurora sans serveur.	20 novembre 2018
Prise en charge de TLS pour Aurora Serverless v1	Aurora Serverless v1les clusters prennent en charge TLS/SSL le chiffrement. Pour plus d'informations, consultez Chiffrement TLS/SSL pour Aurora sans serveur.	19 novembre 2018

Points de terminaison personnalisés	Vous pouvez désormais créer des points de terminaison associés à un ensemble arbitraire d'instances de base de données. Cette fonction contribue à l'équilibrage de charge et à la haute disponibilité des clusters Aurora dans lesquels certaines instances de base de données ont une capacité ou une configuration distincte à celle des autres. Vous pouvez utiliser des points de terminaison personnalisés au lieu de la connexion à une instance de base de données spécifique via le point de terminaison de son instance. Pour plus d'informations, consultez Points de terminaison Amazon Aurora.	12 novembre 2018
Prise en charge de l'authentification IAM dans Aurora PostgreSQL	Aurora PostgreSQL prend désormais en charge l'authentification IAM. Pour plus d'informations, consultez Authentification de base de données IAM.	le 8 novembre 2018

Groupes de paramètres personnalisés pour la restauration et la restauration à un instant dans le passé	Vous pouvez désormais spécifier un groupe de paramètres personnalisés lorsque vous restaurez un instantané ou procédez à une opération de restauration à un instant dans le passé. Pour plus d'informations, consultez Restauration à partir d'un instantané de cluster de bases de données et Restauration d'un cluster de bases de données à une date spécifiée.	15 octobre 2018
Protection contre la suppression pour les clusters de bases de données Aurora	Lorsque vous activez la protection contre la suppression pour un cluster de bases de données, la base de données ne peut être supprimée par aucun utilisateur. Pour en savoir plus, consultez la section Deleting a DB cluster.	26 septembre 2018
Fonction d'arrêt et de démarrage Aurora	Vous pouvez désormais arrêter ou démarrer un cluster Aurora complet en une seule opération. Pour plus d'informations, consultez Arrêt et démarrage d'un cluster Aurora.	24 septembre 2018

Fonction de requête parallèle pour Aurora MySQL	Aurora MySQL propose désormais une option permettant de paralléliser le I/O travail pour les requêtes au sein de l'infrastructure de stockage Aurora. Cette fonction permet d'accélérer les requêtes analytiques à usage intensif de données, qui sont souvent les opérations qui prennent le plus de temps dans une charge de travail. Pour plus d'informations, consultez Requête parallèle pour Aurora MySQL.	20 septembre 2018
Nouveau guide	Il s'agit de la première version du Guide de l'utilisateur Amazon Aurora.	31 août 2018

AWSGlossaire

Pour la AWS terminologie la plus récente, consultez le AWSglossaire dans la Glossaire AWSréférence.

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.